

Kryptographie und Datensicherheit

Prof. Dr. Christian Glaßer

Stand: 18. September 2023

Inhaltsverzeichnis

1	Grundlegende Begriffe	1
2	Symmetrische Kryptosysteme	11
2.1	Cäsar-Chiffre	12
2.2	Substitutionschiffre	14
2.3	Vigenère-Chiffre	16
2.4	Vernam-Chiffre	19
2.5	Enigma	23
2.6	Algebraische Grundlagen (Teil 1)	29
2.7	AES	34
2.8	Betriebsmodi für Blockchiffren	40

3 Perfekte Sicherheit	45
3.1 Grundlagen der Wahrscheinlichkeitstheorie	46
3.2 Vernam-Chiffre und perfekte Sicherheit	48
4 Asymmetrische Kryptosysteme	54
4.1 Algebraische Grundlagen (Teil 2)	57
4.2 RSA-Kryptosystem	60
4.3 Diffie-Hellman-Schlüsselaustausch	74
4.4 Elgamal-Kryptosystem	85
4.5 Goldwasser-Micali-Kryptosystem	90
4.6 Sicherheit von Public-Key-Kryptosystemen	100

5	Digitale Signatur	103
5.1	Kryptographische Hashfunktionen	105
5.2	RSA-Signatur	109
5.3	Elgamal-Signatur	112
5.4	Lamport-Diffie-Einmal-Signatur	114
6	Identifikation	117
6.1	Einmal-Passwörter	119
6.2	Challenge-Response-Identifikation	120
6.3	Zero-Knowledge-Beweise	127
7	Secure Multiparty Computation	138
7.1	Secret Sharing	140

7.2	Das Millionärsproblem	144
7.3	Secure Circuit Evaluation	147
7.4	Vollständig Homomorphe Verschlüsselung	154
8	Miller-Rabin-Primzahltest	165
Index		181

1 Grundlegende Begriffe

Definition 1.1 (Zahlenbereiche und Wortmengen)

- $\mathbb{N} \stackrel{\text{df}}{=} \{0, 1, \dots\}$ (*Menge der natürlichen Zahlen*)
- $\mathbb{N}^+ \stackrel{\text{df}}{=} \{1, 2, \dots\}$ (*Menge der positiven natürlichen Zahlen*)
- $\mathbb{Z} \stackrel{\text{df}}{=} \{\dots, -2, -1, 0, 1, 2, \dots\}$ (*Menge der ganzen Zahlen*)
- $\mathbb{Z}_n \stackrel{\text{df}}{=} \{0, 1, \dots, n-1\}$ (*Reste modulo n*)
- $\mathbb{Z}_n^i \stackrel{\text{df}}{=} \underbrace{\mathbb{Z}_n \times \mathbb{Z}_n \times \dots \times \mathbb{Z}_n}_{i \text{ Mal}}$
- $\mathbb{P} = \{2, 3, 5, 7, 11, 13, \dots\}$ (*Menge der Primzahlen*)
- $\Sigma^{\leq n} \stackrel{\text{df}}{=} \{w \in \Sigma^* \mid |w| \leq n\}$ (*Wörter mit Länge $\leq n$*)
- *Für $x, y \in \mathbb{Z}$ mit $y \neq 0$ sei $(x \bmod y) \stackrel{\text{df}}{=} x - (y \cdot \lfloor x/y \rfloor)$.*
(Für positive x, y ist dies der Rest der Division x/y .)

Probabilistische Algorithmen

Einen probabilistischen 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 Wahrscheinlichkeit $P(r) = 1/2^n$ realisiert.*

2. Ein probabilistischer Algorithmus M liefert mit folgender Wahrscheinlichkeit bei Eingabe x die Ausgabe y :

$$P(M(x) = y) \stackrel{\text{df}}{=} \sum_{\substack{r: \text{Rechenweg von } M \\ \text{auf } x \text{ mit Ausgabe } y}} P(r)$$

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. Ein probabilistischer Algorithmus M arbeitet in der Zeit $t : \mathbb{N} \rightarrow \mathbb{N}$, falls M auf x nur Rechenwege der Länge $\leq t(|x|)$ besitzt.

Amplifizierung der Erfolgswahrscheinlichkeit:

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

Beispiel 1.3 (Erfolgswahrscheinlichkeit von Algorithmen)

$$(\exists x + 4) = ?$$


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 können Geheimtext nicht entschlüsseln
(Sicherheit)

In der folgenden 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 prob. Polynomialzeitalgorithmus mit Erfolgswkt. $\geq 1/2$, $\mathcal{K}(1^n)$ generiert Schlüsselpaare (e, d) für Klartexte der Länge $\leq n$ (Bezeichnung: **Schlüsselgenerator**, key generator)
3. \mathcal{E} ist ein prob. Polynomialzeitalgorithmus mit Erfolgswkt. $\geq 1/2$, $\mathcal{E}(e, m)$ verschlüsselt den Klartext m mit dem Schlüssel e (Bezeichnung: **Verschlüsselungsalgorithmus**, encryption algor.)
4. \mathcal{D} ist ein deterministischer Polynomialzeitalgorithmus, $\mathcal{D}(d, c)$ entschlüsselt den Chiffretext c mit dem Schlüssel d (Bezeichnung: **Entschlüsselungsalgorithmus**, decryption algor.)
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 Definition 1.4 hängen die Schlüssel nur von der Nachrichtenlänge, aber nicht vom Nachrichteninhalt ab. Diese Annahme ist sinnvoll:

Standardszenario für Kryptosysteme:

A möchte an B über unsicheren Kanal eine Nachricht senden.

Vorher wurde ein Passwort vereinbart/veröffentlicht.

⇒ Bei der Schlüsselerzeugung ist Nachricht noch nicht bekannt.

Benennung der Parteien in kryptographischen Szenarien

- 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

Standardszenario: Alice möchte eine Nachricht an Bob über einen unsicheren Kanal senden

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 (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: d lässt sich nicht effizient aus e berechnen.

Bob veröffentlicht 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 nötig.

Überblick zu Ansätzen der Kryptoanalyse

Annahme: Verschlüsselungsverfahren ist bekannt
(sonst müssen alle gängigen Verfahren getestet werden)

Angriffsszenarien:

- 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)

2 Symmetrische Kryptosysteme (Private-Key-Verfahren)

Ausblick

- Wir behandeln die folgenden klassischen Verfahren:
 - Cäsar-Chiffre
 - Substitutionschiffre
 - Vigenère-Chiffre
 - Vernam-Chiffre
 - Enigma
- Danach lernen wir das AES-Verfahren als Beispiel für ein modernes, weit verbreitetes Kryptosystem kennen.

2.1 Cäsar-Chiffre (um das Jahr -50)

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

Alphabet wird mit $\mathbb{Z}_{26} = \{0, \dots, 25\}$ identifiziert und jeder Buchstabe wird durch den dritt nächsten ersetzt.

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

Beispiel: KLEOPATRA ⇒ NOHRSDWUD

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

Definition 2.1 (Cäsar-Chiffre, Verschiebungsschiffe)

Cäsar $\stackrel{\text{df}}{=} (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit

- $\Sigma = \mathbb{Z}_{26}$
- $\mathcal{K}(1^n)$ liefert gleichverteilt zufälliges 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 Verschiebungsschiffre 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.

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 $\stackrel{df}{=} (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit

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

- Bemerkung 2.4**
1. Etwas sicherer als Cäsar-Chiffre, da $26!$ Permutationen existieren. Daher ist eine Brute-Force-Attacke nicht möglich.
 2. Anfällig gegen statistische Analysen. Häufigkeit deutscher Buchstaben: $E=15\%$, $N=9\%$, ..., $Q=0.014\%$, $X=0.013\%$. Hierdurch ergibt sich sehr schnell die Zuordnung der Buchstaben.

2.3 Vigenère-Chiffre (ca. 1580)

Benannt nach Blaise de Vigenère, französischer Diplomat.

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

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)

↓

↓

SAKAOHXHWEAE: (18,00,10,00,14,07,23,07,22,04,00,04)

Wir definieren die Vigenère-Chiffre für eine feste Passwortlänge k .

Definition 2.5 (Vigenère-Chiffre)

Vigenère_k $\stackrel{\text{df}}{=} (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit

- $\Sigma = \mathbb{Z}_{26}$
- $\mathcal{K}(1^n)$ liefert gleichverteilt zufälliges 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 zur Substitutionschiffre etwas weniger anfällig gegen Häufigkeitsanalysen. Bei guten Passwörtern wenig anfällig gegen Brute-Force-Attacken.
3. Zu einfache Passwörter ermöglichen Wörterbuchattacke.

4. Gleiche Klartextteile haben gleiche Chiffretextteile, wenn der Abstand ein Vielfaches der Passwortlänge ist. Dies liefert Hinweise über die Passwortlänge k .
5. Bei bekannter Passwortlänge k betrachtet man jeden k -ten Buchstaben im Klartext. Die Wahrscheinlichkeiten dieser Buchstaben liefern Wahrscheinlichkeiten für den entsprechenden Buchstaben des Schlüssels.
6. Vigenère-Chiffre 1854 durch Charles Babbage gebrochen.
7. Als **Autokey-Verschlüsselung** bezeichnet man die Variante der Vigenère-Chiffre, bei der das Passwort durch Anhängen des Klartexts verlängert wird:

DASISTGEHEIM: (03,00,18,08,18,19,06,04,07,04,08,12)

+

+

KEYDASISTGEH: (10,04,24,03,00,18,08,18,19,06,04,07)

↓

↓

NEQLSLOWAKMT: (13,04,16,11,18,11,14,22,00,10,12,19)

2.4 Vernam-Chiffre (1918)

Benannt nach Gilbert Vernam, US-amerikanischer Ingenieur.

Das Verfahren ist auch bekannt als **One-Time-Pad** (OTP).

