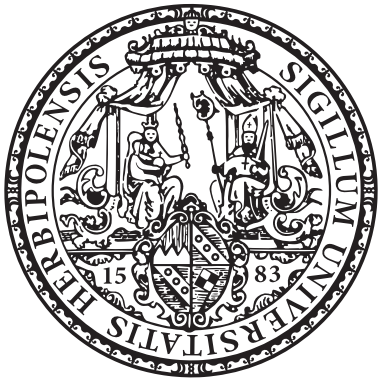
Kryptographie und Datensicherheit

Vorlesungsskript



Julius-Maximilians-Universität Würzburg
Fakultät für Mathematik und Informatik

gelesen von
Prof. Dr. Christian Glaßer
im Winter 2023/24

Dieses Skript basiert auf der privaten Mitschrift der Vorlesung „Kryptographie und Datensicherheit" von Marc Technau aus dem Wintersemester 2013/14.
Es wurde im Wintersemester 2021/22 von Fynn Godau überarbeitet.
Vielen Dank an die beiden für das Erstellen dieses Skripts.
Christian Glaßer

Inhaltsverzeichnis

Kapitel 1

Grundlegende Begriffe

Definition 1.1. (Zahlenbereiche und Wortmengen) Wir definieren:

- $\mathbb{N} := \{0, 1, 2, \dots\}$ (beachte $0 \in \mathbb{N}$) (Menge der natürlichen Zahlen)
- $\mathbb{N}^+ := \{1, 2, \dots\}$ (Menge der natürlichen Zahlen ohne Null)
- $\mathbb{Z} := \{\dots, -2, -1, 0, 1, 2, \dots\}$ (Menge der ganzen Zahlen)
- $\mathbb{Z}_n := \{0, 1, \dots, n-1\}$ (Reste modulo n)
- $\mathbb{Z}_n^i := \underbrace{\mathbb{Z}_n \times \mathbb{Z}_n \times \cdots \times \mathbb{Z}_n}_{i \text{ mal}}$
- $\mathbb{P} := \{p \in \mathbb{N} \mid p \text{ prim}\} = \{2, 3, 5, 7, 11, 13, \dots\}$ (Menge der Primzahlen)
- $\Sigma^{\leq n} := \{w \in \Sigma^* \mid |w| \leq n\}$ (Worte über Σ mit Länge $\leq n$)
- Für $x, y \in \mathbb{Z}$ mit $y \neq 0$ seit $(x \bmod y) := x - (y \cdot \left\lfloor \frac{x}{y} \right\rfloor)$. (Für positive x, y ist dies der Rest der Division $\frac{x}{y}$.)

Probabilistische Algorithmen

Einen probabilistischer Algorithmus kann man sich als nichtdeterministischen Algorithmus vorstellen, der bei jeder Verzweigung auswürfelt, auf welchem Weg die Berechnung fortgesetzt wird.

Definition 1.2. (probabilistischer Algorithmus)

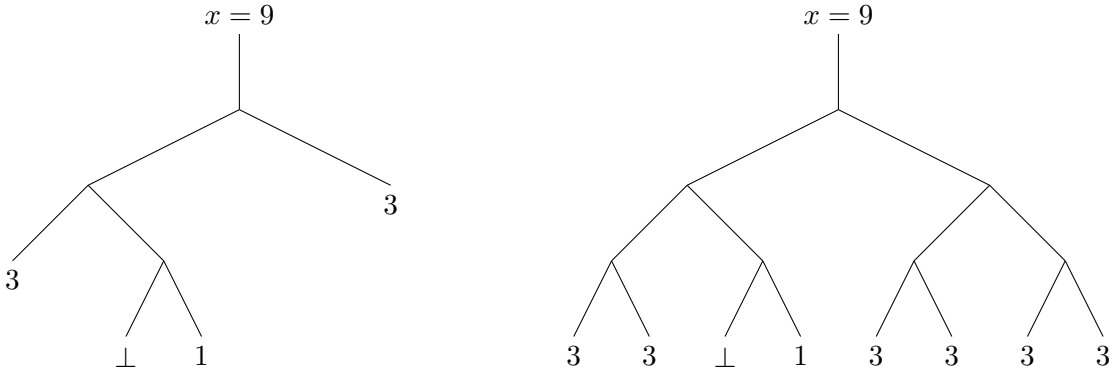
1. Ein *probabilistischer Algorithmus* ist ein nichtdeterministischer Algorithmus, mit folgenden Eigenschaften.
   (a) Jede nichtdeterministische Verzweigung erzeugt genau zwei Rechenwege.
   (b) Ein Rechenweg r, auf dem n nichtdeterministische Verzweigungen stattfinden, wird mit einer Wahrscheinlichkeit von $P(r) = \frac{1}{2^n}$ realisiert.
2. Ein probabilistischer Algorithmus M liefert mit folgender Wahrscheinlichkeit bei Eingabe x die Ausgabe y:
$$P(M(x) = y) := \sum_{\substack{r\text{: Rechenweg von } M \\ \text{auf } x \text{ mit Ausgabe } y}} P(r)$$
3. Ein Rechenweg heißt *erfolglos*, falls er die Ausgabe $\perp$ liefert.
4. Die *Erfolgswahrscheinlichkeit* eines probabilistischen Algorithmus M bei Eingabe x ist definiert als $1 - P(M(x) = \perp)$.
5. Eine probabilistischer Algorithmus M *arbeitet in der Zeit* $t : \mathbb{N} \to \mathbb{N}$, falls M auf x nur Rechenwege der Länge $\leq t(|x|)$ besitzt.

Amplifizierung der Erfolgswahrscheinlichkeit

Die n-fache Ausführung eines probabilistischen Polynomialzeitalgorithmus mit Erfolgswahrscheinlichkeit $\geq \frac{1}{2}$ ergibt einen probabilistischen Polynomialzeitalgorithmus mit Erfolgswahrscheinlichekeit $\geq 1 - \frac{1}{2^n}$.

Beispiel 1.3. (Erfolgswahrscheinlichkeit von Algorithmen) Wir betrachten die folgenden beiden äquivalenten Berechnungsbäume des probabilistischen Algorithmus M.

Folgende beiden Berechnungsbäume sind äquivalent:



$P(M(9) = 0) = 0$
$P(M(9) = 1) = \frac{1}{2^3} = \frac{1}{8}$
$P(M(9) = 2) = 0$
$P(M(9) = 3) = \frac{1}{2^2} + \frac{1}{2} = \frac{3}{4}$
$P(M(9) = \perp) = \frac{1}{8}$
$\Rightarrow$ Erfolgswahrscheinlichkeit von M bei Eingabe 9 ist $\frac{7}{8}$.

Kryptosysteme

Kryptographische Verfahren (Kryptosysteme) sollen sicherstellen, dass bestimmte Informationen nur von befugten Personen gelesen werden können.

Ein Kryptosystem soll folgendes leisten:

1. Umwandlung vom Klartext in den Geheimtext (Verschlüsselung)
2. Umwandlung vom Geheimtext in den ursprünglichen Klartext (Entschlüsselung)
3. Unbefugte sollen den Geheimtext nicht entschlüsseln können (Sicherheit)

In der Definition des Begriffs Kryptosystem fordern wir nur die Punkte 1 und 2. Im Kapitel 3 behandeln wir dann Sicherheit.

Definition 1.4. (Kryptosystem) Ein Kryptosystem ist ein Quadrupel $(\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$, sodass gilt:

1. Σ ist eine nichtleere, endliche Menge. (Bezeichnung: Alphabet)
2. $\mathcal{K}$ ist ein probabilistischer Polynomialzeitalgorithmus mit Erfolgswahrscheinlichkeit $\geq \frac{1}{2}$, $\mathcal{K}(1^n)$ generiert Schlüsselpaare (e, d) für Klartexte der Länge $\leq n$ (Bezeichnung: Schlüsselgenerator, key generator)
3. $\mathcal{E}$ ist ein probabilistischer Polynomialzeitalgorithmus mit Erfolgswahrscheinlichkeit $\geq \frac{1}{2}$, $\mathcal{E}(e, m)$ verschlüsselt den Klartext m mit dem Schlüssel e (Bezeichnung: Verschlüsselungsalgorithmus, encryption algorithm)
4. $\mathcal{D}$ ist ein deterministischer Polynomialzeitalgorithmus, $\mathcal{D}(d, c)$ entschlüsselt den Chiffretext c mit dem Schlüssel d (Bezeichnung: Entschlüsselungsalgorithmus, decryption algorithm)

5. Für alle $n \geq 1$, alle von $\mathcal{K}(1^n)$ generierten Schlüsselpaare (e, d), alle $m \in \Sigma^{\leq n}$ und alle von $\mathcal{E}(e, m)$ erzeugten Chiffretexte c gilt $\mathcal{D}(d, c) = m$.

Nach der Definition hängen Schlüssel nur von der Nachrichtenlänge, aber nicht vom Nachrichteninhalt ab.

Bei Betrachtung des Standardszenarios für Kryptosysteme erscheint diese Einschränkung sinnvoll: In diesem möchte Partei A an Partei B über einen unsicheren Kanal eine Nachricht senden. Zuvor wurde ein Passwort vereinbart bzw. veröffentlicht. Die Nachricht ist bei der Schlüsselerzeugung oft noch nicht bekannt.

Benennung der Parteien in kryptographischesn Szenarien

Wir legen folgende Standardbenennungen für die Parteien bei kryptographischen Szenarien fest.

- Alice und Bob: gewöhnliche Beteiligte; Alice sendet meist eine Nachricht an Bob
- Carol und Dave: stehen für eine dritte/vierte Partei
- Eve (eavesdropper, Lauscher): passiver Angreifer; will Nachrichten mithören, aber nicht verändern
- Mallory (malicious, hinterhältig): aktiver Angreifer; will Nachrichten fälschen oder austauschen
- Oscar (opponent, Gegner): wie Mallory
- Peggy (prover, Beweiser) und Victor (verifier, Prüfer): Verwendung bei Zero-Knowledge-Protokollen

Unser Standardszenario besteht darin, dass Alice eine Nachricht an Bob über einen unsicheren Kanal senden möchte.

Symmetrische und asymmetrische Kryptosysteme

1. Ein Kryptosystem heißt *symmetrisches Kryptosystem* oder *Private-Key-Kryptosystem*, falls für Schlüsselpaare (e, d) gilt: d lässt sich leicht aus e berechnen (meist gilt sogar $d = e$). Alice und Bob müssen zuvor den Schlüssel e ausgetauscht haben (etwa über eine sichere Leitung oder bei einem Treffen) und diesen geheimhalten.

2. Ein Kryptosystem heißt *asymmetrisches Kryptosystem* oder *Public-Key-Kryptosystem*, falls für Schlüsselpaare (e, d) gilt, dass sich d nicht effizient aus e berechnen lässt.
   Bob veröffentlicht hier e (öffentlicher Schlüssel) und hält d (privater Schlüssel) geheim. Jeder kann e benutzen, um verschlüsselte Nachrichten an Bob zu erstellen. Insbesondere ist dazu keine Kommunikation mit Bob notwendig.

Überblick zu Ansätzen der Kryptoanalyse

Wir betrachten zuletzt einige Angriffszenarien unter der Annahme, dass das Verschlüsselungsverfahren bekannt ist (sonst müssen alle gängigen Verfahren getestet werden):

- Ciphertext only: nur der Chiffretext bekannt (Entschlüsselung durch Brute Force und statistische Analyse)
- Known plaintext: Paare von Klar- und Chiffretexten bekannt (wenn viele Paare, dann verwendeter Schlüssel vielleicht ableitbar)

- Chosen Plaintext: Verwendung des Verschlüsselungsverfahrens (liefert viele Paare von Klar- und Chiffretexten)
- Chosen Ciphertext: Verwendung des Entschlüsselungsverfahrens (liefert viele Paare von Klar- und Chiffretexten)

Kapitel 2

Symmetrische Kryptosysteme

2.1 Cäsar-Chiffre (um 50 v. Chr.)

Julius Cäsar verwendete folgendes Verfahren, um Nachrichten an das römische Heer zu verschlüsseln:

Das Alphabet wird mit $\mathbb{Z}_{26} = \{0, \ldots, 25\}$ indiziert. Zum Verschlüsseln wird jeder Buchstabe durch den drittnächsten ersetzt.

Beispiel.

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
                        ⇓
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Beispielsweise verschlüsselt sich „KLEOPATRA" zu „NOHRSDWUD".

Erlaubt man beliebige Verschiebungen des Alphabets, so wird dieses Verfahren als Verschiebungschiffre bezeichnet.

Definition 2.1. (Cäsar-Chiffre, Verschiebungschiffre) Cäsar $:= (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit

- $\Sigma = \mathbb{Z}_{26}$
- $\mathcal{K}(1^n)$ liefert gleichverteilt zufällig ein Element aus $\{(e, e) \mid e \in \Sigma\}$
- $\mathcal{E}(e, m_1 \cdots m_n) = c_1 \cdots c_n$, wobei $c_i := m_i + e \bmod 26$
- $\mathcal{D}(e, c_1 \cdots c_n) = m_1 \cdots m_n$, wobei $m_i := c_i - e \bmod 26$

Bemerkung 2.2.

1. Die Verschiebungschiffre ist leicht zu brechen, da sie nur 26 Schlüssel besitzt. Schon zu Cäsars Zeiten bestand ihre Sicherheit nicht auf der Geheimhaltung des Schlüssels, sondern auf der Geheimhaltung des Verfahrens.
2. Cäsars Nachfolger Augustus verwendete ein ähnliches Verfahren: jeder Buchstabe wurde durch den Nachfolger ersetzt und der letzte Buchstabe durch AA.

2.2 Substitutionschiffre

Idee: Ähnlich zur Cäsar-Chiffre, nur dass jetzt mit einer beliebigen Permutation des Alphabets gearbeitet wird.

Beispiel.

