Kryptologie Zusammenfassung

July 4, 2024

Contents

1 Einführung

Im Folgenden sei m eine Nachricht, E ein Verschlüsselungsverfahren, das m und einen geheimen Schlüssel k als Argumente nimmt und den Code $c = E(m, k)$ berechnet. Der Empfänger verwendet k und einen Dechiffrieralgorithmus D, um $D(c, k) = m$ zu berechnen. Wie der Schlüssel k zwischen den beiden Parteien ausgetauscht werden kann, ist ein anderes Problem, das hier als gelöst angenommen werde.

1.1 Transpositions- und Substitutionschiffre

Bei Transpositionschiffren bleiben die Klartextbuchstaben unverändert, es wird nur ihre Position im Chiffre verändert (d.h. es wird ein Anagramm erstellt). Bei einer Substitutionschiffre werden die Buchstaben durch andere ersetzt (z.B. Cäsarverschlüsselung).

1.2 Mono- und Polyalphabetische Chiffre

Bei monoalphabetischen Chiffren wird ein Klartextbuchstaben immer durch das selbe Zeichen ersetzt (z.B. Cäsar Chiffre). Bei polyalphabetischen Chiffren werden nicht immer die selben Zeichen zur Verschlüsselung eines Buchstabens verwendet.

1.3 Block- und Stromchiffren

Bei Blockchiffren werden immer Blöcke von Klartextbuchstaben zusammen chiffriert und die Verschlüsselung zwischen zwei Blöcken unterscheiden sich jeweils. Bei Stromchiffren hat der Verschlüsselungsalgorithmus einen inneren Zustand, der sich bei jeder Verschlüsselung ändert und Einfluss auf die Verschlüsselung der nächsten Buchstaben ausübt. Wenn eine Veränderung eines einzelnen Buchstabens im Klartext zu großen Veränderungen in der Chiffre führt, spricht man von Lawineneffekt.

1.4 Moderne Kryptographie

Das Entscheidende ist, dass jedwede Kommunikation über einen abhörbaren Kanal verläuft. Es wird daher wichtig sein, dass es keinen Schlüsselaustausch im Voraus benötigt.

2 Historische Chiffren

2.1 Cäsar Chiffre

Hierbei wird das Alphabet durchnummeriert mit Zahlen von 0 bis 25. Das Verfahren E ist dann definiert durch

$$E(x) = x + 3 \mod 26$$

Man kann das mit 26 verschiedenen Schlüsseln verallgemeinern mit dem Verfahren

$$E(x) = x + k \mod 26$$

Die Umkehrfunktion ist

$$D(x) = c - k \mod 26$$

Für $k = 13$ ist $E \equiv D$.

2.2 Affine Chiffre

Der Schlüssel besteht aus zwei Komponenten $k = (a, b)$ und man berechnet

$$E(x, k) = ax + b \mod 26$$

Damit E injektiv ist, muss $ggT(a, 26) = 1$. Das heißt $a \in \mathbb{Z}_n^*$. Da jedes Element ein Inverses hat, ist der Code eindeutig dechiffrierbar. Insbesondere ist D auch ein affines Chiffre.

2.3 Allgemeine Monoalphabetische Chiffre

Jeder Buchstabe wird auf genau einen anderen gemappt. Es gibt daher 26! viele Schlüssel. Das Sicherheitsniveau ist aber nicht 88 Bits. Ab einer Textlänge von ca 50 Buchstaben kann der Text mittels Häufigkeitsanalyse entschlüsselt werden.

2.4 Homophone Chiffre

Jedem Buchstaben wird eine Menge an möglichen Verschlüsselungen zugeordnet, die abhängig von der Häufigkeit des Klartextbuchstabens sind. Um einen Buchstaben zu verschlüsseln, wird aus der Menge der möglichen Cypher zufällig ein Zeichen gezogen.

2.5 Playfair Chiffre

Hierbei werden immer zwei aufeinanderfolgende Zeichen (Bigramme) verschlüsselt. Man kann damit aber kein Bigramm verschlüsseln, das aus zwei gleichen Buchstaben besteht. Das kann aber leicht gelöst werden, indem im Klartext doppelte Buchstaben entfernt werden. Man identifiziert zwei Buchstaben miteinandern (z.B. I und J) und wählt ein Geheimwort der Länge 5 mit paarweise verschiedenen Buchstaben. Damit wird eine 5×5 Tabelle gefüllt (Geheimwort in die erste Zeile und das restliche Alphabet darunter). Wenn die Buchstaben des zu verschlüsselnen Bigramms in der selben Zeile sind, wird der nächste Buchstabe dieser Zeile verwendet (bei Spalten analog). Ansonsten bildet man mit den beiden Buchstaben ein Rechteck wählt im Uhrzeigersinn die jeweils anderen Buchstaben, die in den Ecken des Rechtecks stehen.