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

ROTESTELEFON: (17,14,19,04,18,19,04,11,04,05,14,13)

+

+

KWTUNRGQFLXZ: (10,22,19,20,13,17,06,16,05,11,23,25)

↓

↓

BKMYFKKBQJQLM: (01,10,12,24,05,10,10,01,09,16,11,12)

Definition 2.7 (Vernam-Chiffre)

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

- $\Sigma = \mathbb{Z}_{26}$
- $\mathcal{K}(1^n)$ liefert gleichverteilt zufälliges Element aus $\{(e, e) \mid e \in \Sigma^n\}$
- $\mathcal{E}(e_1 \cdots e_n, m_1 \cdots m_n) = c_1 \cdots c_n$, 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. Im Kapitel 3 beweisen wir, dass die

Vernam-Chiffre perfekte Sicherheit besitzt.

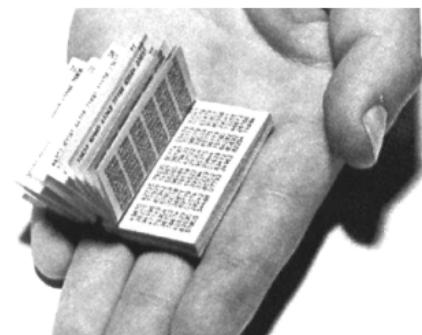
2. Wichtig für Sicherheit der Vernam-Chiffre: gleichverteilt zufälliger Schlüssel.

3. KGB-Spione nutzten OTPs, um Nachrichten mit Papier und Bleistift zu verschlüsseln. OTPs auf Pyropapier, das sehr schnell und 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. Sender fügt dem Signal ein Rauschen hinzu, das beim Empfänger wieder entfernt wird. Das Rauschsignal wurde durch Paare identischer Schellackplatten verteilt.
6. Rotes Telefon (Fernschreibverbindung Moskau-Washington) wird durch OTPs geschützt.
7. 1944-1945 konnten US-Kryptoanalytiker einen OTP des deutschen Außenministeriums brechen. Der Schlüsselgenerator produzierte keine zufälligen Schlüssel.

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.



In einem Kleiderbügel versteckte Codes des MfS.



Ein Miniatur-Codebuch des KGB.

2.5 Enigma

Die im Zweiten Weltkrieg vom deutschen Militär verwendete Chiffriermaschine.

1918 von Arthur Scherbius zunächst für den zivilen Gebrauch entwickelt.

Während des Zweiten Weltkriegs militärisch verwendet und mehrfach weiterentwickelt.

Häufigste Variante: Enigma I



Funktionsprinzip

Maschine besteht aus 3 Walzen (Rotoren),
jede bewirkt Permutation auf Alphabet.
(Bezeichnung: α, β, γ)

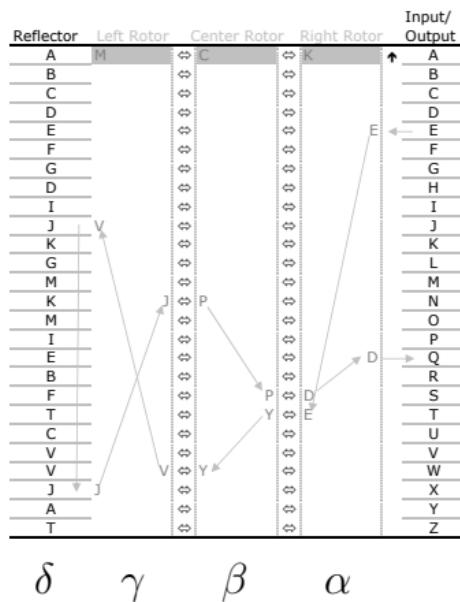
Walzen lassen sich gegeneinander verdrehen, pro Buchstabe werden eine oder mehrere Walzen gedreht. (wie ein Zähler)

Schließlich gibt es eine weitere, feststehende Walze, den sogenannten Reflektor.
(Bezeichnung: δ , Alphabetspermutation)

Verschlüsselung eines Buchstabens p :

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

Informationsfluss durch elektrischen 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
⇒ Enigma liefert fixpunktfreie Abbildung,
d.h. chiffrierter Buchstabe C hat Klartext $\neq C$
- wenn Reflektor A auf U abbildet, so auch U auf A
⇒ Enigma liefert selbstinverse Abbildung,
d.h. zweimalige Verschlüsselung liefert wieder den Klartext

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

⇒ Schwächung der kombinatorischen Komplexität der Maschine

Entzifferung

1932 polnischer Mathematiker Marian Rejewski bricht Enigma I

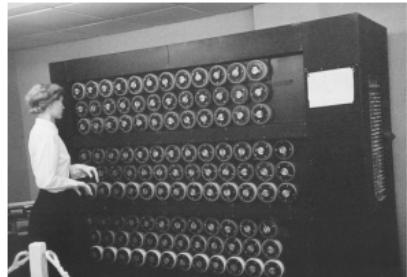
- nutzt legal gekaufte, zivile Variante der Enigma
- bestimmt mit Permutationstheorie die Verdrahtungsreihenfolge von Walzen und Reflektor der militärischen Variante
- bestimmt Walzenlage und Walzenstellung mit spezieller Maschine (Zyklometer)

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

- Bletchley 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 Alliierte entziffern monatlich tausende wichtige geheime Nachrichten der Deutschen.

Auf deutscher Seite bleibt dies unbemerkt.

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

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 in 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 $* : G \times G \rightarrow G$ 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 $K \times K \rightarrow K$
- $(K, +)$ 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 $\mathbb{F}_p \stackrel{\text{df}}{=} (\mathbb{Z}_p, +, \cdot)$, 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 vom Grad n . Der Körper \mathbb{F}_{p^n} ist definiert durch $\mathbb{F}_{p^n} \stackrel{\text{df}}{=} (\mathbb{Z}_p^n, +, \cdot)$, wobei das Produkt $a \cdot b$ und die Summe $a + b$ für $a = (a_{n-1}, \dots, a_0), b = (b_{n-1}, \dots, b_0) \in \mathbb{Z}_p^n$ wie folgt definiert sind:

- betrachte a und b als Polynome aus $\mathbb{F}_p[x]$:

$$a(x) = a_{n-1}x^{n-1} + \dots + a_1x^1 + a_0x^0$$

$$b(x) = b_{n-1}x^{n-1} + \dots + b_1x^1 + b_0x^0$$

- $c(x) \stackrel{\text{df}}{=} (a(x) \cdot b(x)) \bmod r(x)$, $d(x) \stackrel{\text{df}}{=} (a(x) + b(x)) \bmod r(x)$

- $a \cdot b \stackrel{\text{df}}{=} (c_{n-1}, \dots, c_0)$ mit $c(x) = c_{n-1}x^{n-1} + \dots + c_1x^1 + c_0x^0$
 $a + b \stackrel{\text{df}}{=} (d_{n-1}, \dots, d_0)$ mit $d(x) = d_{n-1}x^{n-1} + \dots + d_1x^1 + d_0x^0$

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 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})

$$(-3x+4) = 7x$$


2.7 AES

- 1976 DES (Data Encryption Standard)
- 1990er Jahre 3DES (Triple DES, 3×DES anwenden)
- 2000 AES (Advanced Encryption Standard, auch Rijndael-Algorithmus genannt nach Entwicklern Joan Daemen und Vincent Rijmen)
- AES ist Blockchiffre (Klartextblöcke fester Größe, 16 Bytes)
- Prinzip: Hintereinanderschaltung verschiedener Permutationen
- wird als sicher angesehen, keine relevanten Angriffe bekannt
- Anwendung: WPA2, SSH, Skype, WhatsApp, EFS, ...

Definition 2.15 (AES)

AES $\stackrel{df}{=} (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit

- $\Sigma = \mathbb{Z}_{2^{128}}$
- $\mathcal{K}(1^n)$ liefert gleichverteilt zufälliges Element aus $\{(e, e) \mid e \in \mathbb{Z}_{2^{128}}\}$
- $\mathcal{E}(e, m_1 \cdots m_n) = \text{AES}_e(m_1) \cdots \text{AES}_e(m_n)$
- $\mathcal{D}(e, c_1 \cdots c_n) = \text{AES}_e^{-1}(c_1) \cdots \text{AES}_e^{-1}(c_n)$

Im Folgenden beschreiben wir die Bijektion AES_K , die einen 16-Byte-Klartext mit dem 16-Byte-Schlüssel K zu einem 16-Byte-Chiffretext verschlüsselt.

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, entspricht 4×4 Matrix über \mathbb{F}_{256} .
(AES-Standard erlaubt auch 24-Byte- und 32-Byte-Schlüssel)

Schlüssel K wird mit einer festgelegten Funktion $f : \mathbb{F}_{256}^{4 \times 4} \rightarrow (\mathbb{F}_{256}^{4 \times 4})^{11}$ expandiert. Es entstehen die Schlüssel $K_0, \dots, K_{10} \in \mathbb{F}_{256}^{4 \times 4}$.

Wir definieren Operationen, die im AES verwendet werden.

- Die Bijektion **SubByte** : $\mathbb{F}_{256}^{4 \times 4} \rightarrow \mathbb{F}_{256}^{4 \times 4}$ wendet eine Funktion S-box komponentenweise auf 4×4 Matrizen über \mathbb{F}_{256} an.

Dabei ist S-box : $\mathbb{F}_{256} \rightarrow \mathbb{F}_{256}$ 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	e3	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	le0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	le7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	lb4	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	l70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	le1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	l8c	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** : $\mathbb{F}_{256}^{4 \times 4} \rightarrow \mathbb{F}_{256}^{4 \times 4}$ verschiebt die Matrixzeilen.