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
                        ⇓
E K Y V X N B I H U M Z D G W J F L T R O Q C P A S
```

Definition 2.3. (Substitutionschiffre) Substitution $:= (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit:

- $\Sigma = \mathbb{Z}_{26}$
- $\mathcal{K}(1^n)$ liefert gleichverteilt zufällig ein Element aus $\{(\pi, \pi) \mid \pi : \mathbb{Z}_{26} \to \mathbb{Z}_{26} \text{ ist bijektiv}\}$
- $\mathcal{E}(\pi, m_1 \cdots m_n) = \pi(m_1) \cdots \pi(m_n)$
- $\mathcal{D}(\pi, c_1 \cdots c_n) = \pi^{-1}(c_1) \cdots \pi^{-1}(c_n)$

Bemerkung 2.4.

1. Das Verfahren ist etwas sicherer als die Cäsar-Chiffre, da $26! \approx 4 \cdot 10^{26}$ Permutationen existieren. Daher ist eine Brute-Force-Attacke nicht möglich.
2. Das Verfahren ist jedoch sehr anfällig gegen statistische Analysen.
   Beispielsweise sei hier die Häufigkeit deutscher Buchstaben dargestellt:

   $$\mathrm{E} = 15\%, \mathrm{N} = 9\%, \ldots, \mathrm{Q} = 0.014\%, \mathrm{X} = 0.013\%$$

   Hierdurch ergibt sich sehr schnell die Zuordnung der Buchstaben.

2.3 Vigenère-Chiffre (ca. 1580)

Die folgende Chiffre ist benannt nach Blaise de Vigenère, einem französischen Kryptographen und Diplomaten.

Idee: Ähnlich zur Cäsar-Chiffre, nur dass jetzt Blöcke von k Buchstaben mit Hilfe eines Passworts verschlüsselt werden.

Beispiel.

	`DASISTGEHEIM`:	$(03, 00, 18, 08, 18, 19, 06, 04, 07, 04, 08, 12)$
	$+$	$+$
	`PASSWORDPASS`:	$(15, 00, 18, 18, 22, 14, 17, 03, 15, 00, 18, 18)$
	$\Downarrow$	$\Downarrow$
	`SAKAOHXHWEAE`:	$(18, 00, 10, 00, 14, 07, 23, 07, 22, 04, 00, 04)$

Definition 2.5. (Vigenère-Chiffre) Wir definieren die Chiffre für eine feste Passwortlänge k.
$\text{Vigenère}_k := (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit

- $\Sigma = \mathbb{Z}_{26}$
- $\mathcal{K}(1^n)$ liefert gleichverteilt zufällig ein Element aus $\{(e, e) \mid e \in \Sigma^k\}$
- $\mathcal{E}(e_0 \cdots e_{k-1}, m_0 \cdots m_{n-1}) = c_0 \cdots c_{n-1}$, wobei $c_i := m_i + e_{i \bmod k} \bmod 26$
- $\mathcal{D}(e_0 \cdots e_{k-1}, c_0 \cdots c_{n-1}) = m_0 \cdots m_{n-1}$, wobei $m_i := c_i - e_{i \bmod k} \bmod 26$

Bemerkung 2.6.

1. Die Cäsar-Chiffre und die Verschiebungschiffre sind Spezialfälle der Vigenère-Chiffre.
2. Die Vigenère-Chiffre ist im Vergleich zu Substitutionschiffre etwas weniger anfällig gegen Häufigkeitsanalysen und bei guten Passwörtern wenig anfällig gegen Brute-Force-Attacken.
3. Zu einfache Passwörter ermöglichen eine Wörterbuchattacke.
4. Gleiche Klartexte haben gleiche Ciphertextteile, wenn der Abstand ein Vielfaches der Passwortlänge ist. Dies liefert Hinweise über die Passwortlänge k.
5. Bei bekannter Passwortlänge k betrachtet man dann jeden k-ten Buchstaben im Klartext. Die Wahrscheinlichkeiten dieser Buchstaben liefern Wahrscheinlichkeiten für den entsprechenden Buchstaben des Schlüssels.

6. Die Vigenère-Chiffre wurde 1854 durch Charles Babbage gebrochen.

7. Als *Autokey-Verschlüsselung* bezeichnet man die Variante der Vigenère-Chiffre, bei der das Passwort durch anhängen des Klartextes verlängert wird:

$$\begin{array}{lll} \texttt{DASISTGEHEIM:} & (03, 00, 18, 08, 18, 19, 06, 04, 07, 04, 08, 12) \\ \quad + & \quad + & \\ \texttt{KEYDASISTGEH:} & (10, 04, 24, 03, 00, 18, 08, 18, 19, 06, 04, 07) \\ \quad \Downarrow & \quad \Downarrow & \\ \texttt{NEQLSLOWAKMT:} & (13, 04, 16, 11, 18, 11, 14, 22, 00, 10, 12, 19) \end{array}$$

2.4 Vernam-Chiffre (1918)

Das folgende Verfahren ist benannt nach Gilbert Vernam, einem US-amerikanischen Ingenieur. Das Verfahren ist auch als One-Time-Pad (OTP) bekannt.

Idee: Schlüssel ist genauso lang wie der Klartext. Verschlüsselung durch buchstabenweise Kombination von Klartext und Schlüssel.

Beispiel.

$$\begin{array}{lll} \texttt{ROTESTELEFON:} & (17, 14, 19, 04, 18, 19, 04, 11, 04, 05, 14, 13) \\ \quad + & \quad + & \\ \texttt{KWTUNRGQFLXZ:} & (10, 22, 19, 20, 13, 17, 06, 16, 05, 11, 23, 25) \\ \quad \Downarrow & \quad \Downarrow & \\ \texttt{BKMYFKKBJQLM:} & (01, 10, 12, 24, 05, 10, 10, 01, 09, 16, 11, 12) \end{array}$$

Definition 2.7. (Vernam-Chiffre) Vernam $:= (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit

- $\Sigma = \mathbb{Z}_{26}$
- $\mathcal{K}(1^n)$ liefert gleichverteilt zufällig ein Element aus $\{(e, e) \mid e \in \Sigma^n\}$
- $\mathcal{E}(e_1 \cdots e_n, m_1 \cdots m_n) = c_1 \cdots c_n$, wobei $c_i := m_i + e_i \bmod 26$
- $\mathcal{D}(e_1 \cdots e_n, c_1 \cdots c_n) = m_1 \cdots m_n$, wobei $m_i := c_i - e_i \bmod 26$

Variante: $\Sigma = \{0, 1\}$ und Verschlüsselung durch bitweises XOR

Bemerkung 2.8.

1. In Kapitel 3 beweisen wir, dass die Vernam-Chiffre perfekte Sicherheit besitzt.

2. Wichtig für die Sicherheit der Vernam-Chiffre ist die gleichverteilt zufällige Wahl des Schlüssels.

3. KGB-Spione nutzten One-Time-Pads, um Nachrichten mit Papier und Bleistift zu verschlüsseln. Die Schlüssel wurden auf Pyropapier gedruckt, welches sehr schnell ohne Rückstände verbrennt.

4. Spezialeinheiten des MI6 verwendeten OTPs im Zweiten Weltkrieg zur Verschlüsselung der Kommunikation zwischen den Dienststellen.

5. SIGSALY: Voice-Scrambler der Alliierten im Zweiten Weltkrieg. Der Sender fügt dem Signal ein Rauschen hinzu, das beim Empfänger wieder entfernt wird. Das Rauschsignal wurde durch Paare identischer Schellackplatten verteilt.

6. Das Rote Telefon (Fernschreibverbindung zwischen Moskau und Washington DC) wird durch OTPs geschützt.

7. Zwischen 1944 und 1945 konnten US-Kryptoanalytiker einen OTP des deutschen Außenministeriums brechen, da der Schlüsselgenerator keine zufälligen Schlüssel produzierte.

8. 1945 konnten US-Kryptoanalytiker mit einem OTP verschlüsselte Nachrichten zwischen Canberra und Moskau entschlüsseln. Die Schlüssel waren zuvor für Nachrichten zwischen Moskau und Washington verwendet worden.

9. Auch das Ministerium für Staatssicherheit (MfS) der DDR verwendete OTPs.

2.5 Die Enigma

Die Enigma ist eine 1918 von Arthur Scherbius zunächst für den zivilen Gebrauch entwickelte Chiffriermaschine. Im Zweiten Weltkrieg wurde sie vom deutschen Militär verwendet und mehrfach weiterentwickelt. Die häufigste Variante ist die Enigma I.

Funktionsprinzip

Die Maschine besteht aus drei Walzen (Rotoren), die jeweils eine Permutation auf dem Alphabet bewirken (Bezeichnung α, β, γ). Die Walzen lassen sich gegeneinander verdrehen; pro verschlüsseltem Buchstabe werden eine oder mehrere Walzen gedreht (wie ein mechanischer Zähler). Schließlich gibt es eine weitere, feststehende Walze, den sogenannten Reflektor (Bezeichnung: δ Alphabetspermutation).

Die Verschlüsselung eines Buchstaben p zum Chiffretextbuchstaben c sieht dann so aus:

$$p \xrightarrow{\alpha} \xrightarrow{\beta} \xrightarrow{\gamma} \xrightarrow{\delta} \xrightarrow{\gamma^{-1}} \xrightarrow{\beta^{-1}} \xrightarrow{\alpha^{-1}} c$$

In der Maschine fließt die Information als elektrischer Strom.

Schwachpunkte

Der Reflektor sollte die Sicherheit steigern, da der Strom die Walzen zweimal durchläuft. Dies war ein Trugschluss.

Konsequenzen der Verdrahtung des Reflektors:

- kein Buchstabe kann mit sich selbst verdrahtet sein
  $\Rightarrow$ Enigma liefert fixpunktfreie Abbildung zwischen Klar- und Chiffretexten,
  d. h. chiffrierter Buchstabe C hat Klartext $\neq$ C

- wenn Reflektor A auf U abbildet, so auch U auf A
  $\Rightarrow$ Enigma liefert selbstinversive Abbildung,
  d. h. zweimalige Verschlüsselung liefert wieder den Klartext

Kennt oder ahnt man ein im Text vorkommendes Wort (z. B. OBERKOMMANDODERWEHRMACHT), so weiß man sofort, an welcher Stelle sich das Wort *nicht* befinden kann.

$\Rightarrow$ Schwächung der kombinatorischen Komplexität der Maschine

Entzifferung

Die Deutschen unterschätzten die Fähigkeiten der Codebreaker und übersahen die Möglichkeit maschineller Entzifferung.

1932 polnischer Mathematiker Marian Rejewski bricht die Enigma I

- nutzt legal gekaufte, zivile Variante der Enigma
- bestimmt mit Permutationstheorie die Verdrahtungsreihenfolge von Walzen und Reflektor der militärischen Variante
- bestimmt Walzenlage und Walzenstellung mit spezieller Maschine (Zyklometer)

1938 Einführung der Walzen 4 und 5 macht Enigma wieder sicher
(die Deutschen wissen aber nichts von Rejewskis Durchbruch)

1939 polnische Codebreaker offenbaren ihr Wissen französischen und britischen Kollegen
(Treffen bei Warschau)

1940 biritische Codebreaker starten mit Rejewskis Wissen einen erneuten Angriff auf die Enigma:

- Bletchly Park, militärische Dienststelle zur Entzifferung des deutschen Nachrichtenverkehrs
- Personaleinsatz 10000-14000 Frauen und Männer
- Alan Turing entwickelt Angriffsmethode, die eine wichtige Komponente der Enigma eliminiert: das Steckerbrett
- dies senkt die Größe des Schlüsselraums von 10^{20} auf 10^6
(3 aus 5 Walzen auswählen und einstellen)
- Turing entwirft elektromechanische Maschine „Turing-Bombe", die alle 10^6 Möglichkeiten in ca. 10h prüft
- 10min mit 60 Turing-Bomben (eine je Walzenkombination)

1940-1945 die Alliierten entziffern monatlich tausende wichtige geheime Nachrichten der Deutschen. Auf deutscher Seite bleibt dies unbemerkt.

Historiker vermuten, dass die Entschlüsselung der Enigma den Krieg um Monate, wenn nicht Jahre, verkürzt hat.

Nach dem Zweiten Weltkrieg wurden Enigmas von Großbritannien und den USA in großer Stückzahl an den Nahen Osten und nach Afrika verkauft, sodass man den Nachrichtenverkehr dieser Staaten mitlesen konnte.

2.6 Algebraische Grundlagen (Teil 1)

Definition 2.9. Eine Gruppe ist ein Paar $(G, *)$, für das gilt:

- G ist eine Menge und $* : G \times G \to G$ eine totale Abbildung
- $\forall a, b, c \in G[(a * b) * c = a * (b * c)]$ (Assoziativität)
- $\exists! e \in G \, \forall a \in G[a * e = e * a = a]$ (neutrales Element)
- $\forall a \in G \, \exists! b \in G[a * b = b * a = e]$ (inverse Elemente)

Die Gruppe heißt *kommutativ*, falls zusätzlich gilt:

- $\forall a, b \in G[a * b = b * a]$ (Kommutativität)

Definition 2.10. Ein Körper ist ein Tripel $(K, +, \cdot)$, für das gilt:

- K ist eine Menge
- $+$ und $\cdot$ sind totale Abbildungen $K \times K \to K$
- $(K, +)$ ist eine kommutative Gruppe (neutrales Element 0)
- $(K - \{0\}, \cdot)$ ist eine kommutative Gruppe (neutrales Element 1)
- $\forall a, b, c \in K[a \cdot (b + c) = (a \cdot b) + (a \cdot c)]$ (Distributivgesetz)

- $\forall a, b, c \in K[(a + b) \cdot c = (a \cdot c) + (b \cdot c)]$ (Distributivgesetz)

Der Körper heißt *endlich*, falls K endlich ist.

Damit funktioniert das Rechnen in Körpern in der gewohnten Weise (d. h. wie in den Körpern $\mathbb{Q}$, $\mathbb{R}$ und $\mathbb{C}$).

Irreduzible Polynome

Definition 2.11. Sei $\mathbb{K} = (K, +, \cdot)$ ein Körper. Der *Polynomring* $\mathbb{K}[x]$ ist die Menge aller Polynome mit der Variablen x und Koeffizienten aus K zusammen mit der Addition und Multiplikation von Polynomen.

Ein Polynom $r \in \mathbb{K}[x]$ heißt *irreduzibel*, wenn es nicht konstant ist und es keine nichtkonstanten Polynome p, q mit $r = p \cdot q$ gibt.

Irreduzible Polynome sind also die „Primzahlen der Polynomringe".

Endliche Körper

Definition 2.12. Sei p eine Primzahl. Der Körper $\mathbb{F}_p$ ist definiert durch $\mathbb{F}_p := (\mathbb{Z}_p, +, \cdot)$, wobei $+$ die Addition modulo p und $\cdot$ die Multiplikation modulo p ist.

Definition 2.13. Sei p eine Primzahl, $n \geq 2$ und $r \in \mathbb{F}_p[x]$ ein irreduzibles Polynom von Grad n. Der Körper $\mathbb{F}_{p^n}$ ist definiert durch $\mathbb{F}_{p^n} := (\mathbb{Z}_p^n, +, \cdot)$, wobei das Produkt $a \cdot b$ und die Summe $a + b$ für $a = (a_{n-1}, \ldots, a_0), b = (b_{n-1}, \ldots, b_0) \in \mathbb{Z}_p^n$ wie folgt definiert sind:

- betrachte a und b als Polynome aus $\mathbb{F}_p[x]$:
  $a(x) = a_{n-1}x^{n-1} + \cdots + a_1x^1 + a_0x^0$
  $b(x) = b_{n-1}x^{n-1} + \cdots + b_1x^1 + b_0x^0$
- $c(x) := (a(x) \cdot b(x)) \bmod r(x)$, $d(x) := (a(x) + b(x)) \bmod r(x)$
- $a \cdot b := (c_{n-1}, \ldots, c_0)$ mit $c(x) = c_{n-1}x^{n-1} + \cdots + c_1x^1 + c_0x^0$
  $a + b := (d_{n-1}, \ldots, d_0)$ mit $d(x) = d_{n-1}x^{n-1} + \cdots + d_1x^1 + d_0x^0$

Fakten zu endlichen Körpern:

1. Es gibt genau dann einen endlichen Körper mit q Elementen, wenn q die Potenz einer Primzahl ist (d. h. $q = p^n$, p prim). Er ist eindeutig[1] bestimmt und wird mit $\mathbb{F}_q$ bezeichnet.
2. Elemente aus $\mathbb{F}_{p^n}$ lassen sich mit $\lceil \log_2 p^n \rceil$ Bits beschreiben. Die Körperoperationen $+$ und $\cdot$ können in Polynomialzeit durchgeführt werden.

Beispiel 2.14. (Rechnen im Körper $\mathbb{F}_{2^8}$) Wir verwenden das irreduzible Polynom $r = x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x]$.

Es gilt $\mathbb{F}_{2^8} = (\mathbb{Z}_2^8, +, \cdot)$, wobei $+$ und $\cdot$ wie in 2.13 definiert sind.

Jedes Polynom mit Koeffizienten aus $\mathbb{F}_2$ und Grad ≤ 7 entspricht genau einem Element aus $\mathbb{Z}_2^8$ und umgekehrt.

Die Elemente von $\mathbb{F}_{2^8}$ können als Bytes betrachtet werden.

Beispiele für die Umrechnung

Dez	Hex	Bin	Element aus $\mathbb{Z}_2^8$	Polynom
3	$0x03$	$0b00000011$	$(0, 0, 0, 0, 0, 0, 1, 1)$	$x + 1$
91	$0x5B$	$0b01011011$	$(0, 1, 0, 1, 1, 0, 1, 1)$	$x^6 + x^4 + x^3 + x + 1$
161	$0xA1$	$0b10100001$	$(1, 0, 1, 0, 0, 0, 0, 1)$	$x^7 + x^5 + 1$

[1] eindeutig heißt in diesem Kontext eindeutig, bis auf Isomorphie. Dieses Konzept wird hier nicht näher ausgeführt.

Beispiel für die Addition

$$\begin{aligned} 45 + 102 &= 0b00101101 + 0b01100110 \\ &= (x^5 + x^3 + x^2 + 1) + (x^6 + x^5 + x^2 + x) \underbrace{\bmod r}_{\text{unnötig}} \\ &= x^6 + x^3 + x + 1 \\ &= 0b01001011 = 75 \end{aligned}$$

Bei der Addition fallen die Terme x^5 und x^2 weg, denn es wird im $\mathbb{F}_2$ addiert. Dort gilt $1 + 1 = 0$.

Beispiel für die Multiplikation

$$\begin{aligned} 133 \cdot 7 &= 0b10000101 \cdot 0b0000111 \\ &= ((x^7 + x^2 + 1) \cdot (x^2 + x + 1)) \mod r \\ &= (x^9 + x^4 + x^2 + x^8 + x^3 + x + x^7 + x^2 + 1) \mod r \\ &= (x^9 + x^8 + x^7 + x^4 + x^3 + x + 1) \mod r \\ &\overset{(1)}{=} ((x^9 + x^8 + x^7 + x^4 + x^3 + x + 1) - (x \cdot r)) \mod r \\ &= (x^9 + x^8 + x^7 + x^4 + x^3 + x + 1 - x^9 - x^5 - x^4 - x^2 - x) \mod r \\ &= (x^8 + x^7 - x^5 + x^3 - x^2 + 1) \mod r \\ &\overset{(2)}{=} (x^8 + x^7 + x^5 + x^3 + x^2 + 1) \mod r \\ &= (x^8 + x^7 + x^5 + x^3 + x^2 + 1 - r) \mod r \\ &= (x^7 + x^5 - x^4 + x^2 - x) \mod r \\ &\overset{(2)}{=} (x^7 + x^5 + x^4 + x^2 + x) \\ &= 0b10110110 = 182 \end{aligned}$$

(1) Wir suchen ein äquivalentes Polynom vom Grad ≤ 7. D. h. im Ergebnis dürfen x^9 und x^8 nicht vorkommen. Da wir mod r rechnen, erhalten wir äquivalente Polynome, wenn wir Vielfache von r addieren oder subtrahieren. Wir subtrahieren solange Vielfache von r (nämlich $x \cdot r$ und $1 \cdot r$), bis der Grad ≤ 7 ist.

(2) Da wir im $\mathbb{F}_2$ rechnen, beschreiben $+$ und $-$ die gleiche Operation.

2.7 AES

1976 wurde DES (Data Encryption Standard) entwickelt, in den 1990er Jahren 3DES (Triple DES, drei maliges Anwenden von DES). Im Jahr 2000 gewann der Algorithmus Rijndael (Entwickelt von Joan Daemen und Vincent Rijmen) einen Wettbewerb in Form einer öffentlichen Ausschreibung zur Findung eines neuen sicheren Kryptosystems und trägt seither den Namen AES (Advanced Encryption Standard).

AES ist ein Blockchiffre, d. h. man verschlüsselt den Klartext in Blöcken einer festen Größe (16 Bytes). Grundprinzip ist das Hintereinanderschalten verschiedener Permutationen. Bis heute wird AES als sicher angesehen, da bisher noch kein relevanter Angriff bekannt wurde. AES findet Anwendung in WPA2, SSH, Skype, EFS und vielen weiteren Bereichen.

Definition 2.15. (AES)
AES $:= (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit:

- $\Sigma = \mathbb{Z}_{2^{128}}$
- $\mathcal{K}(1^n)$ liefert gleichverteilt zufälliges Element aus $\{(e, e) \mid e \in \mathbb{Z}_{2^{128}}\}$
- $\mathcal{E}(e, m_1 \cdots m_n) = \mathrm{AES}_e(m_1) \cdots \mathrm{AES}_e(m_n)$, mit einer noch zu beschreibenden Bijektion AES_K, welche einen 16-Byte-Klartext mit dem 16-Byte-Schlüssel K zu einem 16-Byte-Chiffretext verschlüsselt.
- $\mathcal{D}(e, c_1 \cdots c_n) = \mathrm{AES}_e^{-1}(c_1) \cdots \mathrm{AES}_e^{-1}(c_n)$

Verschlüsselung mit AES

Bytes werden wie in Beispiel 2.14 als Elemente des Körpers $\mathbb{F}_{256}$ interpretiert.

- Klartext P: Block aus 16 Bytes, entspricht 4×4 Matrix über $\mathbb{F}_{256}$.
- Schlüssel K: Block aus 16 Bytes[2], entspricht 4×4 Matrix über $\mathbb{F}_{256}$.

Schlüssel K wird mit einer festgelegten Funktion

$$f : \mathbb{F}_{256}^{4\times4} \to \left(\mathbb{F}_{256}^{4\times4}\right)^{11}$$

expandiert. Es entstehen so die Schlüssel $K_0, \ldots, K_{10} \in \mathbb{F}_{256}^{4\times4}$.

Wir definieren Operationen, die im AES verwendet werden.

- Die Bijektion $\mathit{SubByte} : \mathbb{F}_{256}^{4\times4} \to \mathbb{F}_{256}^{4\times4}$ wendet eine Funktion S-box komponentenweise auf 4×4 Matrizen über $\mathbb{F}_{256}$ an.
  Dabei ist S-box : $\mathbb{F}_{256} \to \mathbb{F}_{256}$ eine Bijektion, sie ist durch folgende Tabelle (substitution box) festgelegt:

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e5	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

  Die Tabelle zeigt Elemente von $\mathbb{F}_{256}$ als Bytes in Hexadezimaldarstellung (siehe Beispiel 2.14). Es gilt S-box(`f6`) = `42`.

- Die Bijektion ShiftRow : $\mathbb{F}_{256}^{4\times4} \to \mathbb{F}_{256}^{4\times4}$ verschiebt die Matrixzeilen:

$$\mathrm{ShiftRow}\begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{pmatrix} := \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_2 & b_3 & b_4 & b_1 \\ c_3 & c_4 & c_1 & c_2 \\ d_4 & d_1 & d_2 & d_3 \end{pmatrix}$$

[2] Der AES-Standard erlaubt auch 24-Byte- und 32-Byte-Schlüssel.

- Die Bijektion MixCol : $\mathbb{F}_{256}^{4\times4} \to \mathbb{F}_{256}^{4\times4}$ multipliziert das Argument mit einer festgelegten 4×4 Matrix über $\mathbb{F}_{256}$:

$$\text{MixCol}(A) := \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \cdot A$$

  Die Matrix zeigt Elemente von $\mathbb{F}_{256}$ als Bytes in Hexadezimaldarstellung.

AES_K verschlüsselt den Klartext P wie folgt zum Chiffretext C:

$$\begin{aligned}
&\text{Runde 0} && P \xrightarrow{+K_0} \cdot \\
&\text{Runde 1} && \cdot \xrightarrow{\text{SubByte}} \cdot \xrightarrow{\text{ShiftRow}} \cdot \xrightarrow{\text{MixCol}} \cdot \xrightarrow{+K_1} \cdot \\
&\vdots \\
&\text{Runde 9} && \cdot \xrightarrow{\text{SubByte}} \cdot \xrightarrow{\text{ShiftRow}} \cdot \xrightarrow{\text{MixCol}} \cdot \xrightarrow{+K_9} \cdot \\
&\text{Runde 10} && \cdot \xrightarrow{\text{SubByte}} \cdot \xrightarrow{\text{ShiftRow}} \cdot \xrightarrow{+K_{10}} C
\end{aligned}$$

2.8 Betriebsmodi für Blockchiffren

AES ist eine sogenannte Blockchiffre.

Blockchiffre = deterministisches Kryptosystem, das Klartexte fester Länge auf Chiffretexte fester Länge abbildet

Um lange Klartexte mit Blockchiffren wie AES zu verschlüsseln, muss ein sogenannter Betriebsmodus gewählt werden.

Betriebsmodus = Festlegung, wie mit der Blockchiffre lange Klartexte verschlüsselt werden