2.6 Vigenère Chiffre

Man wählt ein Schlüsselwort und schreibt es fortlaufend unter den Klartext. Man fasst die Buchstaben als Zahlen auf und addiert sie bitweise modulo 26. Dass ein Trigramm mehrfach vorkommt ist sehr selten und daher ist das Chiffre sicher.
Diese Überlegung ist aber der Ansatz zur Lösung. Wenn es ein Trigramm gibt, das mehrfach vorkommt, dann wird jeweils der Abstand zwischen zwei gleichen Trigrammen berechnet. Es ist dann sehr wahrscheinlich, dass der größte gemeinsame Teiler der Abstände, die Länge des Schlüsselworts ist. Anschließend ist es nur noch ein Cäsar Chiffre.

2.7 ENIGMA

Das ist ein polyalphabetisches Stromchiffre. Das bedeutet, es gibt einen inneren Zustand, der sich bei jeder Verschlüsselung ändert.

3 Koinzidenzindex

Sei $p = (p_1, p_2, ..., p_n)$ eine diskrete Wahrscheinlichkeitsverteilung. Dann ist der Koinzidenzindex von p definiert als

$$IC(p) = \sum_{i=1}^{n} p_i^2$$

Der kleinstmögliche Wert für $IC(p)$ ist $\frac{1}{n}$ bei einer Gleichverteilung. Der größtmögliche Wert ist 1. Wir wollen nun den Wert schätzen.
Es sei ein Text gegeben und es gebe eine Wahrscheinlichkeitsverteilung, nach der die

Buchstaben im Text verteilt seien. Eine erwartungstreue Schätzung ist dann

$$IC = \sum_a \frac{n(a)}{m} \cdot \frac{n(a)-1}{m-1}$$

wobei a jeder mögliche Buchstabe im Text ist, $n(a)$ die Anzahl wie oft a vorkommt und m die Länge des Texts.

3.1 Methoden zum Knacken der Vigenère Chiffre

Der Text sei $c_1c_2, ..., c_m$ und für ein gegebenes l definieren wir uns

$$s_i = c_i c_{i+l} c_{i+2l} ...$$

für $i \in [l]$. Für alle i berechnen wir uns den Koinzidenzindex und vergleichen den Wert mit dem der Sprache in der der Text geschrieben wurde (bei Englisch ungefähr 7%). Falls l die gesuchte Schlüssellänge ist, sollten die beiden Werte ungefähr übereinstimmen. Nun sind nur noch Cäsar Chiffren übrig. Um diese zu lösen, kann man alle 26 möglichen Schlüssel ausprobieren und anschließend jenen wählen, sodass die Wahrscheinlichkeitsverteilung des dechiffrierten Buchstaben am nächsten zu jener der Sprache ist.

3.2 Visuelle Kryptographie

Der Schlüssel ist ein Bild auf einer Folie. Es werden jeweils vier Pixel in 2×2 Blöcken gesammelt, wobei auf einer Diagonalen zwei schwarze Pixel und sonst zwei durchsichtige Pixel sind. Es gibt also zwei verschiedene solche Cluster. Wenn diese zufällig gesetzt werden, scheint das Bild ein druchgehender Grauton zu sein. Die Dechiffrierung kann dieselbe Folie verwenden wie die Chiffrierung. Man kann den Schlüssel nur einmal verwenden.

4 Shannons Modell eines klassischen Kryptosystems

Ein Kryptosystem ist ein Tripel (M, K, C) mit

- M ist die Nachricht
- K ist der Schlüssel
- C ist die Chiffre

Alle drei kann man als Zufallsvariablen modellieren, die nicht stochastisch unabhängig sind. Weiter nimmt man an

1. Die Verteilung von M entspricht der Buchstabenhäufigkeit
2. K ist gleichverteilt
3. M und K sind unabhängig

Es solches Kryptosystem mit De- und Entschlüsselungsfunktionen D, E ist sicher, wenn M und C unabhängig sind.