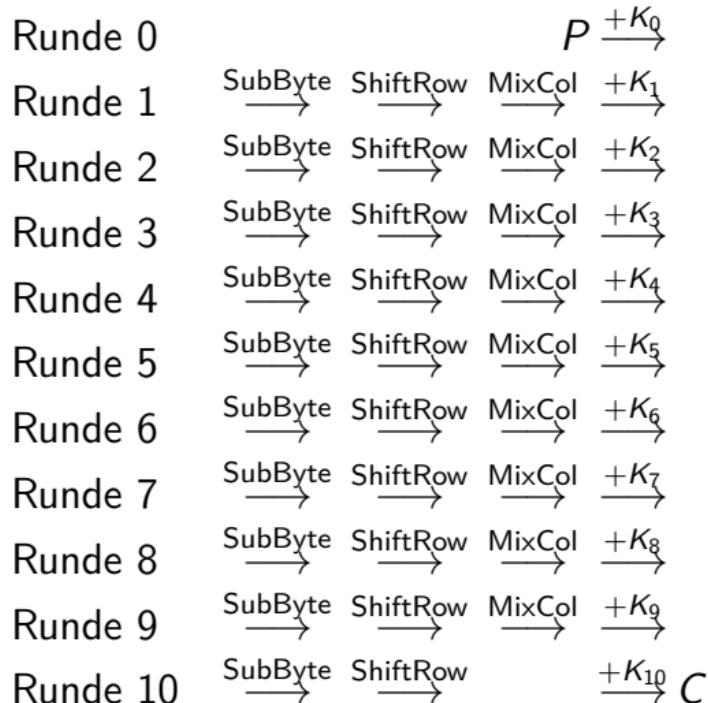
$$\text{ShiftRow} \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{pmatrix} \stackrel{df}{=} \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_2 & b_3 & b_4 & b_1 \\ c_3 & c_4 & c_1 & c_2 \\ d_4 & d_1 & d_2 & d_3 \end{pmatrix}$$

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

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

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

AES_K verschlüsselt den Klartext P wie folgt zum Chiffretext C .



2.8 Betriebsmodi für Blockchiffren

AES ist eine sogenannte Blockchiffre.

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

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

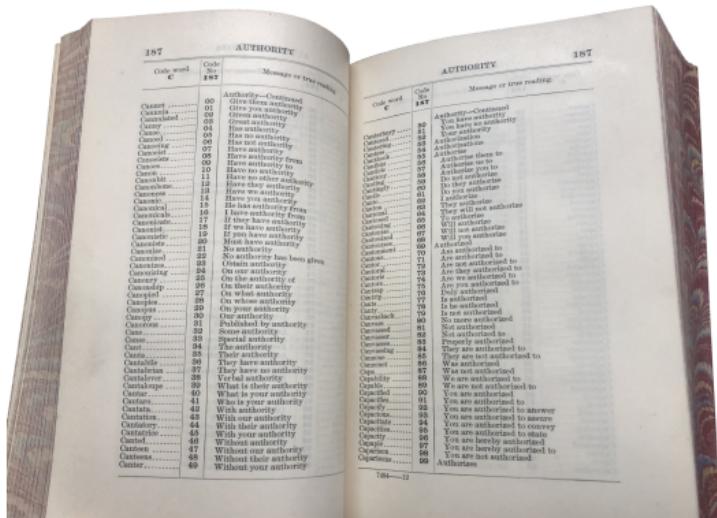
Betriebsmodus = Festlegung, wie mit der Blockchiffre lange Klartexte verschlüsselt werden

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
- ⇒ Chiffertext kann viel Information über den Klartext liefern

Beispiel: AES-Verschlüsselung im ECB-Modus



Klartext



Chiffertext

⇒ 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

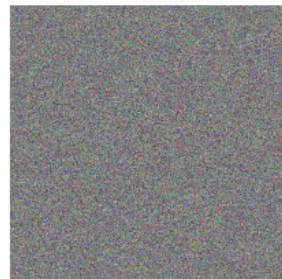
Eigenschaften des CBC-Modus

- Chiffretext von Block i hängt von Initialisierungsvektor und allen Blöcken $1, \dots, i$ ab
 - gleiche Blöcke werden unterschiedlich verschlüsselt
- ⇒ Chiffretext liefert wenig Information über den Klartext

Beispiel: AES-Verschlüsselung im CBC-Modus



Klartext



Chiffretext

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

3 Perfekte Sicherheit

Ausblick

- Nach einigen Grundlagen aus der Wahrscheinlichkeitstheorie erarbeiten wir eine Definition für *perfekte Sicherheit*.
- Dabei folgen wir der Idee, dass ein Angreifer durch das Abhören einer verschlüsselten Nachricht kein zusätzliches Wissen über die Originalnachricht bekommen soll.
- Wir erarbeiten einfache Bedingungen, aus denen die perfekte Sicherheit eines Kryptosystems folgt.
- Es zeigt sich, dass es sehr einfache Kryptosysteme mit perfekter Sicherheit gibt. Wir beweisen die perfekte Sicherheit der Vernam-Chiffre.

3.1 Grundlagen der Wahrscheinlichkeitstheorie

Definition 3.1 (Wahrscheinlichkeitsverteilung) Sei \mathcal{M} eine höchstens abzählbar unendliche Menge (die **Ergebnismenge**).

- Eine **Verteilung** auf \mathcal{M} ist eine totale Funktion $P : \mathcal{M} \rightarrow [0, 1]$, sodass $\sum_{x \in \mathcal{M}} P(x) = 1$.
- Ein **Ereignis** A ist eine Teilmenge von \mathcal{M} .
- Die **Wahrscheinlichkeit eines Ereignisses** A ist definiert durch $P(A) \stackrel{\text{df}}{=} \sum_{x \in A} P(x)$.
- Sind $A, B \subseteq \mathcal{M}$ Ereignisse mit $P(B) > 0$, so ist die **Wahrscheinlichkeit von A unter der Bedingung B** definiert durch $P(A | B) \stackrel{\text{df}}{=} P(A \cap B) / P(B)$.
- A und B heißen **unabhängig**, falls $P(A \cap B) = P(A) \cdot P(B)$, oder äquivalent $P(A | B) = P(A)$.

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

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

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

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 .

$K_n \stackrel{\text{df}}{=} \{e \mid (e, d) \text{ wird auf einem erfolgreichen Rechenweg von } \mathcal{K}(1^n) \text{ ausgegeben}\}$

(mögliche Verschlüsselungsschlüssel für Nachrichten aus Σ^n)

$C_n \stackrel{\text{df}}{=} \{c \mid c \text{ wird auf einem erfolgreichen Rechenweg von } \mathcal{E}(e, m) \text{ mit } e \in K_n \text{ und } m \in \Sigma^n \text{ ausgegeben}\}$

(mögliche Chiffretexte von Nachrichten aus Σ^n)

$P_{K_n}(e) \stackrel{\text{df}}{=} \text{Wahrscheinlichkeit, dass ein erfolgreicher Rechenweg von } \mathcal{K}(1^n) \text{ eine Ausgabe } (e, d) \text{ macht}$
(die von $\mathcal{K}(1^n)$ realisierte Verteilung auf K_n)

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

$$P(m, e, c) \stackrel{\text{df}}{=} P_{\Sigma^n}(m) \cdot P_{K_n}(e) \cdot \left(\begin{array}{l} \text{Wkt., dass ein erfolgreicher} \\ \text{Rechenweg von } \mathcal{E}(e, m) \text{ eine} \\ \text{Ausgabe } c \text{ macht} \end{array} \right).$$

Außerdem definieren wir folgende Ereignisse.

$$E_m \stackrel{\text{df}}{=} \{m\} \times K_n \times C_n, \quad \text{für } m \in \Sigma^n \quad (\text{Nachricht } m \text{ liegt vor})$$

$$E_e \stackrel{\text{df}}{=} \Sigma^n \times \{e\} \times C_n, \quad \text{für } e \in K_n \quad (\text{Schlüssel } e \text{ liegt vor})$$

$$E_c \stackrel{\text{df}}{=} \Sigma^n \times K_n \times \{c\}, \quad \text{für } c \in C_n \quad (\text{Chiffertext } c \text{ liegt vor})$$

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 der Definition perfekter Sicherheit: „Abhören lohnt nicht.“

D.h. eine abgehörte verschlüsselte Nachricht c liefert dem Angreifer kein zusätzliches Wissen über die unverschlüsselte Nachricht m .

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, $n \geq 1$, P_{Σ^n} eine Verteilung auf Σ^n und P die zugehörige Verteilung auf $\Sigma^n \times \mathcal{K}_n \times \mathcal{C}_n$.

1. S ist **perfekt sicher bezüglich P_{Σ^n}** $\overset{df}{\iff}$ für alle $m \in \Sigma^n$ und alle $c \in \mathcal{C}_n$ mit $P(E_c) > 0$ gilt $P(E_m | E_c) = P(E_m)$.
2. S ist **perfekt sicher** $\overset{df}{\iff}$ für alle $n \geq 1$ und alle Verteilungen P_{Σ^n} auf Σ^n ist S perfekt sicher bezüglich P_{Σ^n} .

Mit den Notationen 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. $\forall e \in K_n P(E_e) = 1/|K_n|$
(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

$$(\neg 3x + 4) = 7 \cup$$


□

Satz 3.6 Vernam ist perfekt sicher.

Beweis

$$(\exists x + 4) = \exists u$$


□

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

$$(\exists x \in \mathcal{K}) \vdash \neg \psi$$


□

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

Dort enthält der Chiffretext zwar Informationen über den Klartext, aber diese 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.

4 Asymmetrische Kryptosysteme (Public-Key-Verfahren)

Ausblick

- Wir behandeln das RSA-Kryptosystem und zeigen seine Korrektheit. Außerdem betrachten wir algorithmische Aspekte und erfahren, wie sich die Komponenten des Verfahrens (z.B. Invertieren mod n) effizient implementieren lassen.
- Wir lernen den Diffie-Hellman-Schlüsselaustausch, das darauf basierende Elgamal-Kryptosystem und das Goldwasser-Micali-Kryptosystem kennen. Wir zeigen die Korrektheit der Verfahren und sehen uns auch hier algorithmische Aspekte an (Erzeugersuche in \mathbb{Z}_p^*).
- Schließlich gehen wir kurz auf Sicherheitsbegriffe für Public-Key-Kryptosysteme ein.