Im Folgenden sehen wir uns zwei Beispiele für Betriebsmodi an.

2.8.1 Electronic Code Book Mode (ECB)

Der Modus ist nach herkömmlichen Codebüchern benannt.

Sie enthalten lange Listen von Wörtern / Sätzen und zugehörigen Codewörtern. Gleiche Wörter / Sätze führen zum gleichen Codewort.

Funktionsweise und Eigenschaften des ECB-Modus:

- die Blöcke werden unabhängig voneinander verschlüsselt
- gleiche Blöcke werden gleich verschlüsselt

$\Rightarrow$ Chiffretext kann viel Information über den Klartext liefern

Beispiel. (AES-Verschlüsselung im ECB-Modus)

Klartext

Chiffretext

⇒ ECB-Modus unsicher

2.8.2 Cipher Block Chaining Mode (CBC)

Funktionsweise des CBC-Modus:

- vor dem Verschlüsseln wird Klartext von Block i mit Chiffretext von Block $i-1$ durch bitweises XOR verknüpft
- für den ersten Block wird ein Initialisierungsvektor verwendet (z. B. Zeitstempel oder Zufallszahl), der unverschlüsselt übertragen wird

Eigenschaften des CBC-Modus

- Chiffretext von Block i hängt von Initialisierungsvektor und allen Blöcken $1, \ldots, i$ ab
- gleiche Blöcke werden unterschiedlich verschlüsselt

⇒ Chiffretext liefert wenig Informationen über den Klartext

Beispiel. (AES-Verschlüsselung im CBC-Modus)

Klartext

Chiffretext

⇒ CBC-Modus besitzt nicht die Schwächen des ECB-Modus

Kapitel 3

Perfekte Sicherheit

3.1 Grundlagen der Wahrscheinlichkeitstheorie

Definition 3.1. (Wahrscheinlichkeitsverteilung) Sei $\mathcal{M}$ eine höchstens abzählbar unendliche Menge („Ergebnismenge").

- Eine *Verteilung* auf $\mathcal{M}$ ist eine totale Funktion $P : \mathcal{M} \to [0, 1]$, so dass

$$\sum_{x \in \mathcal{M}} P(x) = 1.$$

- Ein *Ereignis* A ist eine Teilmenge von $\mathcal{M}$.
- Die *Wahrscheinlichkeit eines Ereignisses* A ist definiert durch

$$P(A) := \sum_{x \in A} P(x).$$

- Sind $A, B \subseteq \mathcal{M}$ Ereignisse mit $P(B) > 0$, so ist die *Wahrscheinlichkeit von* A *unter der Bedingung* B definiert durch

$$P(A \mid B) := \frac{P(A \cap B)}{P(B)}.$$

- A und B heißen *unabhängig*, falls $P(A \cap B) = P(A) \cdot P(B)$, oder äquivalent $P(A \mid B) = P(A)$.

Satz 3.2. (Bayes) Sind P eine Verteilung von $\mathcal{M}$ auf A, B Ereignisse mit $P(A), P(B) > 0$, so gilt

$$P(B) \cdot P(A \mid B) = P(A) \cdot P(B \mid A).$$

Beweis $P(B) \cdot P(A \mid B) = P(A \cap B) = P(A) \cdot P(B \mid A)$. □

3.2 Vernam-Chiffre und perfekte Sicherheit

Definition 3.3. Sei $S = (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ ein Kryptosystem, $n \geq 1$ und P_{Σ^n} eine Verteilung auf Σ^n.

$$
\begin{aligned}
K_n &:= \{e \mid (e,d) \text{ wird auf einem erfolgreichen Rechenweg von } \mathcal{K}(1^n) \text{ ausgegeben} \quad \} \\
&\quad \text{(möglicher Verschlüsselungsschlüssel für Nachrichten aus } \Sigma^n) \\
C_n &:= \{c \mid c \text{ wird auf einem erfolgreichen Rechenweg von } \mathcal{E}(e,m) \\
&\qquad \text{mit } e \in K_n \text{ und } m \in \Sigma^n \text{ ausgegeben} \quad \} \\
&\quad \text{(mögliche Chiffretexte von Nachrichten aus } \Sigma^n) \\
P_{K_n}(e) &:= \text{Wahrscheinlichkeit, dass ein erfolgreicher Rechenweg von } \mathcal{K}(1^n) \\
&\qquad \text{eine Ausgabe } (e,d) \text{ macht} \\
&\quad \text{(die von } \mathcal{K}(1^n) \text{ realisierte Verteilung auf } K_n)
\end{aligned}
$$

Ferner definieren wir die durch P_{Σ^n}, $\mathcal{K}$ und $\mathcal{E}$ realisierte Verteilung auf unserer Ergebnismenge $\Sigma^n \times K_n \times C_n$ als

$$
P(m,e,c) := P_{\Sigma^n}(m) \cdot P_{K_n}(e) \cdot \begin{pmatrix} \text{Wahrscheinlichkeit, dass ein} \\ \text{erfolgreicher Rechenweg von} \\ \mathcal{E}(e,m) \text{ eine Ausgabe } c \text{ macht} \end{pmatrix}.
$$

Außerdem definieren wir folgende Ereignisse.

$$
\begin{aligned}
E_m &:= \{m\} \times K_n \times C_n, && \text{für } m \in \Sigma^n && \text{(Nachricht } m \text{ liegt vor)} \\
E_e &:= \Sigma^n \times \{e\} \times C_n, && \text{für } e \in K_n && \text{(Schlüssel } e \text{ liegt vor)} \\
E_c &:= \Sigma^n \times K_n \times \{c\}, && \text{für } c \in C_n && \text{(Chiffretext } c \text{ liegt vor)}
\end{aligned}
$$

Mit der Definition von P schränken wir uns auf Verteilungen ein, bei denen die Ereignisse E_m und E_e unabhängig sind. Dies gilt jedoch in allen Anwendungsfällen mit folgenden Eigenschaften:

1. Schlüssel enthalten keine Information über die Nachrichten.
   (gilt nach Definition 1.4 für alle Kryptosysteme)

2. Nachrichten enthalten keine Information über die Schlüssel.

Idee für die Definition für perfekte Sicherheit

„Abhören lohnt sich nicht". Das soll heißen, dass das Abhören einer verschlüsselten Nachricht c dem Angreifer kein zusätzliches Wissen über die unverschlüsselte Nachricht m liefert.

Nach dem Abhören von c kennt der Angreifer nur die Anfangswahrscheinlichkeit von m.

Definition 3.4. (Perfekte Sicherheit) Seien $S = (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ ein Kryptosystem und $n \geq 1$. Ferner sei P_{Σ^n} eine Verteilung auf Σ^n und P die zugehörige Verteilung auf $\Sigma^n \times K_n \times C_n$.

1. S ist *perfekt sicher bezüglich* $P_{\Sigma^n} \overset{\text{df}}{\iff}$ für alle $m \in \Sigma^n$ und alle $c \in C_n$ mit $P(E_c) > 0$ gilt $P(E_m \mid E_c) = P(E_m)$.

2. S ist *perfekt sicher* $\overset{\text{df}}{\iff}$ für alle $n \geq 1$ und alle Verteilungen P_{Σ^n} auf Σ^n ist S perfekt sicher bezüglich P_{Σ^n}.

Mit der Notation aus 3.3 gilt der folgende Satz von Shannon.

Satz 3.5. (Shannon) Sei $S = (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ ein Kryptosystem mit deterministischem $\mathcal{E}$. Falls folgende Eigenschaften für alle $n \geq 1$ erfüllt sind, so ist S perfekt sicher.

1. $\forall e \in K_n\ P(E_e) = \frac{1}{|K_n|}$ (d. h. Schlüssel sind gleichverteilt)
2. $\forall m \in \Sigma^n\, \forall c \in C_n\, \exists! e \in K_n$ mit $\mathcal{E}(e,m) = c$ (d. h. jede Nachricht kann in jeden möglichen Chiffretext überführt werden, nämlich auf genau eine Weise)

Beweis Sei $n \geq 1$ und P_{Σ^n} eine Verteilung auf Σ^n.
Sei $m \in \Sigma^n$ und $c \in C_n$ mit $P(E_c) > 0$. Wir zeigen $P(E_m \mid E_c) = P(E_m)$.

1. Fall: $P(E_m) = 0$: dann gilt trivialerweise $P(E_m \mid E_c) = 0 = P(E_m)$.

2. Fall: $P(E_m) > 0$:

- Für $q \in \Sigma^n$ und $c \in C_n$ sei $e_{q,c}$ der wegen Annahme 2 eindeutig bestimmte Schlüssel $e \in K_n$ mit $\mathcal{E}(e,q) = c$
- nach dem Satz 3.2 von Bayes gilt:

$$\begin{aligned} P(E_m \mid E_c) &= \frac{P(E_m) \cdot P(E_c \mid E_m)}{P(E_c)} \\ &= \frac{P(E_m) \cdot P(E_{e_{m,c}} \mid E_m)}{P(E_c)} \\ &\overset{*}{=} \frac{P(E_m) \cdot \overbrace{P(E_{e_{m,c}})}^{=\frac{1}{|K_n|} \text{ wegen 1.}}}{\sum_{q\in\Sigma^n} P(E_q) \cdot \underbrace{P(E_{e_{q,c}})}_{=\frac{1}{|K_n|} \text{ wegen 1.}}} \\ &= \frac{P(E_m)}{\sum_{q\in\Sigma^n} P(E_q)} = \frac{P(E_m)}{1} = P(E_m) \end{aligned}$$

$*$ Schlüssel unabhängig von Klartext

$\Rightarrow S$ ist perfekt sicher □

Satz 3.6. Vernam ist perfekt sicher.

Beweis Vernam $= (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit

- $\Sigma = \mathbb{Z}_{26}$
- $\mathcal{K}(1^n)$ liefert ein gleichverteilt zufälliges Element aus $\{(e,e) \mid e \in \Sigma^n\}$
- $\mathcal{E}(e_1 \cdots e_n, m_1 \cdots m_n) := c_1 \cdots c_n$ mit $c_i = m_i + e_i \bmod 26$
- $\mathcal{D}(e_1 \cdots e_n, c_1 \cdots c_n) := m_1 \cdots m_n$ mit $m_i = c_i - e_i \bmod 26$

$\Rightarrow \mathcal{E}$ ist deterministisch.
Sei $n \geq 1$. Dann ist $K_n = \Sigma^n = C_n$.

1. $$\forall e \in K_n : P(E_e) = \begin{array}{c}\text{Wahrscheinlichkeit, dass ein erfolgreicher} \\ \text{Rechenweg von } \mathcal{K}(1^n) \text{ eine Ausgabe } (e,d) \text{ macht}\end{array} = \frac{1}{|K_n|}$$

   (Bedingung 3.5.1 ist also erfüllt)

2. sei $m = m_1 \cdots m_n \in \Sigma^n$ und $c = c_1 \cdots c_n \in C_n$
   dann gilt $\mathcal{E}(e,m) = c$ für $e = e_1 \cdots e_n$ mit $e_i = (c_i - m_i \bmod 26)$
   für jedes $e' = e'_1 \cdots e'_n \in K_n$ mit $\mathcal{E}(e',m) = c$ gilt:
   $\Rightarrow \quad (m_i + e_i) \bmod 26 = (m_i + e'_i) \bmod 26$ für $i = 1, \ldots, n$

$\Rightarrow$ $e_i = e'_i$ für $i = 1, \ldots, n$

$\Rightarrow$ $e = e'$, also existiert genau ein solcher Schlüssel, d. h. 3.5.2 gilt

$\overset{3.5}{\Longrightarrow}$ Vernam perfekt sicher □

Die Bedingung 3.5.2 impliziert

$$|K_n| = |C_n| \geq |\Sigma^n|.$$

Es stellt sich die Frage, ob ein so großer Schlüsselraum notwendig für perfekte Sicherheit ist. Wir zeigen nun, dass Kryptosysteme mit einem kleineren Schlüsselraum nicht perfekt sicher sind.

Satz 3.7. Sei $S = (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ ein Kryptosystem mit deterministischem $\mathcal{E}$ und sei $n \geq 1$. Falls $|K_n| < |\Sigma^n|$, so ist S nicht perfekt sicher.

Beweis Sei $P_{\Sigma^n}(m) := \frac{1}{|\Sigma^n|}$
Wir zeigen, dass S nicht perfekt sicher bezüglich P_{Σ^n} ist.
Sei $m' \in \Sigma^n$ und $e' \in K_n$; betrachten folgende 2 Mengen von Chiffretexten:

$$|\{\mathcal{E}(e, m') \mid e \in K_n\}| \leq |K_n|$$

$$|\{\mathcal{E}(e', m) \mid m \in \Sigma^n\}| = |\Sigma^n| > |K_n|$$

Falls die genannte Gleichheit nicht gilt, existieren verschiedene $m_1, m_2 \in \Sigma^n$ mit $\mathcal{E}(e', m_1) = \mathcal{E}(e', m_2)$. Wegen $e' \in K_n$ gibt es ein d', sodass (e', d') von $K(1^n)$ erzeugt wird. Nach 1.4.5 gilt $\mathcal{D}(d', \mathcal{E}(e', m_1)) = m_1$ und $\mathcal{D}(d', \mathcal{E}(e', m_2)) = m_2$. Aus $\mathcal{E}(e', m_1) = \mathcal{E}(e', m_2)$ folgt $m_1 = m_2$, ein Widerspruch.

$\Rightarrow$ es ex. $m \in \Sigma^n$ mit $\underbrace{\mathcal{E}(e', m)}_{:=c} \notin \{\mathcal{E}(e, m') \mid e \in K_n\}$

$\Rightarrow$ $\left.\begin{array}{c} P(E_{m'} \mid E_c) = 0 \\ P(E_c) > 0 \text{ nach Voraussetzung} \end{array}\right\}$ ↯ zu 3.4.1

$\Rightarrow$ S nicht perfekt sicher □

Trotzdem gibt es Kryptosysteme mit $|K_n| < |\Sigma^n|$, die in der Praxis als „sicher" gelten.

Dort enthält der Chiffretext c zwar Informationen über den Klartext m (es gilt dort $P(E_m \mid E_c) \neq P(E_m)$), aber diese Informationen lassen sich nicht in polynomieller Zeit extrahieren.

Im folgenden Kapitel lernen wir solche Kryptosysteme kennen.

Sicherheit bedeutet hier, dass das System nicht in probabilistischer Polynomialzeit gebrochen werden kann.

Kapitel 4

Asymmetrische Kryptosysteme

Hauptproblem der bisherigen Verfahren ist der Schlüsselaustausch, der im Geheimen stattfinden musste. Bei Public-Key-Kryptosystemen wird der zum Verschlüsseln benötigte Schlüssel stattdessen öffentlich gemacht.

Schlüsselaustauschproblem

Situation:

- Alice und Bob wollen Informationen über unsicheren Kanal austauschen
- dazu soll symmetrisches Verfahren (AES) verwendet werden

Frage: Wie können sich Alice und Bob auf einen geheimen Schlüssel einigen, wenn sie nur über den unsicheren Kanal kommunizieren können?

Seit den Anfängen der Kryptographie wurde das Schlüsselaustauschproblem als nicht lösbar betrachtet.

1976 sorgten Whitfield Diffie und Martin Hellman für eine Überraschung, als sie das erste Schlüsselaustausch-Protokoll veröffentlichten.

1977 veröffentlichten Ronald Rivest, Adi Shamir und Leonard Adleman mit RSA das erste Public-Key-Kryptosystem.

1985 entwickelte Taher Elgamal ein auf der Idee des Diffie-Hellman-Schlüsselaustauschs basierendes Public-Key-Kryptosystem.

Wir behandeln zunächst das RSA-Kryptosystem und danach den Diffie-Hellman-Schlüsselaustausch und das Elgamal-Kryptosystem.

Allgemeine Arbeitsweise von Public-Key-Kryptosystemen

Jeder Teilnehmer besitzt ein Schlüsselpaar: (öffentlicher Schlüssel e, privater Schlüssel d).

Der öffentliche Schlüssel wird in ein öffentliches Verzeichnis geschrieben, der private Schlüssel bleibt geheim. Will Alice an Bob eine verschlüsselte Nachricht senden, so verwendet sie Bobs öffentlichen Schlüssel, um die Nachricht zu verschlüsseln. (kein gemeinsamer Schlüssel, keine Kommunikation mit Bob)

4.1 Algebraische Grundlagen (Teil 2)

Satz 4.1. Sei $\mathbb{K}$ ein Körper und $f \in \mathbb{K}[x]$ ein Polynom vom Grad $d > 0$. Dann hat f höchstens d Nullstellen in $\mathbb{K}$.
(D. h. es existieren höchstens d Elemente $a \in \mathbb{K}$ mit $f(a) = 0$.)
Ohne Beweis.

Definition 4.2. Für $n \geq 1$ sei folgendes definiert:

- $\mathbb{Z}_n^* := \{i \in \mathbb{N} \mid 1 \leq i \leq n \text{ und } \mathrm{ggT}(i, n) = 1\}$
- $\varphi(n) := |\mathbb{Z}_n^*|$ (Eulersche φ-Funktion)

Satz 4.3. Sei $n \geq 2$. Wählt man für $\cdot$ die Multiplikation modulo n, so ist $(\mathbb{Z}_n^*, \cdot)$ eine Gruppe. Sie wird *multiplikative Gruppe der Ordnung* $\varphi(n)$ oder auch *Einheitengruppe von* $\mathbb{Z}_n$ genannt.
Ohne Beweis.

Satz 4.4. Ist p prim, so ist $\mathbb{F}_p := (\mathbb{Z}_p, +, \cdot)$ ein Körper, wobei $+$ die Addition und $\cdot$ die Multiplikation modulo p bezeichnen.
Ohne Beweis.

Satz 4.5. Seien $m, n \geq 2$.

1. Falls $\mathrm{ggT}(m, n) = 1$, so gilt $\varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$.
2. Falls p prim ist, so gilt $\varphi(p) = p - 1$ und $\varphi(p^k) = p^k - p^{k-1}$ für $k \geq 1$.
3. Für $k \geq 5$ gilt $\varphi(k) \geq k/(6 \ln \ln k)$.

Ohne Beweis.

Kennt man die Faktorisierung von n, so lässt sich $\varphi(n)$ entsprechend Satz 4.5 in polynomieller Zeit berechnen.

Satz 4.6. (Euler) Für $n \geq 2$ und $a \in \mathbb{Z}_n^*$ gilt $a^{\varphi(n)} \equiv 1 \bmod n$.
Ohne Beweis.

Spezialfall:

Satz 4.7. (kleiner Fermatscher Satz) Falls p prim und $a \in \mathbb{Z}_p^*$, so gilt $a^{p-1} \equiv 1 \bmod p$.
Ohne Beweis.

Definition 4.8. Seien $n \geq 2$ und $x \in \mathbb{Z}_n^*$. Die *Ordnung von* x ist definiert als $\mathrm{ord}_n(x) :=$ kleinstes $r \in \mathbb{N}^+$ mit $x^r \equiv 1 \bmod n$.

Satz 4.9. (Lagrange) Für $n \geq 2$ und $x \in \mathbb{Z}_n^*$ gilt $\mathrm{ord}_n(x) \mid \varphi(n)$. (Allgemein teilt die Ordnung jeder Untergruppe die Gruppenordnung.)
Ohne Beweis.

Satz 4.10. (chinesischer Restsatz) Seien $m_1, \ldots, m_k \in \mathbb{N} \setminus \{0, 1\}$ paarweise teilerfremd, $m := \prod_{1 \leq i \leq k} m_i$ und $a_1, \ldots, a_k \in \mathbb{N}$. Dann existiert genau ein $x \in \{0, \ldots, m-1\}$ mit $\forall_{1 \leq i \leq k} x \equiv a_i \bmod m_i$.
(Genauer gilt $x = \sum_{i=1}^{k} a_i \cdot q_i \cdot q_i^{-1} \bmod m$, wobei $q_i := {}^{m}/_{m_i}$ und $q_i^{-1} \equiv$ Inverses von $q_i \in \mathbb{Z}_{m_i}^*$.)
Ohne Beweis.

4.2 RSA-Kryptosystem

Das RSA-Kryptosystem ist nach seinen Entwicklern Ron Rivest, Adi Shamir und Leonard Adleman benannt. Es ist das erste veröffentlichte Public-Key-Kryptosystem.

4.2.1 Das RSA-Protokoll

Sei $\Sigma = \{0,1\}$ und wähle $r \geq$ Nachrichtenlänge.

Wir setzen $l := \lceil (r+1)/2 \rceil$, es bestimmt die Größe der verwendeten Primzahlen.

Alice möchte Bob eine Nachricht m mit $0 \leq m \leq 2^{r+1} - 2$ senden. Dabei werden die folgenden Schritte durchgeführt.

1. Bob macht folgendes:
   - wählt zufällig verschiedene Primzahlen $p, q \in \{2^l, \ldots, 2^{l+1} - 1\}$
   - setzt $n := pq$, damit gilt
     $$\varphi(n) = (p-1)(q-1)$$
   - wählt zufälliges $e \in \{2, \ldots, \varphi(n) - 1\}$ mit $\mathrm{ggT}(e, \varphi(n)) = 1$ (d. h. e ist teilerfremd zu $\varphi(n)$)
   - berechnet $d := e^{-1} \bmod \varphi(n)$ (d. h. das multiplikative Inverse zu e modulo $\varphi(n)$)
   - Bobs public key ist (n, e), sein private key ist (n, d)
   - veröffentlicht public key (n, e)
2. Alice macht folgendes:
   - verschlüsselt ihre Nachricht $m \in \{0, \ldots, 2^{r+1} - 2\}$ als $c := m^e \bmod n$
   - sendet c an Bob
3. Bob entschlüsselt die Nachricht durch $c^d \bmod n$

Wir zeigen, dass Bob die kodierte Nachricht entschlüsseln kann.

Satz 4.11. (Korrektheit von RSA) Seien p, q verschiedene Primzahlen und $n := pq$. Sei $1 < e < \varphi(n)$ mit $\mathrm{ggT}(e, \varphi(n)) = 1$ und $1 < d < \varphi(n)$ mit $e \cdot d \equiv 1 \bmod \varphi(n)$. Dann gilt für alle $m \in \{0, \ldots, n-1\}$
$$m = (m^e)^d \bmod n.$$

Beweis Wegen $e \cdot d \equiv 1 \pmod{\varphi(n)}$ existiert ein k mit $e \cdot d = 1 + k\varphi(n) = 1 + k(p-1)(q-1)$.

$$\begin{aligned} \Rightarrow (m^e)^d &= m^{e \cdot d} \\ &= m^{1+k(p-1)(q-1)} \\ &= m\Big(m^{p-1}\Big)^{k(q-1)} \qquad (*) \end{aligned}$$

Wir zeigen nun
$$(m^e)^d \equiv m \bmod p. \qquad (**)$$

1. Fall: $p \mid m$, dann sind beide Seiten $\equiv 0 \pmod p$

2. Fall: $p \nmid m$, dann ist $m \equiv a \pmod p$ für ein $a \in \{1, \ldots, p-1\} = \mathbb{Z}_p^*$

$$\begin{aligned} &\overset{\text{kleiner Fermat}}{\Longrightarrow} m^{p-1} \equiv 1 \pmod p \\ &\overset{(*)}{\Longrightarrow} (m^e)^d \equiv m \cdot 1^{k(q-1)} \equiv m \pmod p \end{aligned}$$

Dies zeigt $(**)$. Analog zeigt man
$$(m^e)^d \equiv m \bmod q \qquad (***)$$

Aus $(**)$ und $(***)$ folgt:

$$(m^e)^d - m \text{ ist durch } p \text{ und } q \text{ teilbar, also auch durch } p \cdot q, \text{ weil } p \neq q.$$

$$\Rightarrow (m^e)^d \equiv m \bmod n.$$

Der Satz folgt, da $m < n$.

□

Beispiel 4.12. (RSA-Protokoll)

- $\Sigma = \{0, 1\}$
- $r = 6$ (max. Nachrichtenlänge)
- $e := \lceil \frac{r+1}{2} \rceil = 4$

1. Bob erzeugt seine Schlüssel:
   - wählt zufällig $p = 17, q = 23 \in \{2^4, \ldots, 2^5 - 1\}$
   - $n := p \cdot q = 391$, damit gilt $\varphi(n) = 16 \cdot 22 = 352$
   - wählt zufällig $e = 109 \in \{2, \ldots, \varphi(n) - 1\}$
     mit $\text{ggT}(e, \varphi(n)) = 1$
   - $d := e^{-1} \bmod \varphi(n) = 109^{-1} \bmod 352 = \underset{\left(\begin{array}{c}\text{erweiterter Euklidischer}\\ \text{Algorithmus}\end{array}\right)}{\ldots} = 197$

     (Probe: $109 \cdot 197 \equiv 1 \pmod{352}$)
   - public key: $(n, e) = (391, 109)$
     private key: $(n, d) = (391, 197)$
2. Alice verschlüsselt die Nachricht $m = 42$:
   - berechnet c:

$$\begin{aligned} c := m^e \bmod n &= 42^{109} \mod 391 \\ &= 42^9 \cdot \left(42^{10}\right)^{10} \mod 391 \\ &= 263 \cdot 98^{10} \mod 391 \\ &= 247 \end{aligned}$$

   - sendet $c = 247$ an Bob
3. Bob entschlüsselt $c = 247$:

$$\begin{aligned} m' := c^d \bmod n &= 247^{197} \mod 391 \\ &= \left(\left(247^{13}\right)^5\right)^3 \cdot 247^2 \mod 391 \\ &= \left(263^5\right)^3 \cdot 13 \mod 391 \\ &= 111^3 \cdot 13 \mod 391 \\ &= 42 \end{aligned}$$

Definition 4.13. (RSA-Kryptosystem) Wir definieren $\text{RSA} := (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit

- $\Sigma = \{0, 1\}$

- $\mathcal{K}(1^r)$ ist folgende probabilistische Berechnung:
  1. $l := \lceil (r+1)/2 \rceil = \lfloor (r+2)/2 \rfloor$
  2. gleichverteilt zufällige Wahl verschiedener Primzahlen $p, q \in \{2^l, \ldots, 2^{l+1} - 1\}$
  3. gleichverteilt zufällige Wahl eines Exponenten $e \in \{2, \ldots, (p-1)(q-1) - 1\}$ mit $\mathrm{ggT}(e, (p-1)(q-1)) = 1$
  4. $d := e^{-1} \bmod (p-1)(q-1)$.
  5. return $((pq, e), (pq, d))$
- $\mathcal{E}((n, e), m) := m^e \bmod n$ für $m \in \Sigma^{\leq r} = \{0, \ldots, 2^{r+1} - 2\}$
- $\mathcal{D}((n, d), c) := c^d \bmod n$ für $c \in \{0, \ldots, n-1\}$

4.2.2 Effizienz des RSA-Protokolls

Wahl der Primzahlen p und q

Wir betrachten zunächst das Problem bei der Schlüsselgenerierung gleichverteilt Primzahlen $p, q \in \{2^l, \ldots, 2^{l+1} - 1\}$ zu wählen:

1. wähle gleichverteilt zufällige $p, q \in \{2^l, \ldots, 2^{l+1} - 1\}$;
   falls $p \notin \mathbb{P}$ oder $q \notin \mathbb{P}$, so stoppe erfolglos
2. falls $p = q$, so stoppe erfolglos
3. return (p, q)

Im obigen Algorithmus ist der Primzahltest in deterministischer Polynomialzeit möglich (Agrawal, Kayal, Saxena 2002). Die Anzahl der Primzahlen in $\{2, \ldots, N\}$ liegt für große N zwischen

$$0.9 \frac{N}{\ln N} \qquad \text{und} \qquad 1.2 \frac{N}{\ln N}.$$

Damit ist für genügend große l die Anzahl der Primzahlen in $\{2^l, 2^{l+1} - 1\}$ mindestens:

$$\begin{aligned} 0.9 \frac{2^{l+1}}{\ln 2^{l+1}} - 1.2 \frac{2^l}{\ln 2^l} &= \frac{1.8 \cdot 2^l}{(l+1)\ln 2} - \frac{1.2 \cdot 2^l}{l \ln 2} \\ &\geq \frac{2^l}{\ln 2}\left(\frac{1.8}{1.1 \cdot l} - \frac{1.2}{l}\right) \\ &= \frac{2^l}{l} \cdot \frac{1.8 - 1.1 \cdot 1.2}{1.1 \cdot \ln 2} \\ &\geq \frac{2^l}{2l} \end{aligned}$$

Damit gilt: Erfolgswahrscheinlichkeit in Zeile 1 ist $\geq 1/4l^2$.
Bei $4l^2k$-facher Wiederholung ist die Erfolgswahrscheinlichkeit

$$1 - \left(1 - \frac{1}{4l^2}\right)^{4l^2 k} \geq 1 - \frac{1}{2^k}$$

für genügend große l, denn

$$\lim_{n \to \infty} (1 - 1/n)^n = e^{-1} \leq 1/2.$$

Für $k = l$ und $4l^2k$-facher Wiederholung der Zeile 1 ist die Erfolgswahrscheinlichkeit des gesamten Algorithmus mindestens

$$1 - \left(\underbrace{\frac{1}{2^k}}_{\substack{\text{Fehlerwkt.} \\ \text{Zeile 1}}} + \underbrace{\frac{1}{2^l}}_{\substack{\text{Fehlerwkt.} \\ \text{Zeile 2}}} \right) \geq 1 - \frac{1}{2^k - 1}.$$

Bemerkung 4.14. Der deterministische Primzahltest ist mit einer Laufzeit von $\mathcal{O}(n^6)$ nicht praktikabel. Daher verwendet man in der Praxis häufig den Miller-Rabin-Primzahltest, den wir im Kapitel 8 behandeln werden. Dies ist ein probabilistischer Polynomialzeitalgorithmus mit folgenden Eigenschaften:

1. Ist $x \in \mathbb{P}$, so liefert MillerRabin(x) den Wert 1 mit Wahrscheinlichkeit $= 1$.
2. Ist $n \notin \mathbb{P}$, so liefert MillerRabin(x) den Wert 1 mit Wahrscheinlichkeit $\leq 1/2$.

Bei k-facher Wiederholung des Tests erhält man Wahrscheinlichkeiten 1 bzw. $\leq 1/2^k$.

Wahl des Exponenten e

Gleichverteilt zufällige Wahl eines $e \in \{2, (p-1)(q-1)-1\}$ mit $\text{ggT}(e, (p-1)(q-1)) = 1$:

1. wähle gleichverteilt zufälliges $e \in \{2, (p-1)(q-1)-1\}$.
2. falls $\text{ggT}(e, (p-1)(q-1)) \neq 1$, so stoppe erfolglos
3. return e.

Den größten gemeinsamen Teiler berechnen wir mit dem Euklidischen Algorithmus (in Polynomialzeit). Die Anzahl der zu $(p-1)(q-1)$ teilerfremden Zahlen (d. h. $\text{ggT}(e, (p-1)(q-1)) = 1$) $e \in \{2, (p-1)(q-1)-1\}$ ist nach Definition 4.2 und Satz 4.5.3 genau

$$\varphi((p-1)(q-1)) - 1 \geq \frac{(p-1)(q-1)}{6 \ln \ln((p-1)(q-1))} - 1 \geq \frac{(p-1)(q-1)}{\mathcal{O}(\log l)}.$$

Die Erfolgswahrscheinlichkeit bei $k \cdot \mathcal{O}(\log l)$-facher Wiederholung ist dann mindestens $1 - 2^{-k}$. (gleiches Argument wie oben)

Berechnung des privaten Schlüssels d

Die Berechnung von $d := e^{-1} \bmod (p-1)(q-1)$ läuft über den erweiterten Euklidischen Algorithmus. Er liefert $x, y \in \mathbb{Z}$ mit

$$\text{ggT}((p-1)(q-1), e) = x(p-1)(q-1) + ye.$$

Mit $d := y \bmod (p-1)(q-1)$ erhalten wir den gesuchten Wert.

Wegen $\text{ggT}((p-1)(q-1), e) = 1$ gilt $y \cdot e = 1 - x(p-1)(q-1)$ und damit $d \cdot e \equiv y \cdot e \equiv 1 \bmod (p-1)(q-1)$.

Das Verfahren ist effizient, da der erweiterte Euklidische Algorithmus polynomielle Laufzeit hat.

Modulares Potenzieren

Zwischenbilanz: wir haben uns überzeugt, dass der Schlüsselgenerator des RSA ein probabilistischer Polynomialzeitalgorithmus ist.

Jetzt betrachten wir das Problem der effizienten Berechnung von $m^e \bmod n$. Der naive Algorithmus hat eine Laufzeit von mindestens $e \cdot |n| = \mathcal{O}(2^{|n|}|n|)$. Dies ist jedoch eine exponentielle Laufzeitschranke, da die Länge der Eingabe $((n,e),m)$ zwischen $|n|$ und $3|n|$ liegt.

Für ein effizientes Potenzieren gehen wir wie folgt vor:

- sei $e_t \cdots e_0$ die Binärdarstellung von e
- es gilt
$$m^e = m^{e_t 2^t + e_{t-1}2^{t-1} + \cdots + e_0 2^0} = m^{e_0} \cdot m^{e_1 2} \cdots m^{e_t 2^t}$$
- wegen $m^{2^{i+1}} = (m^{2^i})^2$ lassen sich die $(m^{e_i 2^i} \bmod m)$ in polynomieller Zeit in der Eingabelänge berechnen
- mit obiger Gleichung erhält man insgesamt einen Polynomialzeitalgorithmus

Diese Methode nennt man „square & multiply".

Beispiel 4.15. (square and multiply) Berechnen $3^{37} \bmod 5$

$$\begin{aligned}
&\operatorname{bin}(37) = 100101\\
&\Rightarrow 37 = ((((1\cdot 2+0)\cdot 2+0)\cdot 2+1)\cdot 2+0)\cdot 2+1\\
&\Rightarrow 3^{37} \bmod 5 = \underbrace{\Bigg(\underbrace{\Bigg(\underbrace{\Bigg(\underbrace{\Big(\underbrace{\big(3^1\big)^2 \cdot 3^0 \bmod 5}_{4}\Big)^2 \cdot 3^0 \bmod 5}_{1}\Bigg)^2 \cdot 3^1 \bmod 5}_{3}\Bigg)^2 \cdot 3^0 \bmod 5}_{4}\Bigg)^2 \cdot 3^1 \bmod 5}_{3}\\
&= 3
\end{aligned}$$

Folgerung 4.16. RSA ist ein Kryptosystem.

Beweis
- Schlüsselgenerator $\mathcal{K}$ ist ein probabilistischer Polynomialzeitalgorithmus, da die Wahl der Primzahlen p, q, die Wahl der Exponenten e und die Berechnung des privaten Schlüssels d effizient möglich sind
- $\mathcal{E}$ und $\mathcal{D}$ sind Polynomialzeitalgorithmen, wegen des effizienten, modularen Potenzierens
- aus Satz 4.11 folgt $\mathcal{D}((n,d),\mathcal{E}((n,e),m)) = m$ für alle $m \in \Sigma^{\leq r}$ und alle von $\mathcal{K}(1^r)$ generierten Schlüsselpaare $((n,e),(n,d))$

□

4.2.3 Zur Sicherheit von RSA

Satz 4.17. Falls Faktorisierung in polynomieller Zeit möglich ist, so kann RSA in polynomieller Zeit gebrochen werden.

Beweis Der Angreifer macht folgendes:

1. liest den öffentlichen Schlüssel (n, e), faktorisiert $n = pq$ und kennt damit $\varphi(n) = (p-1)(q-1)$
2. berechnet privaten Schlüssel $d = e^{-1} \bmod (p-1)(q-1)$

□

Es ist nicht bekannt, ob die Umkehrung des Satzes gilt.

Es wird vermutet, dass Faktorisieren und das Brechen von RSA nicht NP-hart sind. Ferner weiß man, dass Quantencomputer in polynomieller Zeit faktorisieren und damit RSA brechen können.

Schwachstellen bei reinem RSA

Reines RSA (auch Plain RSA oder Lehrbuch-RSA genannt) besitzt gravierende Schwachstellen:

- sind nur wenige Klartexte möglich (z. B. ja/nein), können diese verschlüsselt und mit dem abgefangenen c verglichen werden
- falls $m^e < n$ (in der Praxis gilt aus Effizienzgründen häufig $e = 3$), so kann m effizient bestimmt werden, denn $m = \sqrt[e]{c}$ in $\mathbb{N}$
- falls m an k Empfänger mit verschiedenen n_i, aber gleichem e versendet wurde und $m^e < n_1 \cdots n_k$, so kann m effizient bestimmt werden (m^e in $\mathbb{N}$ via chinesischem Restsatz bestimmen, dann $\sqrt[e]{}$)
- falls e klein und für bekanntes δ die Klartexte m und $m+\delta$ mit gleichem (n, e) verschlüsselt, so ist m häufig effizient bestimmbar

Verbesserung: Padded RSA

Die genannten Schwachstellen können duch den Einsatz von Padding-Verfahren beseitigt werden.

Funktionsweise und Eigenschaften von Padded RSA:

- verarbeitet nur Klartexte m, die deutlich kürzer als $|n|$ sind
- hängt an m ein zufälliges Wort vorgegebener Struktur an
  (z. B. $\square s \square$ für ein genügend langes, zufälliges $s \in (\Sigma - \{\square\})^*$)
- die Markierungen $\square$ erlauben nach der Decodierung das Entfernen der hinzugefügten Zeichen
- das Verfahren ist probabilistisch und hat nicht die oben genannten Schwachstellen
- für bestimmte Fälle (z. B. 1 Bit Nachricht und langes s) lässt sich die Sicherheit unter kryptographischen Annahmen beweisen

4.3 Diffie-Hellman-Schlüsselaustausch

Im Jahr 1976 lösten Diffie und Hellman das Schlüsselaustauschproblem: Alice und Bob wollen sich über einen unsicheren Kanal auf einen Schlüssel einigen. Das Diffie-Hellman-Verfahren beendet eine lange Tradition von Codebüchern, die zum Beispiel für jeden Tag des Jahres einen eigenen Code für die Enigma beinhalteten.

Es enthält Ansätze eines Public-Key-Kryptosystems, denn es ermöglicht sichere Kommunikation über einen unsicheren Kanal ohne vorher vereinbarten Schlüssel. Der Unterschied besteht darin, dass beim Erstellen einer verschlüsselten Nachricht beide Parteien miteinander kommunizieren müssen.

1985 veröffentlichte Taher Elgamal ein Public-Key-Kryptosystem, das auf dem Diffie-Hellman-Schlüsselaustausch basiert.

Das Diffie-Hellman-Verfahren basiert auf folgender Idee: das Potenzieren modulo einer Primzahl p ist einfach („square & multiply") – Logarithmieren modulo p dagegen schwer.

4.3.1 Algebraische Grundlagen

Definition 4.18. Sei $n \geq 2$. Ein Element $\gamma \in \mathbb{Z}_n^*$ heißt *Erzeuger* von $\mathbb{Z}_n^*$, oder auch *Primitivwurzel* modulo n, falls

$$\{\gamma^i \mid 0 \leq i < \varphi(n)\} = \mathbb{Z}_n^*$$

(äquivalent hierzu: $\operatorname{ord}_n \gamma = \varphi(n)$).

Satz 4.19. (Gauß) $\mathbb{Z}_n^*$ besitzt genau dann einen Erzeuger, wenn $n \in \{2, 4, p^i, 2p^i \mid p \in \mathbb{P} - \{2\} \text{ und } i \in \mathbb{N}^+\}$.

Ohne Beweis.

Falls p prim ist, so besitzt $\mathbb{Z}_p^*$ viele Erzeuger.

Satz 4.20. Ist $p \in \mathbb{P}$ und $d \mid p-1$, so gibt es in $\mathbb{Z}_p^*$ genau $\varphi(d)$ Elemente der Ordnung d.

Ohne Beweis.

Folgerung 4.21. Ist $p \geq 7$ eine Primzahl, so ist die Anzahl der Erzeuger von $\mathbb{Z}_p^*$ genau

$$\varphi(p-1) \geq \frac{p-1}{6 \ln \ln(p-1)}.$$

Definition 4.22. (Exponential- und Logarithmusfunktion) Sei $p \in \mathbb{P}$ und sei γ ein Erzeuger von $\mathbb{Z}_p^*$. Wir definieren folgende Funktionen $\mathbb{Z}_p^* \to \mathbb{Z}_p^*$.

$$\begin{aligned} \exp_{\gamma,p}(a) &:= \gamma^a \bmod p, \text{ wobei } 1 \leq a \leq p-1 \\ \log_{\gamma,p}(b) &:= \text{das eindeutig bestimmte } a \in \{1, \ldots, p-1\} \\ &\qquad \text{mit } \exp_{\gamma,p}(a) = b, \text{ wobei } 1 \leq b \leq p-1 \end{aligned}$$

Da γ ein Erzeuger von $\mathbb{Z}_p^*$ ist, bildet $\exp_{\gamma,p}$ auf gesamt $\mathbb{Z}_p^*$ ab. Damit ist $\exp_{\gamma,p}$ bijektiv und folglich ist $\log_{\gamma,p}$ wohldefiniert und auch bijektiv.

4.3.2 Das Diffie-Hellman-Protokoll

Alice und Bob möchten sich auf einen gemeinsamen Schlüssel einigen (über einen unsicheren Kanal).

1. Alice und Bob einigen sich auf eine große Primzahl p und einen Erzeuger γ von $\mathbb{Z}_p^*$. (γ und p sind öffentlich)

2. Alice wählt ein zufälliges $a \in \{1, \ldots, p-1\}$ und sendet $\alpha := \exp_{\gamma,p}(a)$.
3. Bob wählt ein zufälliges $b \in \{1, \ldots, p-1\}$ und sendet $\beta := \exp_{\gamma,p}(b)$.
4. Alice berechnet den geheimen Schlüssel $k_A := \beta^a \bmod p$.
   Bob berechnet den geheimen Schlüssel $k_B := \alpha^b \bmod p$.

Satz 4.23. Mit den Bezeichnungen von oben gilt $k_A = k_B \bmod p$.

Beweis $k_A = (\beta^a \bmod p) = ((\gamma^b)^a \bmod p) = ((\gamma^a)^b \bmod p) = (\alpha^b \bmod p) = k_B$. □

Bemerkung 4.24.

- Das Potenzieren ist effizient möglich.
- Ein Angreifer sieht $(\gamma^a \bmod p)$ und $(\gamma^b \bmod p)$, aber nicht $(\gamma^{ab} \bmod p)$.

4.3.3 Beschaffung eines Erzeugers von $\mathbb{Z}_p^*$

Folgender Satz hilft uns beim Test der Eigenschaft $\mathrm{ord}_p(\gamma) = p-1$.

Satz 4.25. Seien $p \in \mathbb{P}$, $\gamma \in \mathbb{Z}_p^*$ und $1 \le d \le p-1$. Dann gilt

$$\mathrm{ord}_p(\gamma) = d \iff \underbrace{\gamma^d \equiv 1 \bmod p \text{ und für alle } c \mid d \text{ mit } c < d \text{ gilt } \gamma^c \not\equiv 1 \bmod p}_{\text{RHS (right-hand side)}}.$$

Beweis

„⇒" sei $\mathrm{ord}_p(\gamma) = d$, also $\gamma^d \equiv 1 \pmod p$
angenommen es existiert ein $c \mid d$ mit $c < d$ und $\gamma^c \equiv 1 \pmod p$.
dann ist $\mathrm{ord}_p(\gamma) \le c < d$ ↯

„⇐" sei $\mathrm{ord}_p(\gamma) \neq d$, wir zeigen ¬RHS

1. Fall: $\gamma^d \not\equiv 1 \pmod p$, dann sind wir fertig

2. Fall: $\gamma^d \equiv 1 \pmod p$
$\Rightarrow \mathrm{ord}_p(\gamma) < d$, also $d = a \cdot \mathrm{ord}_p(\gamma) + r$ mit $a \ge 1$ und $r < \mathrm{ord}_p(\gamma)$
$\Rightarrow 1 \equiv \gamma^d \equiv \gamma^{a \cdot \mathrm{ord}_p(\gamma) + r} \equiv \gamma^r \pmod p$
falls $r \ge 1$, so $\mathrm{ord}_p(\gamma) \le r < \mathrm{ord}_p(\gamma)$ ↯
$\Rightarrow r = 0$
$\Rightarrow \mathrm{ord}_p(\gamma) \mid d, \mathrm{ord}_p(\gamma) < d, \gamma^{\mathrm{ord}_p(\gamma)} \equiv 1 \pmod p$
$\Rightarrow$ ¬RHS (für $c = \mathrm{ord}_p(\gamma)$)

□

Folgerung 4.26. Sei $p \in \mathbb{P}$ ungerade, $\gamma \in \mathbb{Z}_p^*$ und $p-1 = p_1 \cdot \ldots \cdot p_n$ die Zerlegung von $p-1$ in Primfaktoren. Dann gilt

$$\gamma \text{ erzeugt } \mathbb{Z}_p^* \iff \forall i \in \{1, \ldots, n\}, \gamma^{p-1/p_i} \not\equiv 1 \bmod p.$$

Beweis

$$\begin{aligned}
\gamma \text{ erzeugt } \mathbb{Z}_p^* &\iff \mathrm{ord}_p(\gamma) = p-1 \\
&\overset{4.7+4.25}{\iff} \text{für alle } c \mid p-1 \text{ mit } c < p-1 \text{ gilt } \gamma^c \not\equiv 1 \pmod p \qquad (*) \\
&\iff \text{für alle } i = 1, \ldots, n \text{ gilt } \gamma^{(p-1)/p_i} \not\equiv 1 \pmod p \qquad (**)
\end{aligned}$$

Die Implikation $(*) \Rightarrow (**)$ ist klar. $\neg(*) \Rightarrow \neg(**)$ sieht man wie folgt:
falls $y^c \equiv 1 \pmod p$ für $c \mid p-1$ mit $c < p-1$, so teilt c eine der Zahlen $(p-1)/p_i$, also gilt auch $\gamma^{(p-1)/p_i} \equiv 1 \pmod p$ □

Wir wissen:

1. $\mathbb{Z}_p^*$ hat mindestens $(p-1)/(6 \ln \ln(p-1))$ Erzeuger nach Folgerung 4.21.
2. Wenn wir die Primfaktorzerlegung von $p-1$ kennen, so können wir in Polynomialzeit testen, ob ein gegebenes $\gamma \in \mathbb{Z}_p^*$ ein Erzeuger ist (Folgerung 4.26).

Daher interessieren uns Primzahlen p, sodass sich $p-1$ leicht faktorisieren lässt.

Definition 4.27. (Sophie-Germain-Primzahl) Eine Primzahl q heißt Sophie-Germain-Primzahl, falls $2q+1$ eine Primzahl ist.

Man vermutet, dass es in $\{1, \ldots, n\}$ mindestens $n/(\ln n)^2$ Sophie-Germain-Primzahlen gibt. Allerdings konnte man bisher noch nicht einmal klären, ob es unendlich viele solche Primzahlen gibt.

Dessen ungeachtet beschaffen wir uns Erzeuger von $\mathbb{Z}_p^*$ wie folgt:

1. suchen eine große Sophie-Germain-Primzahl q (nach obiger Vermutung finden wir mit hoher Wahrscheinlichkeit eine solche unter $\mathcal{O}(\log^2 n)$ zufällig gewählten Zahlen aus $\{1, \ldots, n\}$)
2. $p := 2q + 1$
3. wählen zufällige $\gamma \in \{2, \ldots, p-2\}$, bis wir einen Erzeuger von $\mathbb{Z}_p^*$ gefunden haben

Da $p - 1 = 2q$, können wir mit Folgerung 4.26 in Polynomialzeit testen, ob γ ein Erzeuger von $\mathbb{Z}_p^*$ ist.

Ein zufällig gewähltes $\gamma \in \{2, \ldots, p-2\}$ ist mit Wahrscheinlichkeit $1/2$ ein Erzeuger von $\mathbb{Z}_p^*$, denn:

- Die Anzahl der Erzeuger von $\mathbb{Z}_p^*$ ist $\varphi(\varphi(p)) = \varphi(p-1) = \varphi(2) \cdot \varphi(q) = q - 1 = \frac{p-1}{2} - 1 = \frac{p-3}{2}$.
- Alle Erzeuger liegen in $\{2, \ldots, p-2\}$, denn 1 und $p-1$ sind keine Erzeuger.
- $|\{2, \ldots, p-2\}| = p - 3$

Variante des Diffie-Hellman-Protokolls

Das Protokoll wird auch in einer Variante verwendet, bei der γ nur eine Untergruppe von $\mathbb{Z}_p^*$ erzeugt. Dort werden p und γ wie folgt gewählt:

- wähle $p, q \in \mathbb{P}$ mit $p = r \cdot q + 1$ für ein $r \in \mathbb{N}$
- $G := \{x^r \bmod p \mid x \in \mathbb{Z}_p^*\}$ ist Untergruppe von $\mathbb{Z}_p^*$ mit $|G| = q$
- wähle $\gamma \in G - \{1\}$ gleichverteilt zufällig (wähle gleichverteilt zufälliges $x \in \mathbb{Z}_p^*$, $\gamma := (x^r \bmod p)$, teste $\gamma \neq 1$)
- γ erzeugt G, denn $|G|$ ist prim (Satz von Lagrange)

Der restliche Teil des Diffie-Hellman-Protokolls bleibt unverändert.

Vorteil dieser Variante: die Erzeugersuche ist trivial

4.3.4 Sicherheit des Diffie-Hellman-Schlüsselaustausches

Satz 4.28. Wenn der Diffie-Hellman Schlüsselaustausch nicht in Polynomialzeit gebrochen werden kann, so lässt sich der diskrete Logarithmus für Sophie-Germain-Primzahlen nicht in Polynomialzeit berechnen.

Beweis Angenommen es existiert ein Polynomialzeitalgorithmus zur Berechnung des diskreten Logarithmus, d. h. zu einem gegebenen Erzeuger γ von $\mathbb{Z}_p^*$ und gegebenem $\alpha \in \mathbb{Z}_p^*$ lässt sich in Polynomialzeit $\log_{\gamma,p}(\alpha)$ berechnen.

Ein Angreifer auf den Schlüsselaustausch hört $\alpha = \exp_{\gamma,p}(a)$ ab und berechnet $\log_{\gamma,p}(\alpha) = a$. Anschließend hört er $\beta = \exp_{\gamma,p}(b)$ ab und berechnet den gemeinsamen Schlüssel $k = \beta^a \bmod p$. □

Bemerkungen:

1. Der Satz 4.28 liefert nur eine obere Schranke für die Sicherheit.
2. Die Umkehrung des Satzes ist nicht bekannt, sie wäre eine untere Schranke und damit ein Indiz für die Sicherheit.
3. Quantencomputer können effizient diskrete Logarithmen berechnen und damit das Diffie-Hellman-Verfahren brechen.
4. Es ist möglich, dass die Einschränkung auf Sophie-Germain-Primzahlen die Berechnung diskreter Logarithmen vereinfacht.
5. Ein weiteres Problem stellen Man-in-the-middle-Angriffe dar, bei dem die Kommunikation zwischen Alice und Bob über einen Kanal abgewickelt wird, der komplett durch den Angreifer Mallory kontrolliert wird. In dieser Situation kann sich Mallory für Alice als Bob ausgeben und umgekehrt, da der Diffie-Hellman-Schlüsselaustausch keine Authentifizierung vorsieht.

   $$\text{Alice} \overset{\text{Diffie-Hellman}}{\underset{\text{Schlüssel } a}{\longleftrightarrow}} \text{Mallory} \overset{\text{Diffie-Hellman}}{\underset{\text{Schlüssel } b}{\longleftrightarrow}} \text{Bob}$$

   Alice hält a und Bob hält b für den gemeinsamen Schlüssel.

4.4 Das Elgamal-Kryptosystem

Dieses Verfahren basiert auf dem Diffie-Hellman-Schlüsselaustausch und wurde 1985 von Taher Elgamal veröffentlicht.

4.4.1 Das Elgamal-Protokoll

Sei q eine Sophie-Germain-Primzahl. Das Senden einer Nachricht von Alice an Bob funktioniert wie folgt.

1. Bob
   - setzt $p := 2q + 1 \in \mathbb{P}$
   - wählt einen zufälligen Erzeuger γ von $\mathbb{Z}_p^*$
   - wählt ein zufälliges $b \in \{1, \ldots, p-1\}$
   - berechnet $B := \gamma^b \bmod p$
   - veröffentlicht öffentlichen Schlüssel (p, γ, B), hält privaten Schlüssel (p, γ, b) geheim
2. Alice

- stellt ihre Nachricht als Folge von Zahlen $m \in \{1, \ldots, p-1\}$ (d. h. p-adisch) dar, verschlüsselt jedes m wie folgt:
- wählt ein zufälliges $a \in \{1, \ldots, p-1\}$
- berechnet $A := \gamma^a \bmod p$
- berechnet $c := B^a m \bmod p$
- sendet den Chiffretext (A, c) an Bob

3. Bob berechnet $x := p - 1 - b$ und entschlüsselt durch

$$A^x c \equiv A^{p-1} A^{-b} c \equiv \gamma^{a(p-1)} \gamma^{-ab} \gamma^{ab} m \equiv m \bmod p.$$

Die Korrektheit ergibt sich direkt aus der eben geführten Rechnung.

Definition 4.29. (Elgamal-Kryptosystem) Sei $q \in \mathbb{P}$ eine Sophie-Germain-Primzahl. Das Elgamal-Kryptosystem ist definiert durch $\text{Elgamal}_q := (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit $p := 2q + 1$, $\Sigma = \mathbb{Z}_p$ sowie:

- $\mathcal{K}(1^n)$ leistet folgendes:
  1. Wähle gleichverteilt zufällig $\gamma \in \{2, \ldots, p-2\}$.
  2. Falls γ kein Erzeuger von $\mathbb{Z}_p^*$ ist, so stoppe erfolglos.
  3. $B := \gamma^b \bmod p$ für zufälliges $b \in \{1, \ldots, p-1\}$.
  4. return $((p, \gamma, B), (p, \gamma, b))$