4.1 One-Time-Pad

Hier werden die Annahmen verwendet

- der Schlüssel wird nur einmal verwendet
- die Menge der möglichen Schlüssel ist so groß wie die Menge der möglichen Chiffren
- E ist bijektiv für alle k
- der Schlüssel zu m und c ist eindeutig, d.h. $k = k(m, c)$.

Lemma 4.1.1. *Jedes one-time-pad Kryptosystem ist absolut sicher.*

Rechnet man im Binärsystem kann als Schlüssel eine zufällige n-Bit Folge dienen, die bitweise auf die Nachricht addiert werde.

4.2 Pseudozufallsgeneratoren

Man möchte Schlüssel haben, die so lange sind, dass man nicht alle ausprobieren kann. Sie sollten aber ebenso nicht viel länger sein als die benötigte Mindestlänge. Deshalb werden Pseudozufallsgeneratoren. Hierbei wird ein *seed* verwendet, der zur Erzeugung herangezogen wird. Sender und Empfänger haben Zugriff auf den seed.

4.2.1 Linearer Kongruenzgenerator

Dieser liefert Zahlen aus $\mathbb{Z}_n$. Man beginnt hierbei mit dem seed $z_0 < n$ und berechnet anschließend die Folge $z_{i+1} = (a \cdot z_i + b) \mod n$. Aus einem seed von k Bits entstehen bis zu $n \approx 2^k$ Nachfolgezahlen. Die maximale Periodenlänge von n wird genau dann erreicht, wenn

- b und n teilerfremd sind
- $a \equiv 1 \mod p$ für alle Primteiler p von n
- Falls n durch 4 teilbar ist, muss $a \equiv 1 \mod 4$ gelten

4.2.2 Blum-Blum-Shub Generator

Dieser verwendet statt der obigen Iterationsvorschrift höhergradige Polynome der Form

$$z_{i+1} = (az_i^2 + bz_i + c) \mod n$$

wobei meist sogar $a = 1, b = 0$ verwendet wird. Das Modul n sollte $n = p \cdot q$ erfüllen, wobei p und q jeweils eine Blum-Primzahl ist, welche erfüllen müssen, dass $p + 1$ (bzw. $q + 1$) durch 4 teilbar ist. Für den seed muss außerdem gelten $ggT(z_0, n) = 1$. Wenn man nun eine Periode berechnet hat, ergibt sich der Schlüssel beispielsweise als Folge der letzten Bits der Zufallszahlen.

4.2.3 Rückgekoppelte Schieberegister

Die Idee ist, dass die Bits einer Zahl $b_1b_2...b_n$ so verschoben werden, dass sie auf der jeweils nächsten Stelle landen. b_n wird anschließend modular mit b_1 addiert und auf die erste Stelle geschoben. Das heißt, als Ergebnis erhält man

$$(b_1 \oplus b_n)b_1b_2...b_{n-1}$$

4.3 Chinesischer Restsatz

Gegeben sind $n_1, ...n_k \in \mathbb{Z}$ paarweise teilerfremd und $a_i \in \mathbb{Z}_{n_i}$ für $i \in \{1, 2, ..., k\}$. Gesucht ist ein x sodass

$$x = a_i \mod n_i$$

für alle i. Diese Lösung ist eindeutig modulo $\prod_{i=1}^{k} n_i$. Das Problem wird wie folgt gelöst

1. berechne $n = \prod_{i=1}^{k} n_i$
2. berechne $m_i = \frac{n}{n_i}$ für alle i
3. da $ggT(m_i, n_i) = 1$ existiert ein y_i das invers zu m_i modulo n_i ist
4. es gilt $x = \sum_{i=1}^{k} y_i \cdot a_i \cdot m_i$

Lemma 4.3.1. *Es seien $n_1, ..., n_k$ paarweise teilerfremd. Sei $n = \prod_{i=1}^{k} n_i$. Dann ist*

$$x \mod n \mapsto (x \mod n_1, x \mod n_2, ..., x \mod n_k)$$

eine bijektive Abbildung.