Hauptproblem der bisherigen Verfahren ist der Schlüsselaustausch.

Schlüsselaustauschproblem

Situation:

- A,B wollen Informationen über unsicheren Kanal austauschen
- dazu soll symmetrisches Verfahren (AES) verwendet werden

Frage: Wie können sich A und B 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: Veröffentlichung des ersten Schlüsselaustausch-Protokolls.

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:

- $\mathbb{Z}_n^* \stackrel{\text{df.}}{=} \{i \in \mathbb{N} \mid 1 \leq i \leq n \text{ und } \text{ggT}(i, n) = 1\}$
- $\varphi(n) \stackrel{\text{df.}}{=} |\mathbb{Z}_n^*|$ (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 $\mathbb{F}_p \stackrel{\text{df}}{=} (\mathbb{Z}_p, +, \cdot)$ ein Körper, wobei $+$ die Addition und \cdot die Multiplikation modulo p bezeichnen.

Ohne Beweis.

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

1. Falls $\text{ggT}(m, n) = 1$, so gilt $\varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$.
2. Falls p prim, 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 $a^{\varphi(n)} \equiv 1 \pmod{n}$.

Ohne Beweis.

Spezialfall:

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

Ohne Beweis.

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

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

Ohne Beweis.

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

Ohne Beweis.

4.2 RSA-Kryptosystem

- benannt nach seinen Entwicklern Ron Rivest, Adi Shamir, Leonard Adleman
- erstes veröffentlichtes Public-Key-Kryptosystem

4.2.1 Das RSA-Protokoll

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

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

Alice möchte Bob eine Nachricht m mit $0 \leq m \leq 2^{r+1} - 2$ senden.

1. Bob macht folgendes:

- wählt zufällig verschiedene Primzahlen $p, q \in \{2^l, \dots, 2^{l+1}-1\}$
- setzt $n \stackrel{\text{df}}{=} pq$, damit gilt $\varphi(n) = (p-1)(q-1)$
- wählt zufälliges $e \in \{2, \dots, \varphi(n)-1\}$ mit $\text{ggT}(e, \varphi(n)) = 1$
- berechnet $d \stackrel{\text{df}}{=} e^{-1} \bmod \varphi(n)$
- Bobs public key ist (n, e) , sein private key ist (n, d)
- veröffentlicht public key (n, e)

2. Alice macht folgendes:

- verschlüsselt $m \in \{0, \dots, 2^{r+1} - 2\}$ als $c \stackrel{\text{df}}{=} m^e \bmod n$
- 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 $n \stackrel{df}{=} pq$. Sei $1 < e < \varphi(n)$ mit $\text{ggT}(e, \varphi(n)) = 1$ und $1 < d < \varphi(n)$ mit $e \cdot d \equiv 1 \pmod{\varphi(n)}$. Dann gilt für alle $m \in \{0, \dots, n-1\}$

$$m = (m^e)^d \pmod{n}.$$

Beweis

$$(\exists x+4) = 7 \cup$$


□

Beispiel 4.12 (RSA-Protokoll)

$$(\exists x+4) = 7 \cup$$


Definition 4.13 (RSA-Kryptosystem) RSA $\stackrel{df}{=} (\Sigma, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit

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

4.2.2 Effizienz des RSA-Protokolls

Wahl der Primzahlen p und q

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

1. wähle gleichverteilt zufällige $p, q \in \{2^l, \dots, 2^{l+1} - 1\}$;
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 $\{2, \dots, N\}$ liegt für große N zwischen

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

64

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

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

Damit gilt: Erfolgswahrscheinlichkeit in Zeile 1 ist $\geq 1/4l^2$.

Bei $4l^2k$ -facher Wiederholung ist die Erfolgswahrscheinlichkeit

$$1 - \left(1 - \frac{1}{4l^2}\right)^{4l^2k} \geq 1 - \frac{1}{2^k} \quad \text{für genügend große } l,$$

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

Für $k = l$ und $4^{l/2}k$ -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 Wkt. = 1.
 2. Ist $x \notin \mathbb{P}$, so liefert MillerRabin(x) den Wert 1 mit Wkt. $\leq 1/2$.
- Bei k -facher Wiederholung des Tests erhält man die 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).

Anzahl der $e \in \{2, (p-1)(q-1)-1\}$ mit $\text{ggT}(e, (p-1)(q-1)) = 1$ ist nach Definition 4.2 und Satz 4.5.3 genau

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

Die Erfolgswahrscheinlichkeit bei $k \cdot \mathcal{O}(\log I)$ -facher Wiederholung ist dann mindestens $1 - 2^{-k}$. (gleiches Argument wie auf Seite 65)

Berechnung des privaten Schlüssels d

Die Berechnung von $d \stackrel{df}{=} e^{-1} \bmod (p-1)(q-1)$ 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 $d \stackrel{df}{=} y \bmod (p-1)(q-1)$ 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.

Betrachten nun die effiziente Berechnung von $m^e \bmod n$.

Der naive Algorithmus hat eine Laufzeit von $e \cdot |n| \in \mathcal{O}(2^{|n|} |n|)$. Dies ist 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 $m^e = m^{e_t 2^t + e_{t-1} 2^{t-1} + \cdots + e_0 2^0} = m^{e_0 2^0} \cdot m^{e_1 2^1} \cdots m^{e_t 2^t}$
- wegen $m^{2^{i+1}} = (m^{2^i})^2$ lassen sich die $(m^{e_i 2^i} \bmod n)$ in polynomieller Zeit in der Eingabelänge berechnen
- mit obiger Gleichung erhält man insgesamt einen Polynomialzeitalgorithmus

Die beschriebene Methode nennt man „square & multiply“.

Beispiel 4.15 (square and multiply)

$$(-3x+4)^2 = 7x^2$$


Folgerung 4.16 RSA ist ein Kryptosystem.

Beweis

- Schlüsselgenerator \mathcal{K} ist ein prob. Polynomialzeitalgorithmus, da die Wahl der Primzahlen p, q , die Wahl des 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 $\mathcal{K}(1')$ 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 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.

Vermutung: Faktorisieren und Brechen von RSA sind nicht NP-hart.

Quantencomputer können in polynomieller Zeit faktorisieren und damit RSA brechen.

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 Praxis gilt 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 $\sqrt[e]{\quad}$)
- 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 durch 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

- 1976 lösten Diffie und Hellman das Schlüsselaustauschproblem: Alice und Bob einigen sich über unsicheren Kanal auf geheimen Schlüssel
- Diffie-Hellman-Verfahren (DH) beendet lange Tradition von Codebüchern (z.B. bei Enigmas für jeden Tag des Jahres ein eigener Code)
- DH ähnelt einem Public-Key-Kryptosystems: sichere Kommunikation über unsicheren Kanal ohne vorher vereinbarten Schlüssel
- Unterschied zu Public-Key-Kryptosystemen: beim DH müssen beide Parteien online sein und miteinander kommunizieren
- 1985 veröffentlichte Taher Elgamal ein Public-Key-Kryptosystem, das auf DH basiert
- Idee des DH: das Potenzieren modulo einer Primzahl p ist einfach (square & multiply), Logarithmieren modulo p ist 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 **Primivwurzel** modulo n , falls

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

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

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 γ ein Erzeuger von \mathbb{Z}_p^* . Wir definieren folgende Funktionen $\mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$.

$$\exp_{\gamma,p}(a) \stackrel{\text{df.}}{=} \gamma^a \bmod p, \text{ wobei } 1 \leq a \leq p-1$$

$$\begin{aligned} \log_{\gamma,p}(b) &\stackrel{\text{df.}}{=} \text{das eindeutig bestimmte } a \in \{1, \dots, p-1\} \\ &\text{mit } \exp_{\gamma,p}(a) = b, \text{ wobei } 1 \leq b \leq p-1 \end{aligned}$$

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 zufälliges $a \in \{1, \dots, p-1\}$ und sendet $\alpha \stackrel{\text{df}}{=} \exp_{\gamma,p}(a)$.
Bob wählt zufälliges $b \in \{1, \dots, p-1\}$ und sendet $\beta \stackrel{\text{df}}{=} \exp_{\gamma,p}(b)$.
3. Alice berechnet geheimen Schlüssel $k_A \stackrel{\text{df}}{=} \beta^a \bmod p$.
Bob berechnet geheimen Schlüssel $k_B \stackrel{\text{df}}{=} \alpha^b \bmod p$.

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

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

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 $\text{ord}_p(\gamma) = p - 1$.