- $\mathcal{E}((p, \gamma, B), m_1 \cdots m_n)$ mit $m_i \in \mathbb{Z}_p^*$ leistet folgendes:
  1. Wähle gleichverteilt zufällig $a_1 \cdots a_n \in \{1, \ldots, p-1\}$.
  2. Berechne $A_i := \gamma^{a_i} \bmod p$ und $c_i := B^{a_i} m_i \bmod p$ für alle $i = 1, \ldots, n$.
  3. return $[(A_1, c_1), \ldots, (A_n, c_n)]$
- $\mathcal{D}((p, \gamma, b), (A_1, c_1) \cdots (A_n, c_n)) = m_1 \cdots m_n$, wobei $m_i := A_i^{p-1-b} c_i \bmod p$.

Da wir nicht wissen, ob es unendlich viele Sophie-Germain-Primzahlen gibt, haben wir das Elgamal-Kryptosystem für ein festes q definiert. Die Suche nach Erzeugern von $\mathbb{Z}_p^*$ kann, wie im Abschnitt zum Diffie-Hellman-Schlüsselaustausch geklärt, effizient durchgeführt werden.

Ein Nachteil des Verfahrens ist, dass die Chiffretexte doppelt so lang wie die Klartexte sind.

4.4.2 Sicherheit des Verfahrens

Zur Sicherheit lässt sich Ähnliches wie beim Diffie-Hellman-Schlüsselaustausch sagen. Ein Vorteil des Elgamal-Kryptosystems ist die Randomisierung der Verschlüsselung; gleiche Klartexte erzeugen im Allgemeinen verschiedene Chiffretexte.

Zusätzlich ist es für Alice empfehlenswert, für jedes m_k einen neuen Exponenten a_k zu wählen: hat man nämlich

$$m \mapsto (\gamma^a \bmod p, \gamma^{ab} m \bmod p),$$
$$m' \mapsto (\gamma^a \bmod p, \gamma^{ab} m' \bmod p)$$

und kennt ein Angreifer schon m, so kann er auch m' aus dem Chiffretext rekonstruieren:

$$\gamma^{ab} m' (\gamma^{ab} m)^{-1} m \equiv m' \bmod p.$$

4.5 Goldwasser-Micali-Kryptosystem

Wir behandeln im Folgenden das historisch erste probabilistische Public-Key-Kryptosystem aus dem Jahre 1982. Es ist unter bestimmten kryptographischen Annahmen beweisbar sicher. Jedoch ist es ineffizient, da Chiffretexte wesentlich länger werden als zugehörige Klartexte.

Definition 4.30. Seien $n \geq 2$ und $x \in \mathbb{Z}_n^*$.

- x heißt *quadratischer Rest modulo* n, falls es ein $y \in \mathbb{Z}$ mit $x \equiv y^2 \bmod n$ gibt.
- x heißt *quadratischer Nichtrest modulo* n, falls es kein $y \in \mathbb{Z}$ mit $x \equiv y^2 \bmod n$ gibt.

4.5.1 Das Goldwasser-Micali-Protokoll

Alice möchte Bob eine Nachricht schicken. Dies läuft wie folgt ab:

1. Bob
   - wählt zufällige, unabhängige Primzahlen $p, q \in \mathbb{P}$ mit $p \equiv q \equiv 3 \bmod 4$ und $p \neq q$
   - veröffentlicht öffentlichen Schlüssel $n = pq$, hält privaten Schlüssel (p, q) geheim
2. Alice möchte eine Bitfolge $b_1, \ldots, b_m$ an Bob schicken. Sie verschlüsselt wie folgt:
   - wählt zufällige, unabhängige $y_1, \ldots, y_m \in \mathbb{Z}_n^*$
   - $c_i := ((-1)^{b_i} y_i^2) \bmod n$ für $i = 1, \ldots, m$
   - sendet $c_1, \ldots, c_m$ an Bob
3. Bob entschlüsselt $c_1, \ldots, c_m$ zu $b'_1, \ldots, b'_m$ durch

$$b_i = \begin{cases} 0 & \text{falls } c_i \text{ quadratischer Rest mod } n \text{ ist,} \\ 1 & \text{sonst.} \end{cases}$$

Definition 4.31. (Goldwasser-Micali-Kryptosystem) Sei $\mathrm{GM}_\lambda := (\{0,1\}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, wobei $\lambda \in \mathbb{N}$ der Sicherheitsparameter sei, der die Länge des Chiffretextes einzelner Bits festlegt, und folgendes gilt:

- $\mathcal{K}(1^m)$ leistet folgendes:
  1. Wähle verschiedene, zufällige Primzahlen $p, q \leq 2^{\lceil \lambda/2 \rceil}$ mit $2^{\lambda-1} \leq pq \leq 2^\lambda$ und $p \equiv q \equiv 3 \bmod 4$, und setze $n := pq$.
  2. return $(n, (p, q))$
- $\mathcal{E}(n, b_1 \cdots b_m)$ gibt $c_1 \cdots c_m$ mit $c_i := ((-1)^{b_i} y_i^2) \bmod n$ aus, wobei die $y_i \in \mathbb{Z}_n^*$ zufällig und unabhängig gewählt werden.
- $\mathcal{D}((p, q), c_1 \cdots c_m) := b'_1 \cdots b'_m$ durch

$$b_i = \begin{cases} 0 & \text{falls } c_i \text{ quadratischer Rest mod } n \text{ ist,} \\ 1 & \text{sonst.} \end{cases}$$

4.5.2 Korrektheit des Verfahrens

Definition 4.32. Für ungerade $p \in \mathbb{P}$ und $x \in \mathbb{Z}_p^*$ ist das *Legendre-Symbol* wie folgt definiert.

$$\left(\frac{x}{p}\right) := \begin{cases} 1 & \text{falls } x \text{ quadratischer Rest mod } p \text{ ist} \\ -1 & \text{falls } x \text{ quadratischer Nichtrest mod } p \text{ ist} \end{cases}$$

Satz 4.33. (Euler-Kriterium) Für ungerade $p \in \mathbb{P}$ und $x \in \mathbb{Z}_p^*$ gilt

$$\left(\frac{x}{p}\right) \equiv x^{\frac{p-1}{2}} \bmod p.$$

Beweis

1. Fall: x ist quadratischer Rest modulo p
dann existiert ein $y \in \mathbb{Z}$ mit $x \equiv y^2 \bmod p$
für $y' := y \bmod p$ gilt $x \equiv y'^2 \bmod p$ und $y' \in \mathbb{Z}_p^*$ (denn $x \neq 0$ und damit auch $y' \neq 0$)
$\overset{\text{kleiner Fermat 4.7}}{\Longrightarrow} x^{\frac{p-1}{2}} \equiv y'^{p-1} \equiv \underbrace{1}_{=\left(\frac{x}{p}\right)} \bmod p$

2. Fall: x ist quadratischer Nichtrest modulo p
sei γ ein Erzeuger von $\mathbb{Z}_p^*$
dann ist $x \equiv \gamma^m \bmod p$ für ein ungerades $m = 2k+1$ mit $1 \leq m \leq p-1$ (andernfalls wäre $x \equiv y^2 \bmod p$ für $y = \gamma^{m/2}$)
$x^{\frac{p-1}{2}} \equiv \gamma^{(2k+1)(p-1)/2} \equiv \gamma^{k(p-1)} \gamma^{\frac{p-1}{2}} \overset{\text{kleiner Fermat}}{\equiv} \gamma^{\frac{p-1}{2}} \equiv \underbrace{-1}_{=\left(\frac{x}{p}\right)} \bmod p$

die letzte Äquivalenz sieht man wie folgt:

- γ erzeugt $\mathbb{Z}_p^*$, also ex. eindeutig bestimmtes $r \in \{1, \ldots, p-2\}$ mit $\gamma^r \equiv -1 \bmod p$
- $1 \overset{\text{kleiner Fermat}}{\equiv} \gamma^{p-1-r} \cdot \gamma^r \bmod p$ und damit
$$-1 \equiv \gamma^{p-1-r} \cdot \gamma^r \cdot (-1) \equiv \gamma^{p-1-r} \bmod p$$
- aus $p-1-r \in \{1, \ldots, p-2\}$ folgt $r = p-1-r$ (da r eindeutig ist) und $r = \frac{p-1}{2}$.

□

Folgerung 4.34. Für ungerade $p, q \in \mathbb{P}$ mit $p \neq q$ und $x \in \mathbb{Z}_{pq}^*$ gilt

$$x \text{ ist quadratischer Rest mod } pq \quad \Leftrightarrow \quad x^{\frac{p-1}{2}} \bmod p = x^{\frac{q-1}{2}} \bmod q = 1.$$

Beweis x ist quadratischer Rest mod $p \cdot q$
$\Leftrightarrow \exists y \in \mathbb{Z}$ mit $y^2 \equiv x \bmod pq$
$\Leftrightarrow \exists y \in \mathbb{Z}$ mit $pq \mid y^2 - x$
$\overset{p \neq q}{\Longleftrightarrow} \exists y \in \mathbb{Z}$ mit $p \mid y^2 - x$ und $q \mid y^2 - x$
$\Leftrightarrow \exists y \in \mathbb{Z}$ mit $y^2 \equiv x \bmod p$ und $y^2 \equiv x \bmod q$
$\Leftrightarrow \exists y_p, y_q \in \mathbb{Z}$ mit $y_p^2 \equiv x \bmod p$ und $y_q^2 \equiv x \bmod q$ (Richtung „⇐" gilt, denn nach chinesischem Restsatz ex. $y \in \mathbb{Z}_{pq}$ mit $y \equiv y_p \bmod p$ und $y \equiv y_q \bmod q$)
$\Leftrightarrow x_p := (x \bmod p)$ ist quadratischer Rest mod p und $x_q := (x \bmod q)$ ist quadratischer Rest mod q
$\Leftrightarrow \left(\frac{x_p}{p}\right) = \left(\frac{x_q}{q}\right) = 1$

$$\overset{4.32}{\iff} \underbrace{x_p^{\frac{p-1}{2}} \bmod p}_{=x^{\frac{p-1}{2}} \bmod p} = \underbrace{x_q^{\frac{q-1}{2}} \bmod q}_{=x^{\frac{q-1}{2}} \bmod q} = 1 \qquad \square$$

Folgerung 4.35. Für ungerade $p, q \in \mathbb{P}$ mit $p \equiv q \equiv 3 \bmod 4$ und $q \neq p$ ist $pq - 1$ ein quadratischer Nichtrest modulo pq.

Beweis Angenommen $pq - 1$ ist ein quadratischer Rest mod pq. Aus 4.34 folgt $(pq-1)^{\frac{p-1}{2}} \bmod p = 1$ und damit $-1^{\frac{p-1}{2}} \equiv 1 \bmod p$. Also ist $p-1/2$ gerade und damit $4 \mid p-1$. D. h. $p \equiv 1 \bmod 4$. Dies widerspricht $p \equiv 3 \bmod 4$. $\square$

Wir begründen die Korrektheit des Goldwasser-Micali-Kryptosystems wie folgt:

- Man weiß, dass für $k \to \infty$ folgendes gilt:
  Die Hälfte der Primzahlen $p \leq k$ hat die Eigenschaft $p \equiv 3 \bmod 4$.
- Im Kryptosystem sind die Primzahlen p, q so gewählt, dass $p \equiv q \equiv 3 \bmod 4$ und $p \neq q$. Weiter gilt $n = pq$.
- Nach Folgerung 4.35 ist $n - 1$ ein quadratischer Nichtrest mod n.
- Falls $b_i = 0$, so ist $c_i = y_i^2 \bmod n$ ein quadratischer Rest mod n.
- Falls $b_i = 1$, so ist $c_i = (-1 \cdot y_i^2) \bmod n$. Wäre dies ein quadratischer Rest mod n, so wäre $-1 \cdot y_i^2 \equiv x^2 \bmod n$ für ein $x \in \mathbb{Z}_n^*$, also $n - 1 \equiv (xy_i^{-1})^2 \bmod n$ und damit wäre $n - 1$ ein quadratischer Rest mod n, ein Widerspruch. Also ist c_i ein quadratischer Nichtrest modulo n.
- Insgesamt gilt:
  $$b_i = 0 \;\Leftrightarrow\; c_i \text{ ist quadratischer Rest mod } n.$$
- Es folgt $b_i = b_i'$ und damit die Korrektheit des Verfahrens.

4.5.3 Effizienz des Verfahrens

Der Schlüsselgenerator arbeitet in probabilistischer Polynomialzeit, da asymptotisch die Hälfte der Primzahlen die Eigenschaft $p \equiv 3 \bmod 4$ hat.

Da Bob die Faktorisierung des öffentlichen Schlüssels $n = pq$ kennt, kann er nach 4.34 wie folgt entschlüsseln.

$$b_i' = \begin{cases} 0 & \text{falls } c_i^{\frac{p-1}{2}} \bmod p = c_i^{\frac{q-1}{2}} \bmod q = 1, \\ 1 & \text{sonst} \end{cases}$$

Damit lassen sich die b_i' effizient berechnen (square and multiply). Die restlichen Komponenten des Verfahrens lassen sich effizient implementieren.

4.5.4 Sicherheit des Verfahrens

Falls GM_λ effizient gebrochen werden kann, so kann man effizient bestimmen, ob c ein quadratischer Rest modulo n ist (zumindest für $n \in \mathbb{N}$ mit genau zwei verschiedenen Primfaktoren $p, q \in \mathbb{P}, p \equiv q \equiv 3 \bmod 4$).

Man vermutet, dass sich dieses Problem nicht effizient lösen lässt.

4.5.5 Anwendung des Goldwasser-Micali-Protokolls beim Commitment

Bob möchte eine Nachricht (z. B. eine Entscheidung $b \in \{0, 1\}$) so hinterlegen, dass gilt:

- Nachricht für keinen lesbar
- Bob kann Nachricht nachträglich nicht ändern
- Nachricht kann später aufgedeckt werden, d. h. Bob kann seine Entscheidung belegen

So funktioniert das Commitment mithilfe des Goldwasser-Micali-Protokolls:

- Bob wählt p, q, y wie bei GM und veröffentlicht $n = p \cdot q$ und $c = (-1)^b y^2 \bmod n$.
- Bob veröffentlicht $c = (-1)^b y^2 \bmod n$.
- Damit ist b nicht lesbar und kann nachträglich nicht geändert werden.
- Bob kann seine Entscheidung b durch das Veröffentlichen von (p, q) belegen.

Beispiel 4.36. (Münzwurf via Telefon) Alice und Bob wollen gemeinsam ein Zufallsbit erzeugen, ohne dass einer von beiden betrügen kann.

1. Alice:
   - wählt p_a, q_a wie im Goldwasser-Micali-Protokoll und setzt $n_a = p_a \cdot q_a$
   - wählt zufälliges $y_a \in \mathbb{Z}_{n_a}^*$
   - erzeugt Zufallsbit b_a (Münzwurf)
   - berechnet $c_a = ((-1)^{b_a} \cdot y_a^2) \bmod n_a$
   - sendet (c_a, n_a) an Bob
2. Bob:
   - wählt $p_b, q_b, n_b, y_b, b_b, c_b$ analog zu Alice
   - sendet (c_b, n_b) an Alice
3. Alice und Bob:
   - veröffentlichen p_a, q_a, p_b, q_b
   - prüfen $p_a \equiv q_a \equiv p_b \equiv q_b \equiv 3 \pmod 4$
   - berechnen b_a und b_b
   - $b := b_a \oplus b_b$ (Ergebnis des Münzwurfs)

$\Rightarrow b$ ist zufällig, falls mindestens eine Partei ehrlich ist

4.6 Sicherheit von Public-Key-Kryptosystemen

Für Public-Key-Kryptosysteme existieren Sicherheitsbegriffe verschiedener Stärke. Eine relativ schwache Forderung ist die Folgende.

Definition 4.37. Ein Kryptosystem $S = (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ besitzt die *Einwegeigenschaft* genau dann, wenn für jeden probabilistischen Polynomialzeit-Algorithmus A, jedes Polynom q und für fast alle $n \in \mathbb{N}$ die Erfolgswahrscheinlichkeit von A bei folgendem Protokoll höchstens $\frac{1}{q(n)}$ ist:

1. Bob erzeugt ein Schlüsselpaar (e, d) mittels $\mathcal{K}(1^n)$.
2. Bob wählt zufällig $m \in \Sigma^n$ und berechnet $c := \mathcal{E}(e, m)$.

3. A ist erfolgreich, falls $A(1^n, e, c)$ die Nachricht m ausgibt.

Dies ist eine relativ schwache Forderung, da beispielsweise nicht gefordert ist, dass das System bei Kenntnis von zwei verschiedenen Chiffretexten $\mathcal{E}(e, m_1)$ und $\mathcal{E}(e, m_2)$ noch sicher ist.

Häufig fordert man die semantische Sicherheit von Kryptosystemen: bei Wahl von m_1, m_2 und Kenntnis von $c \in \{\mathcal{E}(e, m_1), \mathcal{E}(e, m_2)\}$ soll ein Angreifer nicht herausfinden können, ob $\mathcal{D}(d, c) = m_1$ oder $\mathcal{D}(d, c) = m_2$ ist.

Definition 4.38. Ein Kryptosystem $S = (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ ist genau dann *semantisch sicher*, wenn für jeden probabilistischen Polynomialzeitalgorithmus A, jedes Polynom q und für fast alle $n \in \mathbb{N}$ die Erfolgswahrscheinlichkeit von A bei folgendem Protokoll höchstens $\frac{1}{2} + \frac{1}{q(n)}$ beträgt:

1. Bob erzeugt mittels $\mathcal{K}(1^n)$ ein Schlüsselpaar (e, d) und sendet den öffentlichen Schlüssel e an A.

2. A wählt verschiedene $m_1, m_2 \in \Sigma^n$ und sendet diese an Bob.

3. Bob wählt zufällig $m \in \{m_1, m_2\}$ und sendet $c := \mathcal{E}(e, m)$ an A.

4. A muss angeben, ob Bob m_1 oder m_2 für m gewählt hat.

Als direkte Konsequenz erkennt man, dass semantisch sichere Verfahren probabilistisch sein müssen. Andernfalls wäre A erfolgreich, indem es $\mathcal{E}(e, m_1)$ und $\mathcal{E}(e, m_2)$ berechnet und mit c vergleicht.

Man vermutet, dass das Goldwasser-Micali-Kryptosystem semantisch sicher ist. Unter kryptographischen Annahmen kann man dies auch beweisen.

Kapitel 5

Digitale Signatur

Neben Vertraulichkeit (durch Verschlüsselung) möchten wir nun auch folgendes sicherstellen:

- Integrität: Fälschung oder Manipulation der Nachricht nicht möglich
- Authentizität: Nachweis, dass die erhaltene Nachricht von einem bestimmten Absender stammt
- Verbindlichkeit: Absender kann die Urheberschaft an seinen Nachrichten nicht leugnen

Hierfür verwendet man digitale Signaturen. Diese können meist nur kurze Dokumente direkt signieren – längere Dokumente werden zuvor mit einer kryptographischen Hashfunktion komprimiert.

5.1 Kryptographische Hashfunktionen

Definition 5.1. Eine *Hashfunktion* über einem Alphabet Σ ist eine Abbildung

$$h : \Sigma^* \to \Sigma^n$$

für ein $n \in \mathbb{N}$.

Beispiel 5.2. Die Funktion $h : \{0,1\}^* \to \{0,1\}$ mit

$$h(x) = \begin{cases} 1 & \text{falls Anzahl der 1en in } x \text{ ungerade,} \\ 0 & \text{sonst,} \end{cases}$$

ist eine Hashfunktion.

Hashfunktionen werden bei der Implementierung von assoziativen Arrays (auch bekannt als Dictionaries, Hashes, Maps) verwendet.

In kryptographischen Anwendungen eingesetzte Hashfunktionen dürfen nicht leicht invertierbar sein.

Definition 5.3. Eine Funktion $f : \Sigma^* \to \Sigma^*$ heißt *Einwegfunktion*, wenn f in Polynomialzeit berechenbar ist und für jeden probabilistischen Polynomialzeitalgorithmus A und jedes Polynom p für fast alle $n \in \mathbb{N}$ folgendes gilt:

$$\sum_{x \in \Sigma^n} \frac{1}{2^n} \cdot P\Big(\underbrace{\underbrace{A(f(x), 1^n)}_{\substack{\text{Ausgabe von } A \text{ auf} \\ \text{Eingabe } (f(x), 1^n)}} \in f^{-1}(f(x))}_{\substack{\text{Wkt., dass } A(f(x),1^n) \text{ ein} \\ \text{Urbild von } f(x) \text{ ausgibt}}} \Big) \leq \underbrace{\frac{1}{p(n)}}_{\substack{\text{geringe} \\ \text{Erfolgswkt.}}} .$$

Nach dieser Definition ist f effizient berechenbar und es gibt keinen effizienten Algorithmus A, der $f(x)$ mit akzeptabler Wahrscheinlichkeit invertieren kann.

Es ist allerdings nicht bekannt, ob es solche Einwegfunktionen gibt. Man vermutet und hofft, dass das Potenzieren eines Erzeugers von $\mathbb{Z}_p^*$ eine Einwegfunktion ist (da diskrete Logarithmen bisher nicht effizient berechnet werden können). An kryptographische Hashfunktionen werden jedoch noch weitere Anforderungen gestellt.

Definition 5.4. Sei h eine Hashfunktion. Eine *Kollision* von h ist ein Paar $(x, y) \in (\Sigma^*)^2$ mit $x \neq y$ und $h(x) = h(y)$.

Eine Hashfunktion heißt *kollisionsresistent*, falls es praktisch nicht möglich ist, eine Kollision von h zu berechnen.

Wir verwenden diese informale Beschreibung des Begriffs. Formal wird er über Familien von Hashfunktionen definiert. Man kann dann zeigen, dass kollisionsresistente Hashfunktionen immer Einwegfunktionen sind.

Definition 5.5. Eine *kryptographische Hashfunktion* ist eine Hashfunktion, die kollisionsresistent ist (insbesondere ist diese dann eine Einwegfunktion).

Analog ist auch hier nicht bekannt, ob es solche Funktionen gibt. Die folgenden Hashfunktionen betrachtet oder betrachtete man als Kandidaten für kryptographische Hashfunktionen:

Hash-funktion	Anzahl der Bits der Hashwerte	Verfahren entwickelt	Kollision gefunden	Bemerkung
MD5	128	1991	2005	Aus kryptographischer Sicht gebrochen; Verfahren nicht mehr empfohlen.
MD6	256	2008	—	—
SHA-1	160	1995	2017	Von NSA entwickelt; im Digital Signature Standard (DSS) verwendet.
SHA-2	256/512	2001	—	Von NSA entwickelt.
SHA-3	256/512	2012	—	NIST (National Institute of Standards and Technology) nahm bei Standartisierung Änderungen am Algorithmus vor; Forscher kritisieren, dass diese Änderungen die Sicherheit beeinträchtigen.

5.2 RSA-Signatur

Die Idee zur RSA-Signatur besteht darin, dass Alice einen Text x mit einer kryptographischen Hashfunktion h zu $m := h(x)$ komprimiert. Nun entschlüsselt Alice m und sendet $(x, \mathcal{D}((n, d), m))$ an Bob. Bob prüft nun

$$\mathcal{E}((n, e), \mathcal{D}((n, d), m)) \stackrel{?}{=} h(x).$$

5.2.1 Das Protokoll

Alice möchte einen signierten Text x an Bob senden. Dies geschieht wie folgt.

1. Alice erzeugt ein Schlüsselpaar wie beim RSA-Kryptosystem:

- wählt gleichverteilt zufällig verschiedene Primzahlen p, q
- $n := pq$, damit gilt $\varphi(n) = (p-1)(q-1)$
- wählt zufälliges $e \in \{2, \ldots, \varphi(n)-1\}$ mit $\mathrm{ggT}(e, \varphi(n)) = 1$
- berechnet $d := e^{-1} \bmod \varphi(n)$
- veröffentlicht public key (n, e), hält private key (n, d) geheim
- $h : \{0,1\}^* \to \{0, \ldots, n-1\}$ sei eine öffentlich bekannte kryptographische Hashfunktion

2. Alice signiert:
   - berechnet den Hashwert $m := h(x)$
   - berechnet die Unterschrift so:
   $$\mathrm{sig}_A(m) = m^d \bmod n.$$
   - sendet $(x, \mathrm{sig}_A(m))$ an Bob
3. Bob prüft $h(x) = \mathrm{sig}_A(m)^e \bmod n$.

Bemerkung 5.6.

- Die Korrektheit des Verfahrens zeigt man wie beim RSA:
$$\mathrm{sig}_A(m)^e \equiv m^{d \cdot e} \overset{4.11}{\equiv} m \equiv h(x) \bmod n$$
- Zum Fälschen einer Unterschrift muss man in der Lage sein, verschlüsselte Nachrichten an Alice zu entschlüsseln.
- Das Prinzip der RSA-Signatur lässt sich allgemein auf alle Public-Key-Kryptosysteme mit folgender Eigenschaft übertragen:
$$\forall m : \ \mathcal{E}(e, \mathcal{D}(d, m)) = m$$
D. h. Verschlüsselung und Entschlüsselung dürfen vertauscht werden.

5.3 Elgamal-Signatur

Beim Elgamal-Verfahren sind Verschlüsselung und Entschlüsselung nicht vertauschbar. Hier signiert Alice einen Text x wie folgt.

1. Alice erzeugt Schlüsselpaar:
   - wählt zufällige Primzahl p und einen Erzeuger $\gamma \in \mathbb{Z}_p^*$
   - wählt ein zufälliges $a \in \{1, \ldots, p-2\}$
   - berechnet $A := \gamma^a \bmod p$
   - veröffentlicht public key (p, γ, A), hält private key (p, γ, a) geheim
   - $h : \{0,1\}^* \to \{0, \ldots, p-1\}$ sei eine öffentlich bekannte kryptographische Hashfunktion
2. Alice signiert:
   - wählt zufälliges $k \in \{1, \ldots, p-2\}$ mit $\mathrm{ggT}(k, p-1) = 1$
   - berechnet $r := \gamma^k \bmod p$, $k' := k^{-1} \bmod p-1$, $s := k' \cdot (h(x) - ar) \bmod p-1$ (falls $s = 0$, so wähle k, r, s erneut)