Definition 4.3.2 (Euler Funktion)**.** Wir definieren

$$\varphi(n) = |\{a | ggT(a,n) = 1,\ 1 \leq a \leq n\}|$$

In anderen Worten $\varphi(n) = |\mathbb{Z}_n^*|$

Lemma 4.3.3. *Sei $n = \prod_{i=1}^{k} p_i^{e_i}$ die Primfaktorisierung von n. Dann ist*

$$\varphi(n) = n \cdot \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right)$$

Satz 4.3.4 (Satz von Euler)**.** *Für jedes $a \in \mathbb{Z}_n^*$ gilt $a^{\varphi(n)} \equiv 1 \mod n$.*

Korollar 4.3.5. *Sei k die Ordnung von $a \in \mathbb{Z}_n^*$ und $e \in \mathbb{Z}$. Dann ist $a^e \equiv 1$ genau dann, wenn k e teilt.*

Korollar 4.3.6. *Für $a \in \mathbb{Z}_n^*$ ist $ord(a)$ ein Teiler von $\varphi(n)$.*

4.4 RSA

Wähle zwei Primzahlen p, q, berechne $n = p \cdot q$. Wähle e teilerfremd zu $(p-1)(q-1)$. (n, e) heißt *öffentlicher Schlüssel*. Berechne d mit $d \cdot e \equiv 1 \mod \varphi(n)$. d ist der private Schlüssel.
Verschlüsselung geschieht durch $c = m^e \mod n$ wobei m die Nachricht ist. Es gilt $m = c^d \mod n$, da $m \equiv (m^e)^d \mod n$.

4.5 zyklische Gruppen

Satz 4.5.1. *Die Gruppe $\mathbb{Z}_n^*$ ist zyklisch genau dann, wenn $n \in \{1, 2, 4, p^k, 2p^k\}$ bei p eine ungerade Primzahl ist.*

Lemma 4.5.2. *Ist k die Ordnung von $a \in G$ und $l \in \mathbb{N}$, so ist $\frac{k}{ggT(k,l)}$ die Ordnung von a^l.*

Satz 4.5.3. *Eine endliche zyklische Gruppe G hat genau $\varphi(|G|)$ Erzeuger. Daraus folgt, dass eine zyklische Gruppe $\mathbb{Z}_n^*$ genau $\varphi(\varphi(n))$ Erzeuger hat.*

Die Frage ist nun, wie Erzeuger gefunden werden können. Ein Erzeuger muss gewisse Eigenschaften erfüllen, die leicht überprüft werden können. Es wird dann einfach zufällig ein Element aus $\mathbb{Z}_n^*$ gewählt und die Eigenschaften überprüft. Zum Beispiel das folgende Kriterium.

Satz 4.5.4. *Sei $\mathbb{Z}_n^*$ zyklisch. $a \in \mathbb{Z}_n^*$ ist ein Erzeuger, wenn*

$$a^{\frac{\varphi(n)}{q}} \mod n \not\equiv 1$$

für alle q die $\varphi(n)$ teilen.

4.6 Exponentiation und diskreter Logarithmus

Sei $\mathbb{Z}_n^*$ eine zyklische Gruppe, wobei wir uns auf n prim beschränken wollen. Sei $x \in \mathbb{Z}_n^*$. Dann gibt es zu jedem Erzeuger ein bis auf modulo eindeutiges k, sodass $a^k \equiv x$. Dieses k wird der diskrete Logarithmus genannt. k zu bestimmen ist ein schwieriges Problem und daher Grundlage für viele Kryptosysteme.

Definition 4.6.1 (Diffie-Hellman-Schlüsselvereinbarung)**.** Es wird eine große Primzahl n und eine Primitivwurzel a festgelegt. Alice wählt $x < n$ und Bob $y < n$. Alice berechnet $\tilde{x} = (a^x \mod n)$ und Bob $\tilde{y} = (a^y \mod n)$. Die Ergebnisse werden ausgetauscht. Nun berechnen beide $z = (\tilde{x}^y \mod n) = (\tilde{y}^x \mod n) = (a^{xy} \mod n)$. Dieses z kann nun als gemeinsamer Schlüssel verwendet werden.

Definition 4.6.2 (No-Key-System von Shamir)**.** Es muss eine große Primzahl n gewählt werden. Alice will $m < n$ an Bob schicken. Dafür wählt sie ein zufälliges $a \in \mathbb{Z}_{n-1}^*$ und berechnet zusätzlich $a' = a^{-1}$. Sie berechnet $c = m^a \mod n$ und schickt c an Bob. Dieser wählt ebenso ein zufälliges $b \in \mathbb{Z}_{n-1}^*$ und das Inverse b' und berechnet $c' = c^b$ und schickt dies an Alice zurück. Alice bestimmt $c'' = c'^{a'}$ und schickt das an Bob. Dieser kann anschließend mit $c''^{b'} = m$ die initiale Nachricht bestimmen.