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

$$\text{ord}_p(\gamma) = d \iff \gamma^d \equiv 1 \pmod{p} \text{ und f\"ur alle } c \mid d \\ \text{mit } c < d \text{ gilt } \gamma^c \not\equiv 1 \pmod{p}.$$

Beweis

$$(\exists x+1) = 7 \cup$$


□

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

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

Beweis

$$(\exists x + 4) = 7 \cup$$



□

Wir wissen:

- \mathbb{Z}_p^* hat mindestens $\frac{p-1}{6 \ln \ln(p-1)}$ Erzeuger. (Folgerung 4.21)
- 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 $\{1, \dots, n\}$ 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 große Sophie-Germain-Primzahl q (nach obiger Vermutung ist eine solche mit hoher Wahrscheinlichkeit unter $O(\log^2 n)$ zufällig gewählten Zahlen aus $\{1, \dots, n\}$)
2. $p \stackrel{df}{=} 2q + 1$
3. wählen zufällige $\gamma \in \{2, \dots, p - 2\}$, 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 $\gamma \in \{2, \dots, p - 2\}$ 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 $\{2, \dots, p - 2\}$, denn 1 und $p - 1$ sind keine Erzeuger.
- $|\{2, \dots, p - 2\}| = p - 3$

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 $r \in \mathbb{N}$
- $G \stackrel{df}{=} \{x^r \bmod p \mid x \in \mathbb{Z}_p^*\}$ ist Untergruppe von \mathbb{Z}_p^* mit $|G| = q$
- wähle $\gamma \in G - \{1\}$ gleichverteilt zufällig
(wähle gleichvert. zufälliges $x \in \mathbb{Z}_p^*$, $\gamma \stackrel{df}{=} (x^r \bmod p)$, 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-Schlüsselaustauschs

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 gegebenem Erzeuger γ von \mathbb{Z}_p^* und gegebenem $\alpha \in \mathbb{Z}_p^*$ lässt sich in Polynomialzeit $\log_{\gamma,p}(\alpha)$ berechnen
- Angreifer hört $\alpha = \exp_{\gamma,p}(a)$ ab und berechnet $\log_{\gamma,p}(\alpha) = a$
- Angreifer hört $\beta = \exp_{\gamma,p}(b)$ ab und berechnet gemeinsamen Schlüssel $k = \beta^a \bmod p$

Bemerkungen:

1. Der Satz 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. D.h. Alice und Bob kommunizieren über einen Kanal, der komplett durch den Angreifer Mallory kontrolliert wird.



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

4.4 Elgamal-Kryptosystem

Das 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 zufälligen Erzeuger γ von \mathbb{Z}_p^*
- wählt zufälliges $b \in \{1, \dots, p - 1\}$, berechnet $B := \gamma^b \bmod p$
- öffentlicher Schlüssel (p, γ, B) , privater Schlüssel (p, γ, b)

2. Alice

- stellt Nachricht als Folge von Zahlen $m \in \{1, \dots, p - 1\}$ dar, verschlüsselt jedes m wie folgt:
 - wählt zufälliges $a \in \{1, \dots, p - 1\}$, berechnet $A := \gamma^a \bmod p$
 - berechnet $c := B^a m \bmod p$, sendet Chiffertext (A, c) an Bob

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

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

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

Wir wissen nicht, ob es unendlich viele Sophie-Germain-Primzahlen gibt. Daher haben wir Elgamal für festes q definiert.

Die Suche nach Erzeugern von \mathbb{Z}_p^* kann effizient durchgeführt werden (siehe Abschnitt 4.3.3).

Ein Nachteil des Verfahren ist, dass die Chiffertexte 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 Chiffertexte.

Für Alice ist es empfehlenswert für jedes m_k einen neuen Exponenten a_k zu wählen. Hat man nämlich

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

und kennt schon m , so lässt sich m' rekonstruieren:

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

4.5 Goldwasser-Micali-Kryptosystem

- erstes probabilistisches Public-Key-Kryptosystem (1982), d.h. ein System mit probabilistischem Verschlüsselungsalgorithmus
- beweisbar sicher unter bestimmten kryptographischen Annahmen
- ineffizient, da Chiffretexte wesentlich länger als Klartexte sind

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

4.5.1 Das Goldwasser-Micali-Protokoll

1. Bob

- wählt zufällige, unabhängige $p, q \in \mathbb{P}$
mit $p \equiv q \equiv 3 \pmod{4}$ und $p \neq q$
- öffentlicher Schlüssel $n = pq$, privater Schlüssel (p, q)

2. Alice verschlüsselt eine Bitfolge b_1, \dots, b_m wie folgt:

- wählt zufällige, unabhängige $y_1, \dots, y_m \in \mathbb{Z}_n^*$
- $c_i := ((-1)^{b_i} y_i^2) \pmod{n}$ für $i = 1, \dots, m$
- sendet c_1, \dots, c_m an Bob

3. Bob entschlüsselt c_1, \dots, c_m zu b'_1, \dots, 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 $\text{GM}_\lambda := (\{0, 1\}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, wobei $\lambda \in \mathbb{N}$ die Länge der Chiffrertexte 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 $p \equiv q \equiv 3 \pmod{4}$, und setze $n := pq$.
 2. return $(n, (p, q))$
- $\mathcal{E}(n, b_1 \cdots b_m)$ gibt $c_1 \cdots c_m$ mit $c_i := ((-1)^{b_i} y_i^2) \pmod{n}$ aus, wobei die $y_i \in \mathbb{Z}_n^*$ zufällig und unabhängig gewählt werden.
- $\mathcal{D}((p, q), c_1 \cdots c_m) = b'_1 \cdots b'_m$ mit

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

4.5.2 Korrektheit des Verfahrens

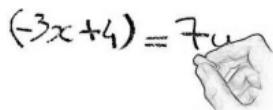
Definition 4.32 Für ungerade $p \in \mathbb{P}$ und $x \in \mathbb{Z}_p^*$ ist das Legendre-Symbol wie folgt definiert.

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

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

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

Beweis



□

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

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

Beweis

$$(\exists x+4) = 7 \cup$$



□

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

Beweis Angenommen $pq - 1$ ist ein quadratischer Rest mod pq .

Aus 4.34 folgt $(pq-1)^{\frac{p-1}{2}} \text{ mod } p = 1$ und damit $-1^{\frac{p-1}{2}} \equiv 1 \pmod{p}$.

Also ist $p-1/2$ gerade und damit $4 \mid p-1$. D.h. $p \equiv 1 \pmod{4}$.

Dies widerspricht $p \equiv 3 \pmod{4}$.

□

Wir begründen die Korrektheit des GM-Kryptosystems wie folgt:

- Man weiß, dass für $k \rightarrow \infty$ folgendes gilt:
Die Hälfte der Primzahlen $p \leq k$ hat die Eigenschaft $p \equiv 3 \pmod{4}$.
- Bei GM sind die Primzahlen p, q so gewählt, dass $p \equiv q \equiv 3 \pmod{4}$ 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 $c_i = y_i^2 \pmod{n}$ ein quadratischer Rest mod n .
- Falls $b_i = 1$, so ist $c_i = (-1 \cdot y_i^2) \pmod{n}$. Wäre dies ein quadratischer Rest mod n , so wäre $-1 \cdot y_i^2 \equiv x^2 \pmod{n}$ für ein $x \in \mathbb{Z}_n^*$, also $n - 1 \equiv (xy_i^{-1})^2 \pmod{n}$ und damit wäre $n - 1$ ein quadratischer Rest mod n , ein Widerspruch. Also ist c_i ein quadratischer Nichtrest mod n .
- Insgesamt gilt: $b_i = 0 \Leftrightarrow c_i$ 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 prob. Polynomialzeit, da asymptotisch die Hälfte der Primzahlen die Eigenschaft $p \equiv 3 \pmod{4}$ hat.

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

$$b'_i := \begin{cases} 0 & \text{falls } c_i^{\frac{p-1}{2}} \pmod{p} = c_i^{\frac{q-1}{2}} \pmod{q} = 1, \\ 1 & \text{sonst} \end{cases}$$

Damit lassen sich die b'_i effizient berechnen (square and multiply).

Die restlichen Komponenten des Verfahrens lassen sich leicht 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 der Form $n = pq$ mit $p, q \in \mathbb{P}$, $p \equiv q \equiv 3 \pmod{4}$ und $p \neq q$).

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

4.5.5 Anwendung des GM 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

Commitment mittels GM:

- Bob wählt p, q, y wie bei GM und veröffentlicht $n = p \cdot q$ und $c = (-1)^b y^2 \bmod n$.
- 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)

$$(-3x+4) = 7u$$


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.

Die Forderung ist relativ schwach, da beispielsweise nicht gefordert ist, dass das System bei Kenntnis von zwei verschiedenen Chiffertexten $\mathcal{E}(e, m_1)$ 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 $\frac{1}{2} + \frac{1}{q(n)}$ ist:

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 sagt, ob Bob $m = m_1$ oder $m = m_2$ gewählt hatte.

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.

5 Digitale Signatur

Ausblick

- Digitale Signaturverfahren stellen sicher, dass Nachrichten vom angegebenen Absender stammen, nicht gefälscht sind und der Absender die Urheberschaft im Nachhinein nicht leugnen kann.
- Wir starten mit kryptographischen Hashfunktionen, da sie ein wichtiger Bestandteil von digitalen Signaturverfahren sind.
- Wir lernen die RSA-Signatur und die Elgamal-Signatur kennen, sie basieren auf den entsprechenden Public-Key-Kryptosystemen.
- Mit der Lamport-Diffie-Einmal-Signatur behandeln wir ein einfaches Verfahren, das nicht auf einem Public-Key-Kryptosystem, sondern nur einer Einwegfunktion basiert.

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

- Integrität: Fälschung/Manipulation der Nachricht nicht möglich
- Authentizität: Nachweis, dass Nachricht vom Absender stammt
- Verbindlichkeit: Absender kann Urheberschaft 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 $h : \Sigma^* \rightarrow \Sigma^n$ für ein $n \in \mathbb{N}$.

Beispiel 5.2 Die Funktion $h : \{0, 1\}^* \rightarrow \{0, 1\}$ 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 $f : \Sigma^* \rightarrow \Sigma^*$ 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:

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

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

Es ist nicht bekannt, ob es Einwegfunktionen gibt.

Man vermutet/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 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(e) man als Kandidaten für kryptographische Hashfunktionen:

Hashfnkt.	Anzahl der Bits der Hashwerte	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 Standardisierung Änderungen am Algorithmus vor; Forscher kritisieren, dass diese Änderungen die Sicherheit beeinträchtigen.

5.2 RSA-Signatur

Idee:

- Alice komprimiert Text x mit kryptogr. Hashfnkt. h zu $m = h(x)$.
- Alice entschlüsselt m und sendet $(x, \mathcal{D}((n, d), m))$ an Bob.
- Bob prüft $\mathcal{E}((n, e), \mathcal{D}((n, d), m)) \stackrel{?}{=} h(x)$.