- sendet (x, r, s) an Bob

3. Bob verifiziert:
   - $1 \leq r \leq p-1$ und $0 < s < p-1$
   - $A^r r^s \equiv \gamma^{h(x)} \bmod p$

Idee der Elgamal-Signatur

Als erste, naheliegende Idee könnte Alice $h(x) - a$ senden, da nur sie das erzeugen kann. Hierbei entsteht jedoch das Problem, dass Bob a erkennen kann.

Als Verbesserung sendet Alice $h(x) - a$ in der maskierten Form

$$\overbrace{k^{-1} \cdot (\boxed{h(x)} - a \cdot \boxed{\gamma^k})}^{s}, \overbrace{\boxed{\gamma^k}}^{r}$$

für ein zufälliges k mit $\mathrm{ggT}(k, p-1) = 1$. Dies hat die gewünschten Eigenschaften:

- nur Alice kann den Wert s erzeugen
- Bob sieht nur $\boxed{\gamma^a}$, $\boxed{\gamma^k}$, $\boxed{h(x)}$, aber nicht a und k
- Unterschrift kann geprüft werden

Satz 5.7. (Korrektheit der Elgamal-Signatur) Mit den obigen Bezeichnungen gilt $A^r r^s \equiv \gamma^{h(x)} \bmod p$.

Beweis Wir rechnen modulo p.

$$\begin{aligned}
A^r \cdot r^s \equiv \gamma^{a \cdot r} \cdot \gamma^{k \cdot s} &\equiv \gamma^{a \cdot r + k \cdot [k' \cdot (h(x) - ar) \bmod p-1]} \\
&\equiv \gamma^{a \cdot r + k \cdot [k' \cdot (h(x) - ar) + i(p-1)]} \text{ für ein geeignetes } i \in \mathbb{Z} \\
&\overset{(*)}{\equiv} \gamma^{a \cdot r + k \cdot k' \cdot (h(x) - ar)} \cdot \gamma^{k \cdot i \cdot (p-1)} \\
&\equiv \gamma^{a \cdot r + h(x) - ar + j \cdot (p-1)(h(x) - ar)} \cdot \gamma^{k \cdot i \cdot (p-1)} \text{ für ein geeignetes } j \in \mathbb{Z} \\
&\equiv \gamma^{a \cdot r + h(x) - ar} \cdot \gamma^{(p-1)[j(h(x) - ar) + k \cdot i]} \\
&\overset{(**)}{\equiv} \gamma^{h(x)} \pmod p
\end{aligned}$$

$(*)$ $k \cdot k' = 1 + j \cdot (p-1)$, denn k' ist das multiplikative Inverse mod $p-1$
$(**)$ es gilt $\gamma^{(p-1)} \equiv 1 \bmod p$ nach kleinem Satz von Fermat □

Bemerkung 5.8.

- Der Wert k muss für jede Signatur zufällig neu gewählt werden.
- Eine effizientere Variante der Elgamal-Signatur wird im Digital Signature Standard verwendet.

5.4 Lamport-Diffie-Einmal-Signatur

Die bisherigen Verfahren benötigten kryptographische Hashfunktionen. Das folgende Verfahren benötigt nur eine Einwegfunktion und basiert damit auf einer vermutlich schwächeren Voraussetzung.

Sei $H : \{0,1\}^* \to \{0,1\}^*$ eine längentreue Einwegfunktion, d. h. $|H(x)| = |x|$ für alle $x \in \{0,1\}^*$. Das Verfahren funktioniert nun wie folgt, wenn Alice einen Text $d = d_1 \cdots d_n$ mit $d_1, \ldots, d_n \in \{0,1\}$ signiert:

1. Alice erzeugt Schlüsselpaar:
   - wählt einen Sicherheitsparameter $k \in \mathbb{N}$
   - erzeugt eine Liste mit $2n$ zufälligen Worten aus $\{0,1\}^k$:

     $$X = (x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, \ldots, x_{n,0}, x_{n1})$$

   - erzeugt $Y = (y_{1,0}, y_{1,1}, y_{2,0}, y_{2,1}, \ldots, y_{n,0}, y_{n1})$ mit $y_{ij} := H(x_{ij})$
   - veröffentlicht Verifikationsschlüssel Y und hält Signaturschlüssel X geheim
2. Alice signiert:
   - erzeugt $S = (s_1, \ldots, s_n)$ mit

     $$s_i = x_{i,d_i} = \begin{cases} x_{i,0} & \text{falls } d_i = 0 \\ x_{i,1} & \text{falls } d_i = 1 \end{cases}$$

   - sendet (d, S) an Bob
3. Bob verifiziert $(H(s_1), \ldots, H(s_n)) = (y_{1,d_1}, \ldots, y_{1,d_n})$

Bemerkung 5.9.

- Beim Signieren wird der geheime Schlüssel X teilweise veröffentlicht. Daher darf X nicht mehrfach verwendet werden (Einmal-Signatur).
- Zum Fälschen der Signatur braucht man die Urbilder der Elemente von Y, also die x_{ij}. Da man eine Einwegfunktion zum Erzeugen von Y aus X benutzt, sind diese Urbilder nach Voraussetzung nicht praktisch bestimmbar.
- Es wird nicht erkannt, wenn das Nachrichtenende abgeschnitten wird. Man kann dies verhindern, indem man das Nachrichtenende markiert (z. B. durch den Text „Nachrichtenende").

Kapitel 6

Identifikation

Wir widmen uns nun der Frage, wie man sicherstellen kann, dass man mit einer bestimmten Person und nicht mit einem Betrüger kommuniziert.

Eine Identifikation durch Senden eines Passworts hat deutliche Nachteile: im Klartext übertragene Passwörter können abgehört werden; zudem kennt der Verifier nach eine Identifikation das Passwort des Provers und könnte sich mit diesem Wissen gegenüber anderen Systemen als Prover ausgeben.

Eine bessere Identifikationsstrategie sollte zwei Dinge erfüllen:

1. ein Betrüger darf sich nicht als Prover ausgeben können
2. der Prover soll bei der Identifikation sein Geheimnis nicht verraten

6.1 Einmal-Passwörter

Der Prover besitzt eine Liste $w_1, \ldots, w_n$ geheimer Passwörter. Der Verifier besitzt eine Liste von Funktionswerten $f(w_1), \ldots, f(w_n)$ der Passwörter unter eine Einwegfunktion f.

Zur Identifikation werden die Passwörter der Reihe nach benutzt, d. h. es wird (k, w_k) für ein $k \in \{1, \ldots, n\}$ an den Verifier gesendet.

1. Ein Betrüger kann sich nicht als Prover ausgeben, da keine Information über die Passwörter öffentlich ist.
2. Der Verifier kennt die unbenutzten Passwörter des Provers nicht, da sich f praktisch nicht invertieren lässt.

An diesem Verfahren ist nachteilhaft, dass zu Beginn eine lange Passwort-Liste erzeugt und ausgetauscht werden muss.

6.2 Challenge-Response Identifikation

Bei diesem Verfahren ist die Idee, dass der Verifier dem Prover eine Frage stellt, deren Beantwortung nur mit Hilfe von Provers Geheimnis möglich ist. Die Antwort lässt keine Rückschlüsse auf das Geheimnis zu.

6.2.1 Das Fiat-Shamir-Protokoll

Das Protokoll wurde 1986 entwickelt und ist ein Zero-Knowledge-Protokoll; dies bedeutet, dass der Verifier sich mit hoher Wahrscheinlichkeit überzeugen kann, dass der Prover ein gewisses Geheimnis besitzt, ohne dass der Provder verwertbare Informationen über sein Geheimnis preisgeben muss.

Die Sicherheit des Fiat-Shamir-Protokolls basiert auf der Vermutung, dass die Berechnung von Quadratwurzeln modulo n nicht effizient durchführbar ist.

Im Folgenden ist das Protokoll dargestellt.

1. Prover erzeugt Geheimnis und öffentlichen Schlüssel:
   - wählt zufällig zwei große Primzahlen $p, q \in \mathbb{P}$
   - setzt $n := pq$
   - wählt ein Geheimnis $s \in \mathbb{Z}_n^*$
   - berechnet $v := s^2 \bmod n$
   - veröffentlicht Schlüssel (v, n)
2. Prover identifiziert sich beim Verifier:
   (a) Prover wählt eine zufällige Zahl $r \in \mathbb{Z}_n^*$ und sendet $x := r^2 \bmod n$ an den Verifier
   (b) Verifier wählt ein zufälliges Bit $a \in \{0, 1\}$ und sendet es an den Prover
   (c) Prover sendet $y := r \cdot s^a \bmod n$ an den Verifier
   (d) Verifier prüft $x \in \mathbb{Z}_n^*$ und $y^2 \equiv x \cdot v^a \bmod n$

Satz 6.1. (Korrektheit des Protokolls) Sind Prover und Verifier ehrlich, so ist die Identifikation erfolgreich.

Beweis $y^2 \equiv r^2 \cdot (s^a)^2 \equiv x \cdot (s^2)^a \equiv x \cdot v^a \bmod n$. □

Sicherheit gegen Betrug

Kann sich ein Betrüger B für den öffentlichen Schlüssel (v, n) mit Erfolgswkt. signifikant größer $1/2$ identifizieren, so kann er mit hoher Wkt. das Geheimnis (Quadratwurzel von v modulo n) bestimmen:

Satz 6.2. Für alle $k \in \mathbb{N}$, alle $\varepsilon > 0$ und alle prob. Polyzeit-Algos B existiert ein prob. Polyzeit-Algo C, sodass für alle $n = pq$ mit $p, q \in \mathbb{P}$ und alle $v = s^2 \bmod n$ mit $s \in \mathbb{Z}_n^*$ gilt:

Falls sich B bei Eingabe (v, n) mit Erfolgswkt. $\geq 1/2 + \varepsilon$ identifizieren kann, so berechnet C bei Eingabe (v, n) mit Erfolgswkt. $\geq 1 - 1/2^k$ eine Quadratwurzel von v modulo n.

Beweis Seien k, ε, B wie im Satz. Wir beschreiben den probabilistischen Polynomialzeit-Algorithmus C' bei Eingabe (v, n) mit $v \in \mathbb{Z}_n^*$:

1. simuliere eine Runde des Fiat-Shamir-Protokolls mit B als Prover:
   2. simuliere B bis x gesendet wird
   3. setze Simulation einmal mit $a = 0$ und einmal mit $a = 1$ fort; die von B gesendeten y seien y_0 und y_1
4. if $(y_1 \cdot y_0^{-1})^2 \equiv v \bmod n$ then return $(y_1 \cdot y_0^{-1} \bmod n)$

C' arbeitet in polynomieller Zeit.

Seien p, q, n, s, v wie im Satz, sodass sich B bei Eingabe (v, n) mit Erfolgswkt. $\geq 1/2 + \varepsilon$ identifizieren kann.

Erfolgswahrscheinlichkeit von C' bei Eingabe (v, n):

- jede der beiden Simulationen in Zeile 3 hat Erfolgswahrscheinlichkeit $\geq \varepsilon$
  (andernfalls wäre Erfolgswahrscheinlichkeit von $B < \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \varepsilon = \frac{1}{2} + \frac{\varepsilon}{2}$ ↯)
- Wahrscheinlichkeit, dass beide Simulationen in Zeile 3 erfolgreich: $\geq \varepsilon^2$

- in diesem Fall gilt in Zeile 4:

$$y_0^2 \equiv x \pmod{n}$$
$$y_1^2 \equiv x \cdot v \pmod{n}$$

$$\text{aus } x, v \in \mathbb{Z}_n^* \text{ folgt } y_0, y_1 \in \mathbb{Z}_n^*$$

$$\Rightarrow y_0^2 \equiv y_1^2 v^{-1} \pmod{n}$$
$$v \equiv \Big(\underbrace{y_1 \cdot y_0^{-1}}_{\substack{\text{Quadratwurzel} \\ \text{von } v}} \Big)^2 \pmod{n}$$

- $\Rightarrow C'$ liefert mit Wahrscheinlichkeit $\geq \varepsilon^2$eine Quadratwurzel von v modulo n

$\frac{k}{\varepsilon^2}$-fache Wiederholung von C' liefert Algorithmus C mit Erfolgswahrscheinlichkeit $\geq 1 - \frac{1}{2^k}$ $\square$

Vermutung: Jeder prob. Polyzeit-Algo findet nur für einen verschwindend kleinen Teil aller (v, n) mit hoher Erfolgswkt. eine Quadratwurzel von v modulo n.

Aus Satz 6.2 folgt:
Falls die Vermutung stimmt, so gilt für alle $\varepsilon > 0$ und jeden prob. Polyzeit-Algo B, dass er sich nur für einen verschwindend kleinen Teil aller öffentlichen Schlüssel (v, n) mit Erfolgswkt. $\geq 1/2 + \varepsilon$ identifizieren kann.

D.h. für fast alle (v, n) hat B eine Erfolgswkt. $\leq 1/2 + \varepsilon$ und bei mehrfacher Anwendung des Protokolls sinkt diese auf $\leq 1/2^k$.

Fazit: Falls obige Vermutung stimmt, so ist das Fiat-Shamir-Protokoll gegen Betrug sicher.

Bemerkung 6.3. Bei einfacher Anwendung des Protokolls erreicht ein Betrüger die Erfolgswahrscheinlichkeit 1/2, indem er das Bit rät, das ihm der Verifier senden wird:
Falls $a = 0$, so wählt er ein $r \in \mathbb{Z}_n^*$ und sendet $x := r^2 \bmod n$.
Falls $a = 1$, so wählt er ein $y \in \mathbb{Z}_n^*$ und sendet $x := y^2 v^{-1} \bmod n$.

Sicherheit gegen Abhören

Satz 6.4. Beim Fiat-Shamir-Protokoll lässt sich keine Information aus der Kommunikation zwischen dem Prover und einem Verifier (oder einem Betrüger, der sich als Verifier ausgibt) gewinnen.

Beweis Seien r_i, x_i, a_i, y_i die in der Runde i gewählten Werte.

1. Fall: $a_i = 0$.
$y_i = r_i \in \mathbb{Z}_n^*$ gleichverteilt und unabhängig von anderen Runden
$x_i = (r_i^2 \bmod n)$
$\Rightarrow$ Beobachter sieht gleichverteiltes, unabhängiges $y_i \in \mathbb{Z}_n^*$ und dessen Quadrat
diese Information kann er selbst erzeugen

2. Fall: $a_i = 1$.
$y_i = (r_i \cdot s \bmod n)$ für ein $r_i \in \mathbb{Z}_n^*$, das gleichverteilt und unabhängig von anderen Runden gewählt wurde
$x_i = (r_i^2 \bmod n)$
$\Rightarrow y_i \in \mathbb{Z}_n^*$ gleichverteilt und unabhängig von anderen Runden,
da es für alle $s, y_i \in \mathbb{Z}_n^*$ genau ein $r_i \in \mathbb{Z}_n^*$ mit $(y_i = r_i \cdot s \bmod n)$ gibt
$x_i \equiv r_i^2 \equiv y_i^2 \cdot v^{-1} \bmod n$

$\Rightarrow$ Beobachter sieht gleichverteilt unabhängiges $y_i \in \mathbb{Z}_n^*$
und den Wert $(y_i^2 \cdot v^{-1} \bmod n)$
diese Information kann er selbst erzeugen

□

Protokolle, die Aussagen im Stil von 6.4 erlauben, nennt man Zero-Knowledge-Protokolle.

6.3 Zero-Knowledge-Beweise

Die beim Fiat-Shamir-Protokoll kennengelernte Zero-Knowledge-Eigenschaft lässt sich ganz allgemein auf Situationen übertragen, in denen eine Partei einer anderen Partei etwas beweisen soll. Versteht man unter einem Beweis eine Information, mit deren Hilfe man eine Aussage effizient überprüfen kann, so führt dies zur Definition der Komplexitätsklasse NP.

Wir werden sehen: Falls das Goldwasser-Micali-Commitment sicher ist, besitzt jedes Problem in NP einen Zero-Knowledge-Beweis.

6.3.1 Ein historischer Zero-Knowledge-Beweis

Im Januar 1535 fand zwischen den italienischen Mathematikern Nicolò Tartaglia und Antonio Maria Fior ein öffentlicher Gelehrten-Wettkampf statt. Fior behauptete, ein Verfahren zur Lösung von Gleichungen der Form $x^3 + px = q$ mit $p, q \in \mathbb{N}$ zu besitzen.

Fior und Tartaglia stellten sich gegenseitig 30 Aufgaben, die innerhalb einer bestimmten Frist gelöst werden sollten. Tartaglia strengte sich an, fand Anfang Februar 1535 das Lösungsverfahren und konnte alle Aufgaben Fiors lösen. Fior hingegen konnte keine von Tartaglias Aufgaben lösen.

Der Ablauf des Wettkampfs gleicht einem Zero-Knowledge-Beweis:

- zeigt den Kenntnisstand beider Mathematiker
- liefert keine Informationen über das Verfahren selbst

6.3.2 Ein Zero-Knowledge-Beweis für Graphisomorphie

Zwei ungerichtete Graphen $G_0 = (V_0, E_0)$ und $G_1 = (V_1, E_1)$ sind isomorph, falls es eine Bijektion $\sigma : V_0 \to V_1$ gibt, sodass für alle $v, w \in V_0$ gilt: $\{v, w\} \in E_0 \iff \{\sigma(v), \sigma(w)\} \in E_1$.

Definition 6.5. Das *Graphisomorphieproblem* ist definiert durch

$$\mathrm{GI} = \{(G_0, G_1) \mid G_0 \text{ und } G_1 \text{ sind ungerichtete, isomorphe Graphen}\}.$$

Es gilt GI $\in$ NP, aber man weiß nicht, ob GI in P liegt. Man vermutet, dass GI nicht NP-vollständig ist.

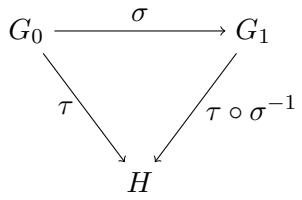
Die Isomorphie von G_0 und G_1 lässt sich durch Angabe des Isomorphismus σ beweisen.

Es ist aber auch folgender Zero-Knowledge-Beweis möglich.

Peggy möchte Victor überzeugen, dass G_0 und G_1 isomorph sind, ohne den ihr bekannten Isomorphismus $\sigma : G_0 \to G_1$ zu verraten. Hierbei nehmen wir o. B. d. A. an, dass G_0 und G_1 die gleiche Kantenmenge besitzen, d. h. $G_0 = (V, E_0)$ und $G_1 = (V, E_1)$.

1. Peggy wählt zufällige Permutationen $\tau : V \to V$, wendet diese auf G_0 an und sende den entstehenden Graphen H an Victor. (H ist zu G_0 und G_1 isomorph.)
2. Victor wählt zufälliges $c \in \{0, 1\}$ und sendet es an Peggy. (Victor erwartet von Peggy einen Isomorphismus $G_c \to H$.)
3. Ist $c = 0$, so sendet Peggy $\pi := \tau$, andernfalls $\pi := \tau\sigma^{-1}$.

4. Victor prüft, ob π tatsächlich ein Isomorphismus $G_c \to H$ ist.



Korrektheit: Sind G_0 und G_1 isomorph, so kann Peggy durch die Kenntnis von σ Victor in jedem Fall überzeugen.

Sicherheit: Sind G_0 und G_1 nicht isomorph, so gilt für jeden von einem Betrüger gesendeten Graphen H, dass G_0 nicht isomorph H oder G_1 nicht isomorph H. Mit Wahrscheinlichkeit $\geq 1/2$ wählt Victor c so, dass G_c nicht isomorph H. In diesem Fall bemerkt er den Betrug in Zeile 4. Bei k-facher Anwendung des Protokolls ist die Betrugswahrscheinlichkeit $\leq 1/2^k$.

Zero-Knowledge: Im Verlauf des Protokolls sieht Victor eine zufällige Permutation $\pi : V \to V$ und den Graphen $H := \pi(G_c)$. Diese Informationen kanner sich selbst durch die zufällige Wahl einer Permutation $\pi : V \to V$ beschaffen.

6.3.3 Ein Zero-Knowledge-Beweis für 3-Färbbarkeit

Ein ungerichteter Graph $G = (\{1, \ldots, n\}, E)$ ist 3-färbbar, falls es $c_1, \ldots, c_n \in \{1, 2, 3\}$ gibt, sodass $c_i \neq c_j$ für alle $\{i, j\} \in E$.

Definition 6.6. Das *3-Färbbarkeitsproblem* ist definiert durch

$$3\text{–FARB} = \{G \mid G \text{ ist 3-färbbarer, ungerichteter Graph}\}.$$

Man weiß, dass dieses Problem NP-vollständig ist.

Die 3-Färbbarkeit von G lässt sich durch Angabe einer entsprechenden Färbung geweisen. Es ist aber auch folgender Zero-Knowledge-Beweis möglich.

Peggy möchte Victor überzeugen, dass der ungerichtete Graph $G = (V, E)$ mit $V = \{1, \ldots, n\}$ 3-färbbar ist, ohne die ihr bekannte 3-Färbung $c_1, \ldots, c_n \in \{1, 2, 3\}$ zu verraten.

Wir verwenden das Goldwasser-Micali-Commitment, um eine Information x in verschlüsselter Form $y = \text{GMC}(x)$ zu hinterlegen.

1. Peggy wählt zufällige Permutation $\pi : \{1, 2, 3\} \to \{1, 2, 3\}$, berechnet permutierte Färbung $c'_1, \ldots, c'_n$ mit $c'_i = \pi(c_i)$ und sendet $y_1, \ldots, y_n$ mit $y_i = \text{GMC}(c'_i)$ an Victor.
2. Victor wählt zufällige Kante $\{a, b\} \in E$ und sendet sie an Peggy.
3. Peggy öffnet die Commitments y_a und y_b. D. h. sie beweist, dass sie den Knoten a mit c'_a und den Knoten b mit c'_b gefärbt hatte.
4. Victor prüft, ob $c'_a, c'_b \in \{1, 2, 3\}$ und $c'_a \neq c'_b$.

Korrektheit: Ist $G \in 3\text{–FARB}$, so kann Peggy durch die Kenntnis der Färbung $c_1, \ldots, c_n$ Victor in jedem Fall überzeugen.

Sicherheit: Ist $G \notin 3\text{–FARB}$, so besitzt jede von einem Betrüger hinterlegte Färbung $c'_1, \ldots, c'_n$ einen Fehler, d. h. $\{a, b\} \in E$ mit $c'_a = c'_b$ oder $c'_a \notin \{1, 2, 3\}$ oder $c'_b \notin \{1, 2, 3\}$. Victor bemerkt den Fehler mit Wahrscheinlichkeit $\geq 1/n^2$. Bei $k \cdot n^2$-facher Anwendung des Protokolls ist die Betrugswahrscheinlichkeit $\leq 1/2^k$.

Zero-Knowledge: Im Verlauf des Protokolls sieht Victor neben den Commitments $y_1, \ldots, y_n$ nur die Farben $c'_a \neq c'_b$. Dabei handelt es sich um zwei zufällige Zahlen aus $\{1, 2, 3\}$, die er sich auch selbst beschaffen kann. Setzt man die Sicherheit des Goldwasser-Micali-Commitments voraus, so erhält Victor aus $y_1, \ldots, y_n$ keine zusätzlichen Informationen.

Beispiel 6.7. (Zero-Knowledge-Beweis für 3-FARB) Professor G. möchte seine Vorlesungsteilnehmenden davon überzeugen, dass ein vorgelegter Graph 3-färbbar ist, ohne seine Färbung preiszugeben.

Dazu markiert er im Geheimen auf einem Zettel die Knoten in einer korrekten 3-Färbung und verdeckt jeden Knoten mit einer Münze. Die Vorlesungsteilnehmenden überlegen sich eine Kante, die Professor G. aufdecken muss. Sind die aufgedeckten zwei Knoten verschieden gefärbt, wird der Vorgang mit einer neuen Permutation der Farben so oft wiederholt, bis der gesamte Kurs hinreichend davon überzeugt ist, dass Professor G. tatsächlich eine 3-Färbung für den gegebenen Graphen kennt.

Werden jedoch zwei Knoten mit einer gleichen Farbe oder wird ein Knoten mit einer ungültigen Farbe aufgedeckt, so enttarnt die Zuhörerschaft, dass der Professor sie betrogen hat und tatsächlich gar keine 3-Färbung für den Graphen kennt.

6.3.4 Zero-Knowledge-Beweis für beliebige NP-Probleme

Sei $A \in \mathrm{NP}$, d. h. es existiert ein $B \in \mathrm{P}$ und ein Polynom p, sodass

$$x \in A \iff \exists y \, (|y| \leq p(|x|) \wedge (x, y) \in B).$$

Ein solches y ist ein Beweis für die Aussage $x \in A$.

Wegen der NP-Vollständigkeit von 3–FARB gibt es eine totale, in polynomieller Zeit berechenbare Funktion f mit

$$x \in A \iff f(x) \in \text{3–FARB}.$$

Die Reduktionen $A \leq \mathrm{3SAT} \leq$ 3–FARB zeigen, dass sich die Beweise y in polynomieller Zeit in 3-Färbungen übersetzen lassen.

D. h. f kann so gewählt werden, dass es ein in polynomieller Zeit berechenbares g gibt, sodass für alle y mit $|y| \leq p(|x|)$ gilt:

$$(x, y) \in B \iff g(x, y) \text{ ist eine 3-Färbung des Graphen } f(x).$$

Peggy kann Victor wie folgt von der Aussage $x \in A$ überzeugen, ohne ihm Informationen über den Beweis y zu geben.

1. Peggy und Victor einigen sich auf ein f mit obiger Eigenschaft.
2. Beide berechnen $G = f(x)$.
3. Peggy liefert Victor einen Zero-Knowledge-Beweis für $G \in$ 3–FARB.