Definition 4.6.3 (ElGamal-Verfahren)**.** Es wird wieder eine große Primzahl n und eine Primitivwurzel a bestimmt. Bob wählt zufällig $y < n$ und berechnet $\tilde{y} = (a^y \mod n)$. Das ist der öffentliche und y der geheime Schlüssel. Alice möchte die Nachricht $m < n$ verschicken. Sie wählt zufällig $x < n$ und berechnet $\tilde{x} = (a^x \mod n)$. Dann berechnet sie $((\tilde{y})^x \mod n) = z$. Damit wird die Zahl verschlüsselt und $c = (mz \mod n)$ versandt. Alice sendet Bob die Informationen $(\tilde{x}, c)$. Mittels $(\tilde{x}^y \mod n) = z$ bestimmt Bob auch z und entschlüsselt damit m mit z^{-1}.

Definition 4.6.4. Ein quadratischer Rest modulo n ist eine Zahl $a \in \mathbb{Z}_n^*$, sodass ein $b \in \mathbb{Z}_n^*$ existiert mit $b^2 \equiv a \mod n$. b heißt dann Quadratwurzel von a modulo n.

Satz 4.6.5. *Sei n eine ungerade Primzahl. Wenn $a \in \mathbb{Z}_n^*$ ein quadratischer Rest modulo n ist, dann besitzt die Gleichung $x^2 \equiv a \mod n$ genau zwei Lösungen.*

Korollar 4.6.6. *Modulo einer Primzahl $n > 2$ gibt es genau $\frac{n-1}{2}$ quadratische Reste und Quadratwurzeln.*

Satz 4.6.7. *Sei $n > 2$ prim. Dann ist $a \in \mathbb{Z}_n^*$ ein quadratischer Rest modulo n genau dann, wenn $a^{\frac{n-1}{2}} \equiv 1 \mod n$.*

Bemerkung 4.6.8 (Berechnung der Quadratwurzeln). Ist $a \in \mathbb{Z}_n^*$ ein quadratischer Rest und ist $n \equiv 3 \mod 4$ (eine sogenannte *Blum*-Primzahl), dann ist $a^{\frac{n+1}{4}}$ eine Quadratwurzel von a. Ist b eine Quadratwurzel von a, so ist $n - b$ die zweite.
Die Fälle $n \equiv 2 \mod 4$ und $n \equiv 0 \mod 4$ sind nicht möglich, da n sonst gerade wäre. n ist aber eine ungerade Primzahl. Der Fall $n \equiv 1 \mod 4$ ist möglich, aber schwierig zu lösen. Dieser wird in der Praxis vermieden.
Für Primzahltests sind Quadratwurzeln nützlich, denn modulo einer Primzahl hat 1 genau die Quadratwurzeln ± 1. Mithilfe des CRT kann man Quadratwurzeln modulo $n = p \cdot q$ bestimmen.

Definition 4.6.9 (Rabin-Verfahren)**.**

1. wähle zufällig zwei große Primzahlen $p \neq q$ mit $p \equiv q \equiv 3 \mod 4$
2. setze $n = p \cdot q$. n ist der öffentliche und (p, q) der geheime Schlüssel
3. Alice berechnet $c ? m^2 \mod n$
4. Bob entschlüsselt durch Wurzelziehen. Er berechnet

$$m_p = c^{\frac{p+1}{4}} \mod p$$

und

$$m_q = c^{\frac{q+1}{4}} \mod q$$

5. er erhält die vier Lösungen $m_p, 1 - m_p, m_q, 1 - m_q$
6. daraus ergeben sich vier Gleichungen $x \equiv a \mod r$ mit $a \in \{m_p, 1-m_p, m_q, 1-m_q\}$ und $r \in \{p, q\}$.
7. Lösen mithilfe von CRT erhält er vier Lösungen für $x^2 \equiv c \mod n$ und eine davon ist m.

Um die richtige Nachricht aus den vier Möglichkeiten zu filtern, wird meistens vor der Verschlüsselung eine vorher vereinbarte Redundanz hinzugefügt. So werden zum Beispiel die ersten oder letzten der 64 Bits wiederholt. Mit hoher Wahrscheinlichkeit wird nur eine der vier Lösungen diese Redundanz widerspiegeln.