5.2.1 Das Protokoll

Alice signiert einen Text x wie folgt.

1. Alice erzeugt Schlüsselpaar wie beim RSA:

- wählt zufällig verschiedene Primzahlen p, q
- $n \stackrel{df}{=} pq$, damit gilt $\varphi(n) = (p - 1)(q - 1)$
- wählt zufälliges $e \in \{2, \dots, \varphi(n) - 1\}$ mit $\text{ggT}(e, \varphi(n)) = 1$
- berechnet $d \stackrel{df}{=} e^{-1} \bmod \varphi(n)$
- Alices public key ist (n, e) , ihr private key ist (n, d)
- $h : \{0, 1\}^* \rightarrow \{0, \dots, n-1\}$ sei öffentliche, kryptogr. Hashfnkt.

2. Alice signiert:

- berechnet $m := h(x)$, dann die Unterschrift $\text{sig}_A(m) = m^d \bmod n$ und sendet schließlich $(x, \text{sig}_A(m))$ an Bob

3. Bob prüft $h(x) = \text{sig}_A(m)^e \bmod n$.

Bemerkung 5.6

- Die Korrektheit des Verfahrens zeigt man wie beim RSA:
 $\text{sig}_A(m)^e \stackrel{4.11}{\equiv} m \equiv h(x) \pmod{n}$
- Zum Fälschen einer Unterschrift muss man in der Lage sein, verschlüsselte Nachrichten an Alice zu entschlüsseln.
- Das Prinzip der RSA-Signatur lässt sich auf alle public-key Kryptosysteme mit folgender Eigenschaft übertragen:

$$\forall m \quad \mathcal{E}(e, \mathcal{D}(d, m)) = m \tag{1}$$

D.h. Ver- und Entschlüsselung dürfen vertauscht werden.

5.3 Elgamal-Signatur

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

1. Alice erzeugt Schlüsselpaar:

- wählt zufällige Primzahl p und Erzeuger $\gamma \in \mathbb{Z}_p^*$
- wählt zufälliges $a \in \{1, \dots, p-2\}$, berechnet $A := \gamma^a \bmod p$
- Alices public key ist (p, γ, A) , ihr private key ist (p, γ, a)
- $h : \{0, 1\}^* \rightarrow \{0, \dots, p-1\}$ sei öffentliche, kryptogr. Hashfnkt.

2. Alice signiert:

- wählt zufälliges $k \in \{1, \dots, p-2\}$ mit $\text{ggT}(k, p-1) = 1$
- $r := \gamma^k \bmod p$, $k' := k^{-1} \bmod p-1$, $s := k'(h(x)-ar) \bmod p-1$
(falls $s = 0$, so wähle k, r, s erneut)
- sendet (x, r, s) an Bob

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

Satz 5.7 (Korrektheit der Elgamal-Signatur)

Mit den obigen Bezeichnungen gilt $A^r r^s \equiv \gamma^{h(x)} \pmod{p}$.

Beweis

$$(-3x+4) = 7x$$


□

Bemerkung 5.8

- Der Wert k muss für jede Signatur 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 $H : \{0, 1\}^* \rightarrow \{0, 1\}^*$ eine längentreue Einwegfunktion, d.h. $|H(x)| = |x|$ für alle $x \in \{0, 1\}^*$.

Alice signiert einen Text $d = d_1 \cdots d_n$ mit $d_1, \dots, d_n \in \{0, 1\}$ wie folgt.

1. Alice erzeugt Schlüsselpaar:

- wählt einen Sicherheitsparameter $k \in \mathbb{N}$
- erzeugt eine Liste $X = (x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, \dots, x_{n,0}, x_{n,1})$ mit $2n$ zufälligen Wörtern aus $\{0, 1\}^k$
- erzeugt $Y = (y_{1,0}, y_{1,1}, y_{2,0}, y_{2,1}, \dots, y_{n,0}, y_{n,1})$ mit $y_{i,j} := H(x_{i,j})$
- veröffentlicht Verifikationsschlüssel Y , hält Signaturschlüssel X geheim

2. Alice signiert:

- erzeugt $S = (s_1, \dots, s_n)$ 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 $(H(s_1), \dots, H(s_n)) = (y_{1,d_1}, \dots, y_{n,d_n})$

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“).

6 Identifikation

Ausblick

- Wir behandeln Verfahren, mit denen sich sicherstellen lässt, dass man mit einer bestimmten Person und nicht mit einem Betrüger kommuniziert.
- Dabei konzentrieren wir uns auf Challenge-Response-Verfahren. Hier ist eine Identifikation möglich, ohne dass man Informationen über das eigene Passwort preisgeben muss (Zero-Knowledge-Eigenschaft).
- Wir behandeln das Fiat-Shamir-Protokoll und diskutieren die Sicherheit sowie die Zero-Knowledge-Eigenschaft.
- Schließlich überlegen wir uns, wie man allgemein einen Beweis in einen Zero-Knowledge-Beweis umwandeln kann.

Wir widmen uns folgender Frage:

Wie kann man sicherstellen, dass man mit einer bestimmten Person und nicht mit einem Betrüger kommuniziert?

Identifikation durch Senden eines Passworts hat Nachteile:

1. im Klartext übertragene Passwörter können abgehört werden
2. nach einer Identifikation kennt der Verifier das Passwort des Provers und könnte sich gegenüber anderen Systemen als Prover ausgeben

Eine bessere Identifikationsstrategie sollte zwei Dinge erfüllen:

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

6.1 Einmal-Passwörter

Der Prover besitzt eine Liste w_1, \dots, w_n geheimer Passwörter.

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

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

Nachteil des Verfahrens: zu Beginn muss eine lange Passwort-Liste erzeugt und ausgetauscht werden.

6.2 Challenge-Response-Identifikation

Idee:

- Verifier stellt Prover eine Frage, 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.

Es ist ein Zero-Knowledge-Protokoll:

D.h. der Verifier kann sich mit hoher Wkt. überzeugen, dass der Prover ein gewisses Geheimnis besitzt, ohne dass der Prover 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.

Fiat-Shamir-Protokoll:

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

$$(\exists x \cdot x^2 = v) \equiv \text{Fals}$$

□

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. $\geq \frac{1}{2} + \varepsilon$ identifizieren kann.

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

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 Erfolgswkt. $1/2$, 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/Betrüger gewinnen.

Beweis

$$(\exists x \in \mathbb{Z}) \vdash \neg \psi$$


□

Protokolle, die Aussagen im Stil von 6.4 genügen, 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 $\sigma : V_0 \rightarrow V_1$ 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

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 $\sigma : G_0 \rightarrow G_1$ zu verraten.

Nehmen o.B.d.A. an, dass G_0 und G_1 die gleiche Knotenmenge besitzen, d.h. $G_0 = (V, E_0)$ und $G_1 = (V, E_1)$.

1. Peggy wählt zufällige Permutation $\tau : V \rightarrow V$, wendet diese auf G_0 an und sendet 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 $G_c \rightarrow H$.)
3. Ist $c = 0$, so sendet Peggy $\pi := \tau$, andernfalls $\pi := \tau\sigma^{-1}$.
4. Victor prüft, ob π tatsächlich ein Isomorphismus $G_c \rightarrow H$ 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 Wkt. $\geq 1/2$ 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 Betrugswkt. $\leq 1/2^k$.

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

6.3.3 Ein Zero-Knowledge-Beweis für 3-Färbbarkeit

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

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

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

Man weiß, dass dieses Problem NP-vollständig ist.

Die 3-Färbbarkeit von G lässt sich durch Angabe einer entsprechenden Färbung beweisen.

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

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

Wir verwenden das Goldwasser-Micali-Commitment (Seite 98), um eine Information x in verschlüsselter Form $y = \text{GMC}(x)$ zu hinterlegen.

1. Peggy wählt zufällige Permutation $\pi : \{1, 2, 3\} \rightarrow \{1, 2, 3\}$, berechnet permutierte Färbung c'_1, \dots, c'_n mit $c'_i = \pi(c_i)$ und sendet y_1, \dots, y_n mit $y_i = \text{GMC}(c'_i)$ an Victor.
2. Victor wählt zufällige Kante $\{a, b\} \in E$ und sendet sie 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, \dots, c_n Victor in jedem Fall überzeugen.

Sicherheit: Ist $G \notin 3\text{-FARB}$, so besitzt jede von einem Betrüger hinterlegte Färbung c'_1, \dots, 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 Wkt. $\geq 1/n^2$. Bei $k \cdot n^2$ -facher Anwendung des Protokolls ist die Betrugswkt. $\leq 1/2^k$.

Zero-Knowledge: Im Verlauf des Protokolls sieht Victor neben den Commitments y_1, \dots, 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, \dots, y_n keine zusätzlichen Informationen.

Beispiel 6.7 (Zero-Knowledge-Beweis für 3-FARB)

$$(3x+4) = ?$$


6.3.4 Zero-Knowledge-Beweis für beliebige NP-Probleme

Sei $A \in \text{NP}$, d.h. es existiert ein $B \in \text{P}$ 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 $A \leq 3\text{SAT} \leq \text{3-FARB}$ 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 $G \in \text{3-FARB}$.
4. Ist der Beweis korrekt, so gilt $f(x) \in \text{3-FARB}$ 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.

7 Secure Multiparty Computation

Ausblick

- In diesem Abschnitt geht es um verschiedene kryptographische Aspekte von Berechnungen, an denen mehrere Parteien beteiligt sind.
- Das Secret Sharing erlaubt die Aufteilung eines Geheimnisses auf n Personen, sodass sich nur in Anwesenheit von t Personen das Geheimnis rekonstruieren lässt.
- Beim Millionärsproblem kommunizieren zwei Superreiche so, dass am Ende der Reichere feststeht, aber keiner das Vermögen des anderen kennt.
- Mit Secure Circuit Evaluation können zwei Parteien einen Wert $f(x)$ berechnen, wobei die eine Partei die geheimen Daten x und die andere den geheimen Algorithmus f besitzt.