4. Ist der Beweis korrekt, so gilt $f(x) \in$ 3–FARB und damit $x \in A$.

Schritt 3 ist möglich, da Peggy den Beweis y in polynomieller Zeit in die 3-Färbung $g(x, y)$ des Graphen G übersetzen kann.

Korrektheit, Sicherheit und Zero-Knowledge-Eigenschaft folgen aus dem Zero-Knowledge-Beweis für 3–FARB.

Kapitel 7

Secure Multiparty Computation

In diesem Abschnitt geht es um verschiedene kryptographische Aspekte von Berechnungen, an denen mehrere Parteien beteiligt sind.

7.1 Secret Sharing

Das Secret Sharing erlaubt die Aufteilung eines Geheimnisses auf n Personen, sodass sich nur in Anwesenheit von t Personen das Geheimnis rekonstruieren lässt.

Definition 7.1. Ein (n,t)-*Secret-Sharing Protokoll* mit $n,t \in \mathbb{N}$ ist ein Protokoll, welches ein Geheimnis auf n Personen so aufteilt, dass eine beliebige Auswahl von t Personen das Geheimnis rekonstruieren kann und keine Auswahl von weniger als t Personen relevante Informationen über das Geheimnis erhalten kann.

Das Shamir-Secret-Sharing-Protokoll verwendet die Lagrange-Interpolation, um ein Polynom zu berechnen, das durch gegebene Punkte verläuft. Diese funktioniert in beliebigen Körpern, wir benötigen sie für die Körper $\mathbb{F}_p$ mit $p \in \mathbb{P}$.

Satz 7.2. (Lagrange-Interpolation) Sei $p \in \mathbb{P}$, $t \in \mathbb{N}^+$ und $x_1, \ldots, x_t, y_1, \ldots, y_t \in \mathbb{F}_p$ mit paarweise verschiedenen x_i. Dann gibt es genau ein Polynom $a \in \mathbb{F}_p[x]$ vom Grad kleiner t mit $a(x_k) = y_k$ für $k = 1, \ldots, t$, nämlich

$$a(x) = \sum_{i=1}^{t} y_i \cdot \prod_{j \in \{1,\ldots,t\}-\{i\}} \frac{x - x_j}{x_i - x_j}.$$

Beweis Der Grad von a ist kleiner t, da in jedem Produkt nur $t-1$ Faktoren $\frac{x-x_j}{x_i-x_j}$ auftreten.

Für $k = 1, \ldots, t$ gilt $a(x_k) = y_k \cdot \prod_{j \in \{1,\ldots,t\}-\{k\}} \frac{x_k - x_j}{x_k - x_j} = y_k$, denn für $i \neq k$ taucht der Faktor $x_k - x_k$ im Produkt auf, diese Produkte sind also 0.

$\Rightarrow$ Polynom a hat die gewünschte Eigenschaft

Angenommen, es gibt ein weiteres Polnom $a' \in \mathbb{F}_p[x]$ vom Grad kleiner t mit $a'(x_k) = y_k$ für $k = 1, \ldots, t$.

$\Rightarrow a - a' \in \mathbb{F}_p[x]$ ist vom Grad kleiner t und hat die Nullstellen $x_1, \ldots, x_t$.

Fundamentalsatz der Algebra: Anzahl der Nullstellen eines nicht konstanten Polynoms ist kleiner gleich seinem Grad.

$\Rightarrow a - a'$ ist konstant und hat eine Nullstelle (z. B. x_1)

$\Rightarrow a - a'$ ist konstant 0, d. h. $a = a'$

$\Rightarrow$ Polynom a ist eindeutig bestimmt $\square$

Das Shamir-Secret-Sharing-Protokoll mit Parametern (n,t) arbeitet wie folgt:

1. Initialisierung:
   Der Dealer wählt eine Primzahl $p \geq n+1$ und paarweise verschiedene Elemente $x_1, \ldots, x_n \in \mathbb{Z}_p^*$ und veröffentlicht diese. (Hier ist kein Zufall nötig, wir können z. B. $x_k := k$ wählen.)

2. Verteilung des Geheimnisses $s \in \mathbb{Z}_p$:
   Der Dealer wählt geheime, zufällige $a_1, \ldots, a_{t-1} \in \mathbb{Z}_p$ und konstruiert das Polynom $a(x) := a_{t-1}x^{t-1} + \cdots + a_1 x + s \in \mathbb{F}_p[x]$. Nun berechnet er die Geheimnisteile $y_i := a(x_i)$ für $i = 1, \ldots, n$ und sendet y_i an den Teilnehmer i.

3. Rekonstruktion
   O. B. d. A. wollen die Teilnehmer $1, \ldots, t$ das Geheimnis rekonstruieren. Nach Satz 7.2 ergibt sich das Geheimnis durch

$$s = a(0) = \sum_{i=1}^{t} y_i \cdot \prod_{j \in \{1,\ldots,t\}-\{i\}} \frac{x_j}{x_j - x_i}.$$

Beispiel 7.3. (Shamir-Secret-Sharing) Informationen, die $\boxed{\text{geheim}}$ gehalten werden, sind durch einen solchen Kasten markiert.

1. Initialisierung
   wir rechnen im $\mathbb{F}_p$ mit $p = 71$
   wollen Geheimnis $\boxed{s = 57}$ auf $n = 5$ Personen aufteilen,
   sodass $t = 3$ beliebige Personen das Geheimnis rekonstruieren können
   $x_i := i$ für $i = 1, \ldots, 5$

2. Verteilung des Geheimnisses
   wählen geheime $\boxed{a_1 = 69}, \boxed{a_2 = 41} \in \mathbb{Z}_{71}$
   das geheime Polynom ist $\boxed{a(x) = 41 \cdot x^2 + 69 \cdot x + 57}$
   senden folgendes an die Personen:

$$\begin{aligned}
y_1 &= a(1) = 25 && \to \text{Person 1}\\
y_2 &= a(2) = 4 && \to \text{Person 2}\\
y_3 &= a(3) = 65 && \to \text{Person 3}\\
y_4 &= a(4) = 66 && \to \text{Person 4}\\
y_5 &= a(5) = 7 && \to \text{Person 5}
\end{aligned}$$

3. Rekonstruktion des Geheimnisses durch die Personen 1, 3, 5
   obige Formel liefert:

$$\begin{aligned}
s &= \sum_{i \in \{1,3,5\}} y_i \cdot \prod_{j \in \{1,3,5\}-\{i\}} \frac{x_j}{x_j - x_i}\\
&= 25 \cdot \frac{3}{3-1} \cdot \frac{5}{5-1} + 65 \cdot \frac{1}{1-3} \cdot \frac{5}{5-3} + 7 \cdot \frac{1}{1-5} \cdot \frac{3}{3-5} \mod 71\\
&= 25 \cdot \frac{15}{2^3} - 65 \cdot \frac{5 \cdot 2}{2^3} + 7 \cdot \frac{3}{2^3} \mod 71\\
&= (375 - 650 + 21) \cdot 2^{-3} \mod 71\\
&\overset{(*)}{=} (375 - 650 + 21) \cdot 36^3 \mod 71\\
&= 57
\end{aligned}$$

   $(*)$ wegen $2^{-1} \equiv 36 \mod 71$

Sicherheit des Shamir-Secret-Sharing

Das Protokoll bietet perfekte Sicherheit, denn weniger als t Teilnehmer können keine Information über das Geheimnis ermitteln:

- Angenommen die Teilnehmer $1, \dots, t-1$ möchten Information über das Geheimnis gewinnen.
- Nach Satz 7.2 gibt es für jedes $y \in \mathbb{Z}_p$ genau ein $p_y \in \mathbb{F}_p[x]$ vom Grad kleiner t mit $p_y(x_k) = y_k$ für $k = 1, \dots, t-1$ und $p_y(0) = y$.
- Damit ist aus Sicher der Teilnehmer $1, \dots, t-1$ jedes Geheimnis $y \in \mathbb{Z}_p$ gleich wahrscheinlich.

7.2 Das Millionärsproblem

Wir betrachten folgendes Problem: zwei Superreiche möchten wissen, wer von beiden reicher ist, ohne die Höhe ihres Vermögens zu offenbaren. Zur Lösung verwenden wir das RSA-Kryptosystem und eine Einwegfunktion f. Wir beschreiben folgendes Protokoll zwischen Alice und Bob mit zugehörigen Vermögen a bzw. b.

1. Bobs Schlüsselpaar sei $(\mathrm{pk}, \mathrm{sk}) = ((n, e), (n, e'))$. Beide Parteien einigen sich auf eine Obergrenze $m < n$, welche größer ist als beide Vermögen a und b.
2. Alice wählt eine zufällige Zahl $x \leq n-1$ und verschlüsselt diese mit Bobs öffentlichem Schlüssel zu $c = \mathcal{E}(\mathrm{pk}, x) = x^e \bmod n$ und sendet $d := (c - a) \bmod n$ an Bob.
3. Bob berechnet $y_i := \mathcal{D}(\mathrm{sk}, d + i) = (d + i)^{e'} \bmod n$ und $z_i := f(y_i)$ für $i = 0, \dots, m$ und sendet folgende Zahlen in beliebiger Reihenfolge:

   $$\{z_0, z_1, \dots, z_b, z_{b+1} + 1, z_{b+2} + 1, \dots, z_m + 1\}$$

   (Genau ein y_i ist gleich x, aber Bob weiß nicht welches. Falls $b \geq a$, so enthält die gesendete Liste $f(x)$, sonst $f(x) + 1$.)
4. Falls Alice in der gesendeten Liste den Wert $f(x)$ findet, so gilt $a \leq b$, andernfalls findet sie $f(x) + 1$ und es gilt $a > b$. Am Ende teil Alice Bob das Ergebnis mit.

Bemerkung 7.4.

- In Schritt 1 kann man z. B. $m = 10^8$ als Obergrenze wählen und das Protokoll für $\lfloor a/1000 \rfloor$ und $\lfloor b/1000 \rfloor$ durchführen. So lassen sich Vermögen bis 10^{11}€ mit einer Genauigkeit von 1000€ vergleichen.
- Falls in Schritt 3 in der Liste $z_0, \dots, z_m$ Zahlen z_i, z_j mit $|z_i - z_j| \leq 1$ auftreten, so muss das Protokoll neu gestartet werden, da Bob sonst unter Umständen sowohl $f(x)$ als auch $f(x) + 1$ sendet.
- In Schritt 3 wird die Einwegfunktion f benötigt, da Alice sonst $\mathcal{E}(\mathrm{pk}, y_i)$ bilden kann und so sieht, ab welcher Stelle Bob inkrementiert hat.

Beispiel 7.5. (Millionärsproblem) Vermögen von Alice: $a = 4$
Vermögen von Bob: $b = 2$
verwenden RSA mit $\mathrm{pk} = (77, 7)$ und $\mathrm{sk} = (77, 43)$

x	2	17	20	25	29	70
$\mathcal{E}(\mathrm{pk}, x)$	51	52	48	53	50	49

verwenden folgende Einwegfunktion f:

x	2	17	20	25	29	70
$f(x)$	7	21	24	18	12	31

1. Obergrenze: $m = 5$
2. Alice wählt zufälliges $x = \boxed{17}$, berechnet $c = \mathcal{E}(\mathrm{pk}, x) = 17^7 \bmod 77 = \boxed{52}$ und sendet $d = (c - a) \bmod 77 = \boxed{48}$ an Bob.
3. Bob berechnet für $i = 0, \ldots, 5$ die Werte $y_i = \mathcal{D}(\mathrm{sk}, d+i) = (d+i)^{43} \bmod 77$ und $z_i = f(y_i)$

i	0	1	2	3	4	5
y_i	20	70	29	2	$x = \boxed{17}$	25
z_i	24	31	12	7	$\boxed{21}$	18

Bob sendet:

	24	31	12	8	$\boxed{22}$	19	in beliebiger Reihenfolge
z. B.	8	12	19	$\boxed{22}$	24	31	

4. Alice prüft, ob $f(x) = 21$ in der Liste ist

   Ja: $b \geq a$

   Nein: $b < a$

   $\Rightarrow$ Alice weiß nun, dass $b < a$ gilt.

Der Weg von $\boxed{x}$ ist umkästelt, und $\|$ zeigt die Höhe von Bobs Vermögen.

7.3 Secure Circuit Evaluation

Nun betrachten wir ein allgemeineres Problem: Alice kennt einen Algorithmus zur Berechnung einer bestimmten Funktion f und Bob besitzt Daten x. Gemeinsam soll $f(x)$ bestimmt werden, ohne f oder x zu verraten.

Zur Lösung verwenden wir das Goldwasser-Micali-Kryptosystem, um einzelne Bits zu verschlüsseln. Seien $p, q \in \mathbb{P}$ mit $p \equiv q \equiv 3 \bmod 4$, $p \neq q$ und $n := pq$. Für $b \in \{0, 1\}$ und $k \in \mathbb{Z}_n^*$ sei

$$E(b, k) := (-1)^b k^2 \bmod n.$$

Zur Erinnerung:

- $E(b, k)$ entspricht im Goldwasser-Micali-Kryptosystem dem Verschlüsseln des Bits b mit der Zufallszahl k.
- Kennt man die Faktorisierung $n = pq$, so lässt sich b aus $E(b, k)$ effizient ermitteln.
- Es ist kein effizienter Algorithmus bekannt, der dies ohne die Faktorisierung leistet.

Simulation von Berechnungen durch Schaltkreise

FP bezeichne die Menge der in polynomieller Zeit berechenbaren Funktionen $\mathbb{N} \to \mathbb{N}$.

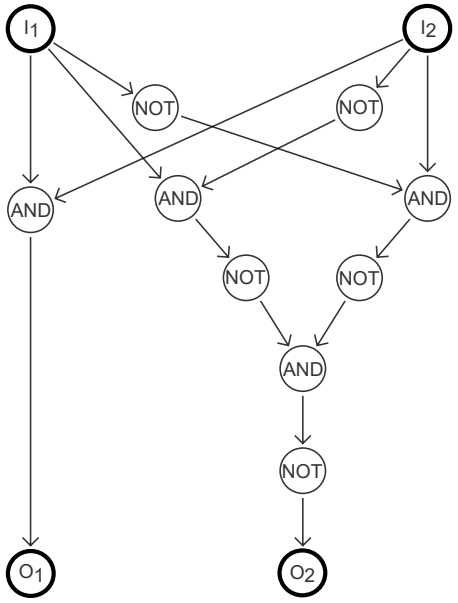
Polynomialzeit-Berechnungen lassen sich durch Schaltkreise polynomieller Größe simulieren. Genauer gilt:

Ist $f \in \mathrm{FP}$, so lässt sich in polynomieller Zeit ein Schaltkreis C_n mit n Eingängen konstruieren, der bei Eingabe einer n-stelligen Binärzahl x den Wert $f(x)$ in Binärdarstellung berechnet. C_n leistet also die Funktionsberechnung für Eingaben der Länge n.

Satz 7.6. Für jedes $f \in \mathrm{FP}$ gibt es einen Polynomialzeit-Algorithmus A, der bei Eingabe 0^n einen Schaltkreis C_n mit folgenden Eigenschaften liefert:

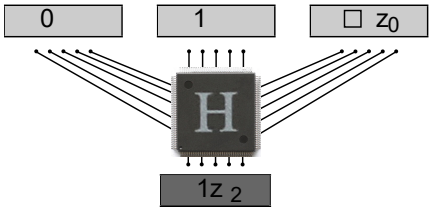
- C_n besitzt genau n Eingänge.
- C_n besteht aus AND- und NOT-Gattern.
- Für alle $x < 2^n$ gilt: Legt man x in Binärdarstellung an den Eingängen an, so liefern die Ausgänge den Wert $f(x)$ in Binärdarstellung.

Beispiel eines solchen Schaltkreises mit Eingängen I_1, I_2 und Ausgängen O_1, O_2:



Beweisskizze Die Berechnung einer deterministischen Polynomialzeit-Turing-Maschine auf Eingaben der Länge n lässt sich als Tabelle darstellen, wobei in Zeile i der Inhalt der Speicherzellen nach dem i-ten Takt steht.

2	1 z_0	0	1	2	2	2
2	1	0 z_0	1	2	2	2
2	1	0	1 z_0	2	2	2
2	1	0	1	2 z_0	2	2
2	1	0	1 z_2	2	2	2
2	1	0 z_3	2	2	2	2
2	1 z_3	2	2	2	2	2
2 z_3	2	2	2	2	2	2
1 z_1	2	2	2	2	2	2



Der Inhalt jeder Zelle der Tabelle ist duch die Inhalte der 3 darüberliegenden, benachbarten Zellen eindeutig festgelegt.

Damit kann jede Zelle aus den 3 darüberliegenden Zellen durch einen Schaltkreis H konstanter Größe mit AND- und NOT-Gattern berechnet werden. Setzt man H an jeder Stelle der Tabelle ein, so ergibt sich der gewünschte Schaltkreis C_n.

Dieser Beweis wird in der Vorlesung Komplexitätstheorie weiter präzisiert. □

Ausgangspunkte der Secure Circuit Evaluation:

- Bob besitzt die Daten x in Binärdarstellung $b_1 \cdots b_m$ (die b_i sind Bits).
- Alice besitzt einen Polynomialzeit-Algorithmus für die Funktion $f \in \mathrm{FP}$ und sie kann den Schaltkreis C_m konstruieren. (C_m leistet Funktionsberechnung für Eingaben der Länge m)

Nun definieren wir das Protokoll zur Berechnung von $f(x)$.

1. Bob wählt verschiedene Primzahlen $p, q \in \mathbb{P}$ mit $p \equiv q \equiv 3 \bmod 4$ und veröffentlicht $n := pq$.

2. Bob verschlüsselt seine Bits mit Zufallszahlen $k_1, \ldots, k_m \in \mathbb{Z}_n^*$ zu $E(b_i, k_i)$ und sendet diese Werte an Alice.

3. Alice simuliert die Rechnung des Schaltkreises C_m, indem sie den Wahrheitswert jedes Knoten in verschlüsselter Form berechnet:
   - die Werte der Eingänge sind die von Bob gesendeten $E(b_i, k_i)$
   - für NOT-Gatter berechnet Alice aus $E(b, k)$ einen Wert $E(\neg b, k')$ wie folgt:
     - für zufälliges $r \in \mathbb{Z}_n^*$ berechnet sie

$$(-1) \cdot r^2 \cdot E(b,k) \bmod n = E(\neg b, r \cdot k)$$
$$= \begin{cases} \text{quadratischer Rest modulo } n, & \text{falls } \neg b = 0 \\ \text{quadratischer Nichtrest modulo } n, & \text{sonst} \end{cases}$$

     - r dient hier der Verschleierung, sodass Bob nicht einzelne Bits anhand von k im Schaltkreis verfolgen kann und damit den Schaltkreis rekonstruieren kann
   - für AND-Gatter berechnet Alice aus $E(b, k)$ und $E(b', k')$ mit Bobs Hilfe einen Wert $E(b \wedge b', k'')$ wie folgt:
     - um Bob keine Information zu liefern, muss Alice die Identitäten *und* die Werte der Bits verschleiern
     - sie wählt zufällige $c, c' \in \{0, 1\}$ und zufällige $r, r' \in \mathbb{Z}_n^*$ und sendet die folgenden beiden Werte an Bob:

$$(-1)^c \cdot r^2 \cdot E(b,k) \equiv E(b \oplus c, k \cdot r) \bmod n$$
$$(-1)^{c'} \cdot (r')^2 \cdot E(b',k') \equiv E(b' \oplus c', k' \cdot r') \bmod n$$

     - Bob entschlüsselt die Werte und kennt nun $b \oplus c$ und $b' \oplus c'$
     - Bob wählt zufällige $k_1, \ldots, k_4 \in \mathbb{Z}_n^*$ und sendet folgende Werte in festgelegter Reihenfolge an Alice:

$$\begin{array}{ll} e_1 = E((b \oplus c) \wedge (b' \oplus c'), k_1) & \text{(richtiger Wert, falls } c = 0 \text{ und } c' = 0) \\ e_2 = E((b \oplus c) \wedge \neg(b' \oplus c'), k_2) & \text{(richtiger Wert, falls } c = 0 \text{ und } c' = 1) \\ e_3 = E(\neg(b \oplus c) \wedge (b' \oplus c'), k_3) & \text{(richtiger Wert, falls } c = 1 \text{ und } c' = 0) \\ e_4 = E(\neg(b \oplus c) \wedge \neg(b' \oplus c'), k_4) & \text{(richtiger Wert, falls } c = 1 \text{ und } c' = 1) \end{array}$$

     Alice wählt in Abhängigkeit von c und c' den passenden Wert $e_i = E(b \wedge b', k_i)$ und verschleiert ihn durch ein zufälliges $r'' \in \mathbb{Z}_n^*$ zu

$$e = e_i \cdot r''^2 \bmod n = E(b \wedge b', r'' \cdot k_i)$$

     (e ist genau dann quadratischer Rest mod n, wenn $b \wedge b' = 0$)

4. Am Ende schickt Alice die verschlüsselten Bits der Ausgänge an Bob, der sie entschlüsselt und so das Ergebnis $f(x)$ erhält.

7.4 Homomorphe Verschlüsselung

Mit der Secure Circuit Evaluation kann Alice beliebige Berechnungen auf Daten durchführen, die von Bob verschlüsselt wurden. Jedoch benötigt sie dabei für die Berechnung der AND-Gatter

Bobs Hilfe.

Vollständig homomorphe Verschlüsselung macht Bobs Hilfe überflüssig. Ziel homomorpher Verschlüsselung ist es, auf verschlüsselten Daten beliebige Funktionsberechnungen f durchzuführen.

$$\mathcal{E}(e,m) \xrightarrow{F} \mathcal{E}(e,f(m)).$$

D. h. $\mathcal{E}$ ist ein Homomorphismus in folgendem Sinne:

$$\mathcal{D}(d,\mathcal{E}(e,f(m))) = \mathcal{D}(d,F(\mathcal{E}(e,m))).$$

Definition 7.7. Ein *homomorphes Kryptosystem* ist ein 6-Tupel $\mathcal{H} = (\Sigma,\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{F},\text{Evaluate})$ mit:

1. $(\Sigma,\mathcal{K},\mathcal{E},\mathcal{D})$ ist ein Kryptosystem.
2. $\mathcal{F}$ ist eine Menge von Funktionen $\{0,1\}^n \to \{0,1\}^n$ (die vom Kryptosystem beherrschten Funktionen).
3. Evaluate ist ein probabilistischer Polynomialzeitalgorithmus, der bei Eingabe eines öffentlichen Schlüssels e, einer Funktion $f \in \mathcal{F}$ und eines Chiffretextes c von m einen Chiffretext c' von $f(m)$ mit $|c'| = |c|$ berechnet.

Bemerkung 7.8.

- Die Forderung $|c| = |c'|$ stellt sicher, dass die durchzuführende Berechnung nicht nur im Chiffretext vermerkt und erst bei der Entschlüsselung durchgeführt wird.
- Die Funktion f wird dem Algorithmus Evaluate in Form eines Schaltkreises mit AND- und NOT-Gattern übergeben.

Definition 7.9. Das homomorphes Kryptosystem $\mathcal{H} = (\Sigma,\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{F},\text{Evaluate})$ heißt *vollständig homomorphes Kryptosystem*, falls $\mathcal{F}$ alle Funktionen $\{0,1\}^n \to \{0,1\}^n$ enthält.

Im Jahr 2009 konstruierte Craig Gentry das erste vollständig homomorphe Kryptosystem und löste damit eine seit 1978 offene Frage. Dies zeigt, dass (wenigstens aus theoretischer Sicht) Cloud-Computing mit Geheimhaltung von Daten vereinbar ist.

Teilweise homomorphes Private-Key-Kryptosystem $\mathcal{H}$

Sei λ der Sicherheitsparameter, er bestimmt die Schlüssellänge. Das Kryptosystem verschlüsselt einzelne Bits $b \in \{0,1\}$. Wir verwenden Zahlen folgender Längen N, P und Q:

- $N := \lambda$ (kurze Zahlen)
- $P := \lambda^2$ (mittllange Zahlen)
- $Q := \lambda^5$ (lange Zahlen)

$\mathcal{K}(1^\lambda)$ liefert (p,p) für eine zufällige, ungerade Zahl $p \in \{0,1\}^P$.
$\mathcal{E}(p,b)$ liefert $pq + m$ für zufällige $q \in \{0,1\}^Q$ und $m \in \{0,1\}^N$ mit $m \equiv b \bmod 2$.
$\mathcal{D}(p,c) := (c \bmod p) \bmod 2$
$\Rightarrow$ Chiffretext = Vielfaches von p + „Rauschen" mit Parität b

Zur Sicherheit von $\mathcal{H}$ sei bemerkt, dass man zum Bestimmen von p aus $c_1 = p \cdot q_1 + m_1$ und $c_2 = p \cdot q_2 + m_2$ eine Art „nährungsweises Größter-Gemeinsamer-Teiler-Problem" lösen muss (sodass man diese Frage beantworten kann: „Gibt es im Bereich von c_1 und c_2 jeweils eine Zahl, sodass beide einen gemeinsamen Teiler haben?"). Hierfür ist kein effizienter Algorithmus bekannt.

Berechnungen auf verschlüsselten Daten

Sei c' ein zuvor veröffentlichter Chiffretext einer 1.

1. spezielle Funktionen:

$$\begin{aligned}\text{Evaluate}(\text{AND}, c_1, c_2) &:= c_1 \cdot c_2\\ \text{Evaluate}(\text{NOT}, c_1, c_2) &:= c + c'\end{aligned}$$

2. allgemeine Funktionen:

$$\text{Evaluate}(f, c_1, \ldots, c_t) := C(c_1, \ldots, c_t),$$

wobei C ein Schaltkreis mit AND- und NOT-Gattern ist, der f berechnet. Er wird schrittweise mit Evaluate(AND, $\cdot$, $\cdot$) und Evaluate(OR, $\cdot$, $\cdot$) ausgewertet.

Berechnung von AND

Seien c_1, c_2 die Chiffretexte der Bits $b_1, b_2 \in \{0, 1\}$.

$$\begin{aligned}c_1 &= pq_1 + m_1 \text{ mit } m_1 \equiv b_1 \bmod 2, |m_1| = N, |q_1| = Q\\ c_2 &= pq_2 + m_2 \text{ mit } m_2 \equiv b_2 \bmod 2, |m_2| = N, |q_2| = Q\end{aligned}$$

$$\begin{aligned}c &:= \text{Evaluate}(\text{AND}, c_1, c_2)\\ &= c_1 \cdot c_2\\ &= p \cdot (pq_1q_2 + q_1m_2 + q_2m_1) + m_1 \cdot m_2\end{aligned}$$

Solange $m_1 \cdot m_2 < p$ (d. h. kein hohes Rauschen) wird AND korrekt berechnet:

$$\begin{aligned}\mathcal{D}(p, c) &= (c \bmod p) \bmod 2\\ &= m_1 \cdot m_2 \bmod 2\\ &= b_1 \cdot b_2 = \text{AND}(b_1, b_2)\end{aligned}$$

Berechnung von NOT

Sei c der Chiffretext eines Bits $b \in \{0, 1\}$.

$$\begin{aligned}c &= pq + m \text{ mit } m \equiv b \bmod 2, |m| = N, |q| = Q\\ c' &= pq' + m' \text{ mit } m' \equiv 1 \bmod 2, |m'| = N, |q'| = Q\end{aligned}$$

$$\begin{aligned}c &:= \text{Evaluate}(\text{NOT}, c)\\ &= c + c'\\ &p(q + q') + m + m'\end{aligned}$$

Solange $m + m' < p$ (d. h. kein hohes Rauschen) wird NOT korrekt berechnet:

$$\begin{aligned}\mathcal{D}(p,c) &= (c \bmod p) \bmod 2\\ &= m + m' \bmod 2\\ &= b + 1 \bmod 2 = \mathrm{NOT}(b)\end{aligned}$$

Homomorphie-Eigenschaften von $\mathcal{H}$