4.7 große Primzahlen finden

Satz 4.7.1. *Es gibt ungefähr $\frac{n}{\ln(n)}$ viele Zahlen die höchstens n sind prim.*

Satz 4.7.2 (Fermat-Test)**.** *Ist n prim, so gilt $\forall a \in \mathbb{Z}_n^*$, dass*

$$a^{n-1} \equiv 1 \mod n$$

Bemerke, dass in obigem Satz keine Äquivalenz gilt. Das motiviert die folgende

Definition 4.7.3 (Carmichael-Zahlen)**.** Ist n nicht prim aber

$$a^{n-1} \equiv 1 \mod n \; \forall a \in \mathbb{Z}_n^*$$

so nennt man n eine Carmichael-Zahl.

Das Ziel ist es nun, den Fermat-Test so zu verbessern, dass auch Carmichael-Zahlen erkannt werden. Der Trick ist hierbei die Beobachtung, dass wenn eine Primzahl $n > 2$ eine Quadratwurzel x von 1 mit $x \not\equiv \pm 1 \mod n$ hat, dann ist n nicht prim. Das daraus entstandene Verfahren heißt *Probabilistischer Primzahltest* nach Miller-Rabin. Wir schreiben die ungerade Zahl n als $1 + 2^r \cdot u$ mit u ungerade.

1. $a = random(1, n-1)$
2. if $ggt(a, n) \neq 1$ return *keine Primzahl*
3. $d = a^u \mod n = modexp(a, u, n)$
4. if $d = 1$ return *(vermutlich) Primzahl*
5. for $i = 0$ to $r - 1$ do
6. if $d = -1$ return *(vermutlich) Primzahl*
7. else $d = d^2 \mod n$
8. endfor
9. return *keine Primzahl*

Satz 4.7.4. *Die Ausgabe **keine Primzahl** des Miller-Rabin-Primzahltests stimmt immer.*
*Die Fehlerwahrscheinlichkeit der Ausgabe **(vermutlich) Primzahl** ist höchstens $\frac{1}{4}$. Die Laufzeit für eine Zahl n mit k Bits ist $\mathcal{O}(k^3)$.*

4.8 Zufallsprimzahlen in Binärdarstellung erzeugen

Ziel ist es, eine k-Bit Zufallsprimzahl zu erzeugen.

1. das erste und das letzte Bit werden auf 1 gesetzt, damit die Zahl ungerade wird
2. die restlichen $k-2$ Bits werden zufällig gewählt und ein Primzahltest durchgeführt.
3. ist die Zahl nicht prim, wiederhole Schritt 2

Man benötigt etwa $0.35n$ Versuche, bis das Verfahren erfolgreich war. Die Laufzeit liegt dabei bei $\mathcal{O}(k^4)$.

5 Zufallsalgorithmen

5.1 Monte-Carlo-Algorithmen

Hierbei handelt es sich um Entscheidungsalgorithmen, das heißt, es wird eines von zwei möglichen Ergebnissen geliefert. Mit einer gewissen Wahrscheinlichkeit p kann ein falsches Ergebnis ermittelt werden. Wird der Algorithmus bei gleicher Eingabe t mal wiederholt, so sinkt die Fehlerwahrscheinlichkeit auf p^t. Ist p jedoch exponentiell klein, so muss der Algorithmus zu oft wiederholt werden, um gut zu funktionieren. Ist die Laufzeit einschließlich der benötigten Anzahl an Wiederholungen polynomiell, so kann ein Monte-Carlo Algorithmus als etwa gleich gut wie ein polynomieller deterministischer Algorithmus erachtet werden.

5.2 Las Vegas Algorithmen

Diese dürfen keine falschen Ergebnisse berechnen, es darf lediglich ein Misserfolg ausgegeben werden. Die Wahrscheinlichkeit hierfür sei p. Dann kann der Algorithmus bei gleicher Eingabe und sich ändernden Zufallszahlen wiederholt werden bis eine Lösung gefunden wurde. Die mittlere Anzahl an Wiederholungen hierfür ist $\frac{1}{1-p}$.

6 Komplexitätsklassen

Ein Problem A ist auf ein anderes Problem B effizient (*Orakel-*)reduzierbar, geschrieben $A \preceq B$, wenn es einen effizienten Algorithmus gibt, der das Problem A löst. Dieser Algorithmus darf ein Unterprogramm mehrfach mit verschiedenen Eingaben aufrufen, das B löst. Die Komplexität von B bleibt unspezifiziert und fließt nicht in die Laufzeit des Algorithmus ein.

Satz 6.0.1. *Ist $B \in P$ und $A \preceq B$, so ist $A \in P$.*

Definition 6.0.2. Ein Problem L in NP ist NP-vollständig, wenn für jedes Problem A in NP die Relation $A \preceq L$ besteht.