- Mit vollständig homomorphen Kryptosystemen lassen sich Daten sicher verschlüsseln und gleichzeitig beliebige Funktionsberechnungen auf den verschlüsselten Daten durchführen.

7.1 Secret Sharing

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.

Die Lagrange-Interpolation funktioniert in beliebigen Körpern, wir benötigen sie für die Körper \mathbb{F}_p mit $p \in \mathbb{P}$.

Satz 7.2 (Lagrange-Interpolation) Seien $p \in \mathbb{P}$, $t \in \mathbb{N}^+$ und $x_1, \dots, x_t, y_1, \dots, y_t \in \mathbb{F}_p$ 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 $k = 1, \dots, t$, nämlich

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

Beweis

$$(-3x+4) = 7$$



Shamir-Secret-Sharing-Protokoll mit Parametern (n, t) :

1. Initialisierung:

Der Dealer wählt eine Primzahl $p \geq n + 1$, paarweise verschiedene Elemente $x_1, \dots, x_n \in \mathbb{Z}_p^*$ 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 $a_1, \dots, a_{t-1} \in \mathbb{Z}_p$, konstruiert das Polynom $a(x) \stackrel{\text{df}}{=} a_{t-1}x^{t-1} + \dots + a_1x + s \in \mathbb{F}_p[x]$, berechnet die Geheimnisteile $y_i \stackrel{\text{df}}{=} a(x_i)$ für $i = 1, \dots, n$ und sendet y_i an den Teilnehmer i .

3. Rekonstruktion:

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

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

Beispiel 7.3 (Shamir-Secret-Sharing)

$$(3x+4) = 7 \text{ für } x=1$$


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 Sicht der Teilnehmer $1, \dots, t-1$ jedes Geheimnis $y \in \mathbb{Z}_p$ gleich wahrscheinlich.

7.2 Das Millionärsproblem

Millionärsproblem: Alice und Bob 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 .

Die Vermögen von Alice und Bob seien a bzw. b .

1. Bobs Schlüsselpaar sei $(\text{pk}, \text{sk}) = ((n, e), (n, e'))$ und $m < n$ sei eine Obergrenze für a und b .
2. Alice wählt eine zufällige Zahl $x \leq n - 1$, verschlüsselt diese zu $c := \mathcal{E}(\text{pk}, x) = x^e \bmod n$ und sendet $d := (c - a) \bmod n$.
3. Bob berechnet $y_i := \mathcal{D}(\text{sk}, d + i) = (d + i)^{e'} \bmod n$ 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 teilt 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 $\mathcal{E}(\text{pk}, y_i)$ bilden kann und so sieht, ab welcher Stelle Bob inkrementiert hat.

Beispiel 7.5 (Millionärsproblem)

$$(-3x+4) = 7u$$


7.3 Secure Circuit Evaluation

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. Seien $p, q \in \mathbb{P}$ mit $p \equiv q \equiv 3 \pmod{4}$, $p \neq q$ und $n \stackrel{\text{df}}{=} pq$. Für $b \in \{0, 1\}$ und $k \in \mathbb{Z}_n^*$ sei $E(b, k) \stackrel{\text{df}}{=} (-1)^b k^2 \pmod{n}$.

Zur Erinnerung:

- $E(b, k)$ entspricht im GM-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 $\mathbb{N} \rightarrow \mathbb{N}$.

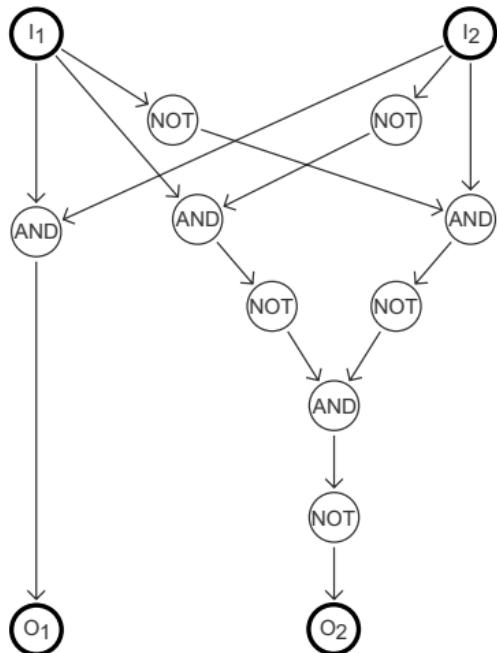
Polynomialzeit-Berechnungen lassen sich durch Schaltkreise polynomieller Größe simulieren. Genauer gilt:

Ist $f \in \text{FP}$, 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 $f \in \text{FP}$ 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.

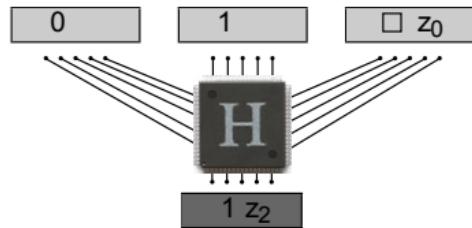


Schaltkreis mit Eingängen I_1, I_2 und Ausgängen O_1, O_2

Beweis (Skizze)

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

□	1 z_0	0	1	□	□	□
□	1	0 z_0	1	□	□	□
□	1	0	1 z_0	□	□	□
□	1	0	1	□ z_0	□	□
□	1	0	1	□	□	□
□	1	0	1 z_2	□	□	□
□	1	0 z_3	□	□	□	□
□	1 z_3	□	□	□	□	□
□ z_3	□	□	□	□	□	□
1 z_1	□	□	□	□	□	□



Der Inhalt jeder Zelle der Tabelle ist durch 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 . □

Ausgangspunkt der Secure Circuit Evaluation:

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

Protokoll zur Berechnung von $f(x)$:

1. Bob wählt verschiedene $p, q \in \mathbb{P}$ mit $p \equiv q \equiv 3 \pmod{4}$ und veröffentlicht $n := pq$.
2. Bob verschlüsselt seine Bits mit Zufallszahlen $k_1, \dots, k_m \in \mathbb{Z}_n^*$ 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 $E(b_i, k_i)$
 - NOT-Gatter:^{*} aus $E(b, k)$ berechnet Alice einen Wert $E(\neg b, k')$
 - AND-Gatter:^{*} aus $E(b, k)$ und $E(b', k')$ berechnet Alice mit Bobs Hilfe einen Wert $E(b \wedge b', k'')$
4. Am Ende schickt Alice die verschlüsselten Bits der Ausgänge an Bob, der sie entschlüsselt und so das Ergebnis $f(x)$ erhält.

*Berechnung der verschlüsselten Wahrheitswerte

- NOT-Gatter: aus $E(b, k)$ berechnet Alice einen Wert $E(\neg b, k')$

$$(\neg 3x + 4) = \text{?}$$


- AND-Gatter: aus $E(b, k)$ und $E(b', k')$ berechnet Alice mit Bobs Hilfe einen Wert $E(b \wedge b', k'')$

$$(\neg 3x + 4) = \text{?}$$


7.4 Vollständig Homomorphe Verschlüsselung

Bei der Secure Circuit Evaluation konnte Alice beliebige Berechnungen auf Daten durchführen, die von Bob verschlüsselt waren.

Die Berechnung der AND-Gatter erforderte jedoch Bobs Hilfe.

Vollständig homomorphe Verschlüsselung macht diese Hilfe überflüssig.

Ziel: beliebige Funktionsberechnungen f auf verschlüsselten Daten durchführen.

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

D.h. \mathcal{E} ist ein Homomorphismus im folgenden 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 $\{0, 1\}^n \rightarrow \{0, 1\}^n$ (die vom Kryptosystem beherrschten Funktionen).
3. Evaluate ist ein prob. 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

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

Wir verschaffen uns einen Überblick wie folgt:

1. lernen homomorphes Kryptosystem \mathcal{H} kennen, das nur eingeschränkte Funktionen unterstützt
2. skizzieren, wie man \mathcal{H} zu einem vollständig homomorphen Kryptosystem weiterentwickelt

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 :

- $N := \lambda$
- $P := \lambda^2$
- $Q := \lambda^5$

$\mathcal{K}(1^\lambda)$ 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 $m \equiv b \pmod{2}$.

$$\mathcal{D}(p, c) := (c \bmod p) \bmod 2$$

\Rightarrow Chiffertext = Vielfaches von p + „Rauschen“ mit Parität b

Berechnungen auf verschlüsselten Daten

Sei c' ein zuvor veröffentlichter Chiffertext einer 1.

1. spezielle Funktionen:

$$\text{Evaluate(AND, } c_1, c_2) \stackrel{\text{df}}{=} c_1 \cdot c_2$$

$$\text{Evaluate(NOT, } c) \stackrel{\text{df}}{=} c + c'$$

2. allgemeine Funktionen:

$$\text{Evaluate}(f, c_1, \dots, c_t) \stackrel{\text{df}}{=} C(c_1, \dots, c_t),$$

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

Berechnung von AND

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

$$c_1 = pq_1 + m_1 \text{ mit } m_1 \equiv b_1 \pmod{2}, |m_1| = N, |q_1| = Q$$

$$c_2 = pq_2 + m_2 \text{ mit } m_2 \equiv b_2 \pmod{2}, |m_2| = N, |q_2| = Q$$

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

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 Chiffertext eines Bits $b \in \{0, 1\}$.

$$c = pq + m \text{ mit } m \equiv b \pmod{2}, |m| = N, |q| = Q$$

$$c' = pq' + m' \text{ mit } m' \equiv 1 \pmod{2}, |m'| = N, |q'| = Q$$

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

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

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

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 $\text{NOT}(c)$ ein Rauschen von höchstens $k + 1$ Bit.
- Haben c_1, c_2 ein Rauschen von k_1, k_2 Bit, so hat $\text{AND}(c_1, c_2)$ 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 zu vollst. homomorphem Kryptosystem

\mathcal{H} lässt sich so modifizieren, dass es die eigene Entschlüsselungsfunktion $\mathcal{D}(\cdot, \cdot)$ beherrscht. Diese lässt sich dann mit einigen NOT und wenigen AND realisieren.