Wie homomorph ist das Kryptosystem $\mathcal{H}$?

- Ein frisch verschlüsseltes Bit hat ein Rauschen von N Bit.
- Hat c ein Rauschen von k Bit, so hat $\mathrm{NOT}(c)$ ein Rauschen von höchstens $k+1$ Bit.
- Haben c_1, c_2 ein Rauschen von k_1, k_2 Bit, so hat $\mathrm{AND}(c_1, c_2)$ ein Rauschen von höchstens $k_1 + k_2$ Bit.

Durch wiederholte Berechnung von AND entsteht schnell ein großes Rauschen, das eine korrekte Entschlüsselung verhindert.

Unser Kryptosystem beherrscht also Funktionen, die sich durch Schaltkreise mit einigen NOT-Gattern und wenigen AND-Gattern berechnen lassen.

Weiterentwicklung zum vollständig homomorphem Kryptosystem

$\mathcal{H}$ lässt sich so modifizieren, sodass es ein vollständig homomorphes Kryptosystem wird; dies wollen wir hier allerdings nur skizzieren. Hierzu benötigen wir, dass $\mathcal{H}$ die eigene Entschlüsselungsfunktion $\mathcal{D}(\,\cdot\,,\,\cdot\,)$ beherrscht. Diese lässt sich dann mit einigen NOT und wenigen AND realisieren.

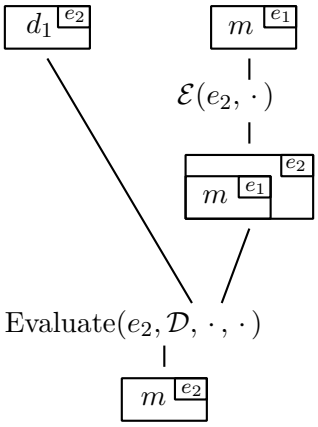
Dadurch kann man den Schlüssel von Ciffretexten wechseln, ohne den Chiffretext zu entschlüsseln. Der Vorteil des Schlüsselwechsels besteht darin, dass sich damit das Rauschen eines Chiffretextes auf ein mittleres Niveau absenken lässt.

Auswertung beliebiger Schaltkreise mit AND- und NOT-Gattern:

- solange Rauschen der Chiffretexte c niedrig oder mittel:
  Berechnung von AND und NOT mittels $\cdot$ und $+$
- falls Rauschen von c hoch:
  Schlüsselwechsel liefert gleichwertiges c' mit mittlerem Rauschen

Schema für Schlüsselwechsel bei Public-Key-Verfahren

- (e_1, d_1) und (e_2, d_2) seien Schlüsselpaare
- $\boxed{m\,\boxed{e}}$ bezeichne eine mit e verschlüsselte Nachricht m.



Bemerkung 7.10.

1. Die Anwendung von Evaluate entfernt das hohe Rauschen der e_1-Verschlüsselung, denn ein Entschlüsseln entfernt das Rauschen vollständig.

2. Das Ausführen von Evaluate erhöht aber das bisher niedrige Rauschen der e_2-Verschlüsselung auf einen mittleren Wert. Er ist genügend klein, um noch mindestens ein Gatter berechnen zu können, ohne hohes Rauschen zu erreichen.

3. Für den Schlüsselwechsel muss eine Liste mit verschlüsselten Entschlüsselungsschlüsseln $\mathcal{E}(e_2, d_1), \mathcal{E}(e_3, d_2), \ldots$ bereitgestellt werden.
   Statt der Liste genügt ein Wert $\mathcal{E}(e, d)$, sofern das Kryptosystem folgende Eigenschaft hat: Für Schlüsselpaare (e, d) lässt sich d nicht aus $\mathcal{E}(e, d)$ bestimmen (circular-security).

Kapitel 8

Miller-Rabin-Primzahltest

Im letzten Kapitel behandeln wir den am häufigsten verwendeten Primzahltest und beweisen seine Korrektheit.

Zunächst definieren wir die Komplexitätsklasse der in randomisierter Polynomialzeit entscheidbaren Mengen.

Definition 8.1. (randomisierte Polynomialzeit) RP sei die Menge aller $A \subseteq \mathbb{N}$, für die es einen probabilistischen Polynomialzeitalgorithmus M gibt, sodass für alle $x \in \mathbb{N}$ gilt:

$$x \in A \Longrightarrow P(M(x) = 1) \geq 1/2$$
$$x \notin A \Longrightarrow P(M(x) = 1) = 0$$

Weiter sei coRP $:= \{A \subseteq \mathbb{N} \mid \overline{A} \in \text{RP}\}$.

Wir zeigen, dass der Miller-Rabin-Primzahltest ein probabilistischer Polynomialzeitalgorithmus M mit folgender Eigenschaft ist:

$$x \in \mathbb{P} \Longrightarrow P(M(x) = 1) = 1$$
$$x \notin \mathbb{P} \Longrightarrow P(M(x) = 1) \leq 1/2$$

Daraus folgt $\mathbb{P} \in$ coRP.

Aus dem kleinen Fermatschen Satz folgt:

$$\exists a\,[1 \leq a < n \text{ und } a^{n-1} \not\equiv 1 \bmod n] \quad \Longrightarrow \quad n \notin \mathbb{P}$$

Ein $a \in \mathbb{Z}_n^*$ mit $a^{n-1} \not\equiv 1 \bmod n$ heißt *Fermat-Zeuge für* n (denn a belegt, dass n nicht prim ist).

Ist n eine zusammengesetzte Zahl und $a \in \mathbb{Z}_n^*$ mit $a^{n-1} \equiv 1 \bmod n$, so nennen wir a einen *Fermat-Lügner für* n (denn a erweckt den falschen Eindruck, dass n prim sei).

Definition 8.2. Ein ungerades, zusammengesetztes $n \in \mathbb{N}$ heißt *Carmichael-Zahl*, falls $a^{n-1} \equiv 1 \bmod n$ für alle $a \in \mathbb{Z}_n^*$.

Es gibt unendlich viele Carmichael-Zahlen, die kleinste ist $3{\cdot}11{\cdot}17 = 561$. Carmichael-Zahlen sind zusammengesetzt, aber besitzen keine Fermat-Zeugen. Sie erfordern eine spezielle Behandlung bei unserem Primzahltest.

Lemma 8.3. Für jede Carmichael-Zahl n gilt:

1. Falls $p \in \mathbb{P}$ mit $p|n$, so $p - 1|n - 1$.
2. Es gibt kein $p \in \mathbb{P}$ mit $p^2|n$ (d. h. n ist quadratfrei).
3. $n = p_1 \cdots p_m$ für $m \geq 3$ und paarweise verschiedene $p_1, \ldots, p_m \in \mathbb{P} - \{2\}$.

Beweis

1. + 2.: Sei $n = p^\alpha \cdot d$ mit $\alpha \in \mathbb{N}^+$ und $p \nmid d$.
Sei $d = q_1^{\beta_1} \cdots q_k^{\beta_k}$ für paarweise verschiedene $q_1, \ldots, q_k \in \mathbb{P}$.
$\mathbb{Z}_{p^\alpha}^*$ ist zyklisch nach dem Satz 4.19, denn $p \in \mathbb{P} - \{2\}$.
Sei γ ein Erzeuger von $\mathbb{Z}_{p^\alpha}^*$.
$\overset{\text{chin. Restsatz}}{\Longrightarrow}$ es existiert ein $\gamma' \in \{0, \ldots, n-1\}$ mit

$$\gamma' \equiv \gamma \pmod{p^\alpha} \text{ und } \gamma' \equiv 1 \pmod{q_i^{\beta_i}} \text{ für } i = 1, \ldots, k$$

$p \nmid \gamma'$, weil sonst $p | \gamma' + j \cdot p^\alpha = \gamma$ ↯ zu $\gamma \in \mathbb{Z}_{p^\alpha}^*$
$q_i \nmid \gamma'$, weil sonst $q_i | \gamma' + j \cdot \acute{} q_i^{\beta_i} = 1$ ↯
$\Rightarrow \text{ggT}(\gamma', n) = 1$, also $\gamma' \in \mathbb{Z}_n^*$
$\text{ord}_n(\gamma') = |\mathbb{Z}_{p^\alpha}^*| = \varphi(p^\alpha) = p^{\alpha-1} \cdot (p-1)$, dies folgt aus dem chinesischem Restsatz, denn:

$$\gamma'^j \equiv \gamma^j \pmod{p^\alpha} \text{ und } \gamma'^j \equiv 1^j \pmod{q_i^{\beta_i}},$$

d. h. bei $j = |\mathbb{Z}_{p^\alpha}^*|$ gilt erstmals $\gamma^j \equiv 1 \pmod{p^\alpha}$ und damit erstmals $\gamma'^j \equiv 1 \pmod{n}$.

$$\Rightarrow \gamma'^{p^{\alpha-1}(p-1)} \equiv 1 \pmod{n} \qquad (*)$$

Da n eine Carmichael-Zahl ist und $\gamma' \in \mathbb{Z}_n^*$, gilt:

$$\gamma'^{n-1} \equiv 1 \pmod{n}$$
$$\overset{(*)}{\Rightarrow} \gamma'^{n-1 \bmod p^{\alpha-1}(p-1)} \equiv 1 \pmod{n}$$

Der Exponent muss 0 sein, weil sonst
$\text{ord}_n(\gamma') < p^{\alpha-1}(p-1)$ ↯

$\Rightarrow p^{\alpha-1}(p-1) | n-1$ also $p-1 | n-1$ (dies zeigt 1.)

Angenommen $\alpha \geq 2$, dann gilt $p | n-1$ ↯ zu $p | n$.
Also $\alpha = 1$ (dies zeigt 2.).

3.: Bisher: $n = p_1 \cdots p_m$ für paarweise verschiedene $p_1, \ldots, p_m \in \mathbb{P} - \{2\}$, denn Carmichael-Zahlen sind nach Definition ungerade
angenommen $m = 2$, d. h. $n = p \cdot q$ für $p, q \in \mathbb{P} - \{2\}$ mit $p > q$
$\overset{(1.)}{\Rightarrow} p - 1 | n - 1 = (p-1)q + q - 1$
$\Rightarrow p - 1 | q - 1$ ↯ zu $p > q$

□

Definition 8.4. Ein Element $w \in \mathbb{Z}_n$ heißt *Quadratwurzel von 1 modulo n*, falls $w^2 \equiv 1 \bmod n$.

Lemma 8.5. Für jedes $p \in \mathbb{P}$ gibt es genau zwei Quadratwurzeln von 1 modulo p, nämlich 1 und $p - 1$.

Beweis Sei $w \in \mathbb{Z}_p$ mit $w^2 \equiv 1 \bmod p$. Also $(w+1)(w-1) \equiv 0 \bmod p$. Damit gilt $p | w+1$ oder $p | w-1$. D. h. $w \equiv -1 \bmod p$ oder $w \equiv 1 \bmod p$. □

Wir verallgemeinern den Begriff des Fermat-Zeugen und definieren Miller-Rabin-Zeugen.

Definition 8.6. Sei $n \geq 3$ eine ungerade Zahl und sei $a \in \mathbb{Z}_n^*$. Wähle $m, k \in \mathbb{N}^+$ so, dass $n - 1 = m \cdot 2^k$ und m ungerade.

- a heißt *Miller-Rabin-Zeuge für* $n :\Longleftrightarrow a^m \not\equiv 1 \bmod n$ und für alle $j \in \{0, \ldots, k-1\}$ gilt $a^{2^j \cdot m} \not\equiv -1 \bmod n$.
- a heißt *Miller-Rabin-Lügner für* $n :\Longleftrightarrow n$ ist zusammengesetzt und a ist kein Miller-Rabin-Zeuge für n.

Miller-Rabin-Zeugen für ungerade $n \geq 3$ belegen, dass n nicht prim ist.

Lemma 8.7. Sei $n \geq 3$ ungerade. Falls es einen Miller-Rabin-Zeugen für n gibt, so ist n zusammengesetzt.

Beweis Angenommen, a ist Miller–Rabin-Zeuge für ein $n \in \mathbb{P}$.
$\overset{\text{kleiner Fermat}}{\Longrightarrow} a^{n-1} \equiv 1 \pmod{a}$, wobei $a^{n-1} = a^{m \cdot 2^k}$ mit $m, k \in \mathbb{N}^+$ wie in Definition 8.6
$\Rightarrow a^{m \cdot 2^{k-1}}$ ist Quadratwurzel von 1 modulo n
$\overset{\text{Lemma 8.5}}{\Longrightarrow} a^{m \cdot 2^{k-1}} \equiv 1 \pmod{n}$ oder $a^{m \cdot 2^{k-1}} \equiv -1 \pmod{n}$, was aber nach Definition des Miller-Rabin-Zeugen nicht möglich ist.

$$\begin{aligned} &\Rightarrow a^{m \cdot 2^{k-1}} \equiv 1 \pmod{n} \\ &\quad \vdots \\ &\Rightarrow a^m \equiv 1 \pmod{n} \qquad \lightning \text{ zur Definition des Miller-Rabin-Zeugen} \end{aligned}$$

□

Lemma 8.8. Ist $(G, *)$ eine endliche Gruppe mit neutralem Element e und ist $H \subseteq G$ unter $*$ abgeschlossen mit $e \in H$, dann ist $(H, *)$ eine Untergruppe von $(G, *)$.

Beweis Es genügt zu zeigen:

$$\forall x \in H \, \exists! y \in H [x * y = y * x = e].$$

Für $x \in H$ sei $\sigma_x : H \to H$ mit $\sigma_x(y) := x * y$.
Die σ_x sind injektiv, denn aus $\sigma_x(y_1) = \sigma_x(y_2)$ folgt $x * y_1 = x * y_2$ und damit $y_1 = y_2$, da $(G, *)$ eine Gruppe ist.
Da G endlich, sind die σ_x bijektiv. D. h. $\forall x \in H \, \exists! y \in H [x * y = e]$.
Aus $x * y = e$ folgt $y * x * y = y$ und damit $y * x = e$.
Dies zeigt $\forall x \in H \, \exists! y \in H [x * y = y * x = e]$. □

Der Miller-Rabin-Primzahltest ist folgender probabilistischer Polynomialzeit-Algorithmus.

MillerRabin(n) für ungerade $n \geq 3$:

1. bestimme $m, k \in \mathbb{N}^+$ mit $n - 1 = m \cdot 2^k$ und m ungerade
2. wähle gleichverteiltes, zufälliges $a \in \{1, \ldots, n-1\}$
3. $x := a^m \bmod n$
4. if $(x \equiv 1 \bmod n)$ return 1
5. for $j = 0$ to $k - 1$
6. if $(x \equiv -1 \bmod n)$ return 1
7. $x := (x^2 \bmod n)$
8. return 0

Satz 8.9. Für alle $n \in \mathbb{P} - \{2\}$ gilt $P(\text{MillerRabin}(n) = 1) = 1$.

Beweis Angenommen, es gilt $P(\text{MillerRabin}(n) = 1) < 1$ für ein $n \in \mathbb{P} - \{2\}$.
D. h. auf mindestens einem Rechenweg stoppt MillerRabin(n) mit 0.
Dieser Rechenweg muss Zeile 8 erreichen. Also gilt:

- $a^m \not\equiv 1 \bmod n$, wegen Zeilen 3+4
- $\forall j \in \{0, \ldots, k-1\}, a^{2^j \cdot m} \not\equiv -1 \bmod n$, wegen Zeile 6

Aus $n \in \mathbb{P}$ folgt $a \in \mathbb{Z}_n^*$. Also ist a ein Miller-Rabin-Zeuge für n.
Aus Lemma 8.7 folgt $n \notin \mathbb{P}$, ein Widerspruch. □

Den Beweis der folgenden Aussage zerlegen wir in mehrere Teile:
„Für ungerade $n \geq 3$ mit $n \notin \mathbb{P}$ gilt $P(\text{MillerRabin}(n) = 1) \leq 1/2$."

Definition 8.10. Für zusammengesetzte Zahlen n seien die Mengen der Miller-Rabin-Lügner bzw. Fermat-Lügner wie folgt definiert.

$$\text{MRL}_n := \{a \in \mathbb{N} \mid 1 \leq a < n \text{ und } a \text{ ist Miller-Rabin-Lügner für } n\}$$
$$\text{FL}_n := \{a \in \mathbb{N} \mid 1 \leq a < n \text{ und } a \text{ ist Fermat-Lügner für } n\}$$

Eigenschaft 8.11. Für zusammengesetzte n gilt $\text{MRL}_n \subseteq \text{FL}_n \subseteq \mathbb{Z}_n^*$.

1. Inklusion: 1.Fall: $a^m \equiv 1 \pmod n$
$\Rightarrow a^{n-1} = (a^m)^{2^k} \equiv 1 \pmod n$
2. Fall: $\exists j \in \{0, \ldots, k-1\}, a^{2^j \cdot m} \equiv -1 \pmod n$
$\Rightarrow a^{2^{j+1} \cdot m} \equiv 1 \pmod n$
$\Rightarrow a^{n-1} \equiv 1 \pmod n$

2. Inklusion: klar, denn alle Fermat-Lügner sind aus $\mathbb{Z}_n^*$

Lemma 8.12. Sei $n \geq 3$ ungerade mit $n \notin \mathbb{P}$. Falls n keine Carmichael-Zahl ist, so gilt $P(\text{MillerRabin}(n) = 1) \leq 1/2$.

Beweis Da n keine Carmichael-Zahl ist, existiert ein $a \in \mathbb{Z}_n^*$ mit $a^{n-1} \not\equiv 1 \bmod n$.
$\overset{8.11}{\Longrightarrow} \text{MRL}_n \subseteq \text{FL}_n \subsetneq \mathbb{Z}_n^*$; wir zeigen, dass FL_n eine Untergruppe von $\mathbb{Z}_n^*$ ist
nach 8.8 g. z. z.: FL_n enthält 1 und ist abgeschlossen unter Multiplikation mod n

- klar: $1 \in \text{FL}_n$
- seien $a, b \in \text{FL}_n$
  $\Rightarrow (a \cdot b)^{n-1} \equiv a^{n-1} \cdot b^{n-1} \equiv 1 \pmod n$
  $\Rightarrow a \cdot b \in \text{FL}_n$

$\Rightarrow \text{FL}_n$ ist eine echte Untergruppe von $\mathbb{Z}_n^*$
Ordnung einer Untergruppe teilt die Gruppenordnung (Satz von Lagrange, 4.9)
$\Rightarrow |\text{FL}_n|$ ist echter Teiler von $|\mathbb{Z}_n^*|$
$\Rightarrow |\text{FL}_n| \leq \frac{|\mathbb{Z}_n^*|}{2} \leq \frac{n-1}{2}$
$\Rightarrow |\text{MRL}_n| \leq \frac{n-1}{2}$
$\Rightarrow$ in Zeile 2 von MillerRabin(n) wählen wir mit Wahrscheinlichkeit $\leq 1/2$ ein $a \in \text{MRL}_n$
mit der gleichen Wahrscheinlichkeit wird 1 ausgegeben, denn:

- a Miller-Rabin-Lügner $\rightarrow$ Ausgabe 1
- a Miller-Rabin-Zeuge $\rightarrow$ Ausgabe 0

- $a \notin \mathbb{Z}_n^* \rightarrow$ Ausgabe 0 (weil sonst $a^{n-1} \equiv 1 \pmod{n}$, also $a \in \mathbb{Z}_n^*$ ↯)

□

Wir müssen nun noch folgende Aussage zeigen:

„Ist n eine Carmichael-Zahl, so gilt $P(\text{MillerRabin}(n) = 1) \leq 1/2$."

Dafür brauchen wir ein Argument, das zeigt, dass $|\text{MRL}_n|$ höchstens halb so groß wie $|\mathbb{Z}_n^*|$ ist.

Die Argumentation über die Untergruppe FL_n (wie in Lemma 8.12) funktioniert hier nicht, denn nach Eigenschaft 8.11 und Definition 8.2 gilt für jede Carmichael-Zahl n:

$$\text{MRL}_n \subseteq \text{FL}_n = \mathbb{Z}_n^*.$$

Wir können aber auch nicht direkt mit MRL_n argumentieren, da MRL_n nicht für jedes n eine Gruppe ist.

Wir suchen eine echte Untergruppe von $\mathbb{Z}_n^*$, die MRL_n enthält.

Sei n eine Carmichael-Zahl und seien $m, k \in \mathbb{N}^+$ so gewählt, dass $n - 1 = m \cdot 2^k$ und m ungerade.

$$z(n) := \max\{j \in \{0, \ldots, k\} \mid \exists a \in \{1, \ldots, n-1\}, a^{2^j \cdot m} \equiv -1 \bmod n\}$$

Das Maximum existiert, denn $(n-1)^m \equiv (-1)^m \equiv -1 \bmod n$, weil m ungerade. Also gehört 0 zur obigen Menge.

Weiter gilt $z(n) \neq k$, denn andernfalls existiert ein $a \in \{1, \ldots, n-1\}$ mit $a^{n-1} \equiv a^{2^k \cdot m} \equiv 1 \bmod n$, also $a \in \mathbb{Z}_n^*$ und $a^{n-1} \equiv -1 \bmod n$, ein Widerspruch zur Annahme, dass n eine Carmichael-Zahl ist. Also gilt $0 \leq z(n) \leq k - 1$.

Wir zeigen, dass folgende Menge eine echte Untergruppe von $\mathbb{Z}_n^*$ ist, die MRL_n enthält.

$$\text{MRL}'_n := \{a \in \{1, \ldots, n-1\} \mid a^{2^{z(n)} \cdot m} \equiv \pm 1 \bmod n\}$$

Lemma 8.13. Für jede Carmichael-Zahl n gilt $\text{MRL}_n \subseteq \text{MRL}'_n$.

Beweis Sei $a \in \text{MRL}_n$. □

1. Fall: $a^m \equiv 1 \pmod{n}$, so $a^{m \cdot 2^{z(n)}} \equiv 1 \pmod{n}$, also $a \in \text{MRL}'_n$

2. Fall: es existiert ein $j \in \{0, \ldots, k-1\}$ mit $a^{2^j \cdot m} \equiv -1 \pmod{n}$
   $\Rightarrow j \leq z(n)$ nach Definition von $z(n)$
   a) falls $j = z(n)$, so $a \in \text{MRL}'_n$
   b) falls $j < z(n)$, so gilt

$$a^{m \cdot 2^{z(n)}} \equiv a^{m \cdot 2^j \cdot 2^{\overbrace{z(n) - j}^{>0}}} \equiv 1 \pmod{n}$$

$\Rightarrow a \in \text{MRL}'_n$

Lemma 8.14. Für jede Carmichael-Zahl n ist MRL'_n eine Untergruppe von $\mathbb{Z}_n^*$.

Beweis $1 \in \text{MRL}'_n$
$\overset{8.8}{\Longrightarrow}$ n. z. z. MRL'_n abgeschlossen unter Multiplikation modulo n
seien $a, b \in \text{MRL}'_n$

$\Rightarrow a^{m \cdot 2^{z(n)}} \equiv \pm 1 \pmod{n}$
$b^{m \cdot 2^{z(n)}} \equiv \pm 1 \pmod{n}$

$\Rightarrow (a \cdot b)^{m \cdot 2^{z(n)}} \equiv \pm 1 \pmod{n}$

□

Lemma 8.15. Für jede Carmichael-Zahl n ist MRL'_n eine echte Untergruppe von $\mathbb{Z}_n^*$.

Beweis Nach 8.14 genügt die Konstruktion eines $a \in \mathbb{Z}_n^* - \mathrm{MRL}'_n$.
$\overset{8.3.3}{\Longrightarrow} n = n_1 \cdot n_2$, wobei $n_1, n_2 > 1$ ungerade mit $\mathrm{ggT}(n_1, n_2) = 1$
wähle $\hat{a} \in \mathbb{Z}_n^*$ so, dass $\hat{a}^{m \cdot 2^{z(n)}} \equiv -1 \pmod{n}$ (existiert nach Definition von $z(n)$)
$a_1 := \hat{a} \pmod{n_1}$
aus chinesischem Restsatz (4.10) folgt, dass es genau ein $a \in \{0, \ldots, n-1\}$ gibt, sodass gleichzeitig gilt:

$$\begin{aligned} a &\equiv a_1 \pmod{n_1} \\ a &\equiv 1 \pmod{n_2} \end{aligned}$$

Wir zeigen $a \in \mathbb{Z}_n^* - \mathrm{MRL}'_n$. Für teilerfremde n_1, n_2 gilt:

$$a \equiv b \bmod n_1 n_2 \iff a \equiv b \bmod n_1 \text{ und } a \equiv b \bmod n_2 \qquad (*)$$

$$a^{m \cdot 2^{z(n)}} \equiv \hat{a}^{m \cdot 2^{z(n)}} \equiv -1 \pmod{n_1} \qquad (**)$$

$$a^{m \cdot 2^{z(n)}} \equiv 1^{m \cdot 2^{z(n)}} \equiv 1 \pmod{n_2} \qquad (***)$$

$\Rightarrow a^{m \cdot 2^{z(n)}} \not\equiv 1 \pmod{n}$, wegen $(**)$ und $(*)$
$a^{m \cdot 2^{z(n)}} \not\equiv -1 \pmod{n}$, wegen $(***)$ und $(*)$
$\Rightarrow a \notin \mathrm{MRL}'_n$

n. z. z. $a \in \mathbb{Z}_n^*$:
aus $(**)$ und $(***)$ folgt:

$$\begin{aligned} a^{m \cdot 2^{z(n)+1}} &\equiv 1 \pmod{n_1} \\ a^{m \cdot 2^{z(n)+1}} &\equiv 1 \pmod{n_2} \end{aligned}$$

$$\begin{aligned} &\overset{(*)}{\Rightarrow} && a^{m \cdot 2^{z(n)+1}} \equiv 1 \pmod{n} \\ &\Rightarrow && a \cdot a^{m \cdot 2^{z(n)+1}-1} \equiv 1 \pmod{n} \\ & && a \cdot a^{m \cdot 2^{z(n)+1}-1} = 1 + i \cdot n \text{ für ein } i \in \mathbb{Z} \end{aligned}$$

$\Rightarrow$ jeder gemeinsame Teiler von a und n teilt 1
$\Rightarrow \mathrm{ggT}(a, n) = 1$, d. h. $a \in \mathbb{Z}_n^*$ □

Lemma 8.16. Ist n eine Carmichael-Zahl, so gilt

$$P(\mathrm{MillerRabin}(n) = 1) \leq 1/2.$$

Beweis Nach 8.15 ist MRL'_n eine echte Untergruppe von $\mathbb{Z}_n^*$.
Nach dem Satz von Lagrange ist $|\mathrm{MRL}'_n|$ echter Teiler von $|\mathbb{Z}_n^*|$.
Also $|\mathrm{MRL}_n| \leq |\mathrm{MRL}'_n| \leq |\mathbb{Z}_n^*|/2 \leq n-1/2$. Demnach wählen wir in Zeile 2 von MillerRabin(n) mit Wahrscheinlichkeit $\leq 1/2$ ein $a \in \mathrm{MRL}_n$.

Falls MillerRabin(n) die Ausgabe 1 liefert, so ist a kein Miller-Rabin-Zeuge und $a^{n-1} \equiv 1 \bmod n$, also gilt $a \in \mathbb{Z}_n^*$. Dies zeigt: Falls MillerRabin(n) die Ausgabe 1 liefert, so wurde ein $a \in \mathrm{MRL}_n$ gewählt. □

Satz 8.17. MillerRabin ist ein probabilistischer Polynomialzeitalgorithmus mit folgender Eigenschaft für ungerade $n \geq 3$:

$$n \in \mathbb{P} \Longrightarrow P(\mathrm{MillerRabin}(n) = 1) = 1$$
$$n \notin \mathbb{P} \Longrightarrow P(\mathrm{MillerRabin}(n) = 1) \leq 1/2$$

Beweis Die erste Implikation folgt aus Satz 8.9, die zweite aus den Lemmas 8.12 und 8.16.
Es ist leicht zu sehen, dass der Algorithmus in probabilisitischer Polynomialzeit arbeitet. □

Bei k-facher Wiederholung des Algorithmus ergeben sich die Wahrscheinlichkeiten 1 bzw. $\leq 1/2^k$.

Folgerung 8.18. $\mathbb{P} \in \mathrm{coRP}$.

Beweis Aus Satz 8.17 lässt sich leicht ein probabilistischer Polynomialzeitalgorithmus M konstruieren mit folgender Eigenschaft für *alle* $n \in \mathbb{N}$:

$$n \in \mathbb{P} \Longrightarrow P(M(n) = 0) = 0$$
$$n \in \overline{\mathbb{P}} \Longrightarrow P(M(n) = 0) \geq 1/2$$

Vertauschen wir in M die Ausgaben 0 und 1, ergibt dies einen probabilistischen Polynomialzeitalgorithmus mit folgender Eigenschaft für alle $n \in \mathbb{N}$:

$$n \in \mathbb{P} \Longrightarrow P(M(n) = 1) = 0$$
$$n \in \overline{\mathbb{P}} \Longrightarrow P(M(n) = 1) \geq 1/2$$

Dies zeigt $\overline{\mathbb{P}} \in \mathrm{RP}$, also $\mathbb{P} \in \mathrm{coRP}$. □