Definition 6.0.3 (Einwegfunktion)**.** Eine Einwegfunktion ist in die Vorwärtsrichtung leicht berechenbar, in die Rückwärtsrichtung hingegen nicht, d.h. $f \in P$ aber $f^{-1} \notin P$. Es gilt außerdem $|x| \leq poly(|y|)$ und $|y| \leq poly(|x|)$.

Lemma 6.0.4. *Die Existenz von Einwegfunktionen würde $P \neq NP$ implizieren.*

Im Folgenden werden aber auch Funktionen, für die bisher keine effizient ausführbare Umkehrung bekannt ist, als Einwegfunktionen bezeichnet.

Definition 6.0.5. Eine *trapdoor one-way function* ist eine Einwegfunktion f, sodass es einen effizienten Algorithmus M gibt und für jede Länge $n(y)$ gibt es einen Schlüssel k_n, sodass für alle y gilt $M(y, k_n) = f^{-1}(y)$. Im Besitz des Schlüssels k_n ist die Umkehrung also leicht berechenbar.

6.1 RSA-Sicherheit

Sind n, e und c bekannt, so können verschiedene Probleme definiert werden.

1. Bestimme m
2. Bestimme d
3. Bestimme $\varphi(n)$
4. Bestimme einen Faktor p von n

Es ist klar, dass $1 \preceq 2 \preceq 3 \preceq 4$ gilt.

Lemma 6.1.1. *Es gilt $4 \preceq 3$.*

Das *schwache Brechen* von RSA basiert auf $(n, e, c) \mapsto m$ und ist auf das Faktorisierungsproblem reduzierbar. Es gilt also $1 \preceq 4$ aber die Umkehrung ist nicht bewiesen.
Für das *starke Brechen*, also $(n, e) \mapsto d$ gilt jedoch auch die Umkehrung also $4 \preceq 2$.

6.2 Rabin-Verfahren Sicherheit

Eine Quadratwurzel $\mod n$ mit $n = pq$ zu ziehen, ist Orakel-reduzierbarkeitsäquivalent zum Faktorisieren.

6.3 Algorithmen zur Faktorisierung

Das Faktorisierungsproblem ist auf das Problem, die Ordnung eines Elementes zu bestimmen, reduzierbar.

Satz 6.3.1 (Shor-Faktorisierung). *1. wähle (ggf. mehrfach) ein zufälliges x mit $1 < x < n$.*

2. berechne $g = ggT(x, n)$. Falls $g \neq 1$, gib g zurück

3. sonst ist $x \in \mathbb{Z}_n^*$

4. it Orakel bestimme die Ordnung k

5. wähle ein neues x falls k ungerade ist oder $x^{\frac{k}{2}} \equiv -1 \mod n$

6. gib $ggT(x^{\frac{k}{2}} - 1, n)$ zurück

Lemma 6.3.2. *Jede ungerade nicht prime Zahl $n > 9$ lässt sich schreiben als $n = x^2 - y^2$.*

Satz 6.3.3 (Fermat-Faktorisierung). *1. Sei $x_0 = \lceil \sqrt{n} \rceil$*

2. Man berechne sukzessive $d_k = x_k^2 - n$ wobei $x_k = x_0 + k$ bis man $d_k = y^2$.

3. dann ist $(x_k + y)(x_k - y) = n$.

Hierbei kann man aus Effizienzgründen verwenden, dass $d_{k+1} = d_k + 2x_k + 1$. Die Laufzeit ist im worst-case $\mathcal{O}(2^{\frac{k}{2}})$.

Das Verfahren lässt sich verallgemeinern:
Man sucht Zahlen x, y sodass $x^2 \equiv y^2 \mod n$, damit $(x-y)(x+y) = kn$ für ein k. Sofern $x \not\equiv \pm y \mod n$, so sind $ggT(x - y, n)$ und $ggT(x + y, n)$ nicht-triviale Teiler von n.

Satz 6.3.4 (Pollards $(p-1)$-Methode). *Diese funktioniert besonders gut, wenn $p-1$ nur kleine Primfaktoren hat, d.h. $p - 1$ ist Teiler einer Zahl $B!$ für ein kleines B.*

1. $a = 2, B = 1$

2. REPEAT

3. $B = B + 1,\ a = modexp(a, B, n),\ g = ggT(a - 1, n)$

4. UNTIL $g \neq 1$

5. if $g = n$ return fail

6. else return g

Die worst-case Laufzeit ist $\mathcal{O}(2^{\frac{k}{2}})$.

Satz 6.3.5 (Pollards ρ-Methode). *1. sei* $n = pq$ *mit* $p \leq q \leq \sqrt{n}$

2. man erzeugt Pseudozufallszahlen $z_{i+1} = f(z_i) = (z_i^2 + 1) \mod n$

3. nach ungefähr $\sqrt{p} \leq n^{\frac{1}{4}}$ *Schritten gibt es* $z_i \equiv z_j \mod p$

4. es ist sehr wahrscheinlich, dass $z_i \not\equiv z_j \mod q$ *also* $z_i \neq z_j$

5. dann ist $ggT(z_i - z_j, n)$ *ein Teiler von* n

Die worst-case Laufzeit ist $\mathcal{O}(\sqrt{p}) \approx \mathcal{O}(2^{\frac{k}{4}})$

6.4 Algorithmen zur Berechnung des Diskreten Logarithmus

Sei n eine Primzahl und $a \in \mathbb{Z}_n^*$ eine Primitivwurzel. Dann hat jedes $y \in \mathbb{Z}_n^*$ eine eindeutige Potenz x, d.h. es existiert ein $x \leq \varphi(n)$ sodass $y = a^x \mod n$. Ein naiver Algorithmus, der alle $x < n$ ausprobiert, hat Laufzeit $\mathcal{O}(2^k)$.

Satz 6.4.1 (Babystep-Giantstep). *Man schreibt* $x = x_1\lceil\sqrt{n}\rceil + x_2$ *und muss nur noch die* x_i *bestimmen. Das funktioniert mit der Überlegung*

$$y \cdot \left(a^{-1}\right)^{x_2} = \left(a^{\lceil\sqrt{n}\rceil}\right)^{x_1}$$

Mit der Definition $u = a^{-1}$ *und* $v = a^{\lceil\sqrt{n}\rceil}$ *muss das Problem* $yu^{x_2} = v^{x_1}$ *gelöst werden. Dafür wird zuerst die Menge*

$$L = \{(x_1, v^{x_1}) : x_1 \in \{0, 1, ..., \lfloor\sqrt{n}\rfloor\}\}$$

der Giantsteps bestimmt. Anschließend werden die Babysteps durchgegangen und schrittweise $y \cdot u^{x_2}$ *berechnet. Ist das Ergebnis an zweiter Stelle in einem Tupel in* L*, so wird* x_2 *und das dazugehörige* x_1 *ausgegeben.*

Satz 6.4.2 (Pollard-ρ-Algorithmus). *1. Mittels Zufallsgenerator, erzeuge* $z_1 = a^{i_1}y^{j_1}$, $z_2 = a^{i_2}y^{j_2}, ...$

2. Nach ungefähr $\sqrt{n}$ *Schritten gibt es* $\mu < \sigma$ *mit* $z_\mu = z_\sigma$

3. Somit gilt $a^{i_\mu}a^{xj_\mu} \equiv a^{i_\sigma}a^{xj_\sigma} \mod n$ *und damit*

$$i_\mu + xj_\mu \equiv i_\sigma + xj_\sigma \mod n$$

7 Digitale Signaturen

7.1 RSA-Signaturen

Unterschreiben eines Dokumentes m mittels RSA bedeutet, $u = m^d \mod n$ zu berechnen. Um sicherzustellen, dass ein Teilnehmer A tatsächlich m unterschrieben hat, wird $m = u^e \mod n$ berechnet. (n, e) ist dabei der öffentliche und d der geheime RSA Schlüssel.

7.2 El-Gamal Signaturen

n sei eine Primzahl und a eine Primitivwurzel. Der Teilnehmer möchte ein Dokument n unterschreiben. Er stellt den geheimen Schlüssel k und den öffentlichen Schlüssel $\tilde{k} = a^k \mod n$. Als nächstes erzeugt er eine Zufallszahl z die teilerfremd zu $n-1$ ist und berechnet $\tilde{z} = a^z \mod n$. Anschließend löst er die Gleichung

$$m \equiv k \cdot \tilde{z} + z \cdot x \mod n - 1$$

nach x auf. Die Unterschrift besteht aus dem Paar $(\tilde{z}, x)$. Verifiziert wird, indem man überprüft, dass

$$a^m \equiv \tilde{k}^{\tilde{z}} \cdot \tilde{z}^x \mod n$$