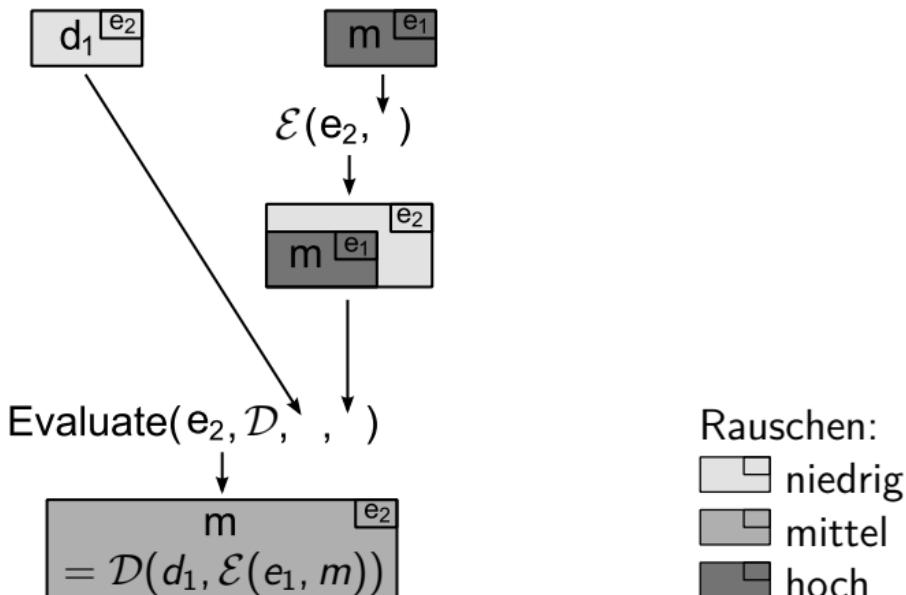
Dadurch kann man den Schlüssel von Chiffertexten wechseln, ohne den Chiffertext zu entschlüsseln.

Vorteil des Schlüsselwechsels: Chiffertext mit mittlerem Rauschen
Auswertung beliebiger Schaltkreise mit AND- und NOT-Gattern:

- solange Rauschen der Chiffertexte 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
- $\boxed{m^e}$ 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 $\mathcal{E}(e_2, d_1), \mathcal{E}(e_3, d_2), \dots$ 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).

8 Miller-Rabin-Primzahltest

Ausblick

- Wir behandeln den am häufigsten verwendeten Primzahltest und beweisen seine Korrektheit.

Wir definieren 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:

$$x \in A \implies P(M(x) = 1) \geq 1/2$$

$$x \notin A \implies P(M(x) = 1) = 0$$

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

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

$$x \in \mathbb{P} \implies P(M(x) = 1) = 1$$

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

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

Aus dem kleinen Fermatschen Satz folgt:

$$\exists a [1 \leq a < n \text{ und } a^{n-1} \not\equiv 1 \pmod{n}] \implies n \notin \mathbb{P}$$

Ein $a \in \mathbb{Z}_n^*$ mit $a^{n-1} \not\equiv 1 \pmod{n}$ 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 $a^{n-1} \equiv 1 \pmod{n}$, 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 $a^{n-1} \equiv 1 \pmod{n}$ 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 $p_1, \dots, p_m \in \mathbb{P} - \{2\}$.

Beweis

$$(\exists x+4) = 7 \cup$$


□

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

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 $w^2 \equiv 1 \pmod{p}$. Also $(w+1)(w-1) \equiv 0 \pmod{p}$. Damit gilt $p|w+1$ oder $p|w-1$. D.h. $w \equiv -1 \pmod{p}$ oder $w \equiv 1 \pmod{p}$. \square

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 **MR-Zeuge für n** $\overset{\text{df}}{\iff} a^m \not\equiv 1 \pmod{n}$ und für alle $j \in \{0, \dots, k-1\}$ gilt $a^{2^j \cdot m} \not\equiv -1 \pmod{n}$.
- a heißt **MR-Lügner für n** $\overset{\text{df}}{\iff} n$ ist zusammengesetzt und a ist kein MR-Zeuge für n .

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

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

Beweis

$$(\exists x + 4) = 7u$$


□

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 $\sigma_x : H \rightarrow H$ mit $\sigma_x(y) \stackrel{df}{=} x * y$.

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 H 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 gleichverteilt, zufälliges $a \in \{1, \dots, n - 1\}$
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 Ang. $P(\text{MillerRabin}(n) = 1) < 1$ für ein $n \in \mathbb{P} - \{2\}$.

D.h. auf mind. einem Rechenweg stoppt $\text{MillerRabin}(n)$ mit 0.

Dieser Rechenweg muss Zeile 8 erreichen. Also gilt:

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

Aus $n \in \mathbb{P}$ folgt $a \in \mathbb{Z}_n^*$. Also ist a ein MR-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 MR-Lügner bzw. Fermat-Lügner wie folgt definiert.

$$\text{MRL}_n \stackrel{\text{df.}}{=} \{a \in \mathbb{N} \mid 1 \leq a < n \text{ und } a \text{ ist MR-Lügner für } n\}$$

$$\text{FL}_n \stackrel{\text{df.}}{=} \{a \in \mathbb{N} \mid 1 \leq a < n \text{ und } a \text{ ist Fermat-Lügner für } n\}$$

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

Beweis

$$(\exists x+4) = 7 \cup$$


□

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

$$(\exists x+4) = 7 \cup$$


□

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.

$$z(n) \stackrel{\text{df}}{=} \max\{j \in \{0, \dots, k\} \mid \exists a \in \{1, \dots, n-1\}, a^{2^{j \cdot m}} \equiv -1 \pmod{n}\}$$

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

Weiter gilt $z(n) \neq k$, denn andernfalls existiert ein $a \in \{1, \dots, n-1\}$ mit $a^{n-1} \equiv a^{2^k \cdot m} \equiv -1 \pmod{n}$, also $a \in \mathbb{Z}_n^*$ und $a^{n-1} \equiv -1 \pmod{n}$, 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.

$$\text{MRL}'_n \stackrel{\text{df}}{=} \{a \in \{1, \dots, n-1\} \mid a^{2^{z(n)} \cdot m} \equiv \pm 1 \pmod{n}\}$$

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

Beweis

$$(\exists x+4) = 7 \cup$$


□

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

Beweis

$$(\exists x+4) = 7 \cup$$


□

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

Beweis

$$(\exists x+4) = 7 \cup$$


□

Lemma 8.16 Ist n eine Carmichael-Zahl so gilt

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

Beweis Nach 8.15 ist MRL'_n eine echte Untergruppe von \mathbb{Z}_n^* .

Nach dem Satz von Lagrange ist $|\text{MRL}'_n|$ echter Teiler von $|\mathbb{Z}_n^*|$.

Also $|\text{MRL}_n| \leq |\text{MRL}'_n| \leq |\mathbb{Z}_n^*|/2 \leq n-1/2$. Demnach wählen wir in Zeile 2 von MillerRabin(n) mit Wkt. $\leq 1/2$ ein $a \in \text{MRL}_n$.

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

□

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

$$n \in \mathbb{P} \implies P(\text{MillerRabin}(n) = 1) = 1$$

$$n \notin \mathbb{P} \implies P(\text{MillerRabin}(n) = 1) \leq 1/2$$

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 prob. Polynomialzeit arbeitet. \square

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

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

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

$$\begin{aligned} n \in \mathbb{P} &\implies P(M(n) = 0) = 0 \\ n \in \overline{\mathbb{P}} &\implies P(M(n) = 0) \geq 1/2 \end{aligned}$$

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

$$\begin{aligned} n \in \mathbb{P} &\implies P(M(n) = 1) = 0 \\ n \in \overline{\mathbb{P}} &\implies P(M(n) = 1) \geq 1/2 \end{aligned}$$

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

Index

3-FARB, 132
3-Färbarkeitsproblem, 132

AES, 35
Alphabet, 6
Authentizität, 104
Autokey-Verschlüsselung, 18

Betriebsmodus, 40
Blockchiffre, 40

Cäsar-Chiffre, 13
Carmichael-Zahl, 167
CBC, 43
Challenge-Response-Identifikation, 120
Chiffertext, 6
chinesischer Restsatz, 59
Cipher Block Chaining Mode, 43

Diffie-Hellman-Schlüsselaustausch, 74
digitale Signatur, 103

ECB, 41
Einheitengruppe von \mathbb{Z}_n , 57
Einmal-Passwörter, 119
Einmal-Signatur, 116

Einwegeigenschaft, 100
Einwegfunktion, 106
längentreue, 114
Electronic Code Book Mode, 41
Elgamal-Kryptosystem, 85, 87
Enigma, 23
Entschlüsselungsalgorithmus, 6
Ereignis, 46
Erfolgswahrscheinlichkeit, 3
Ergebnismenge, 46
Erzeuger, 75
Euklidischer Algorithmus, 67
erweiterter, 68
Euler-Kriterium, 93
Eulersche φ -Funktion, 57

Fermat-Lügner für n, 167
Fermat-Zeuge für n, 167
Fiat-Shamir-Protokoll, 122
 FL_n , 174

GI, 129
Goldwasser-Micali-Kryptosystem, 90, 92, 147
Graphisomorphieproblem, 129
Gruppe, 29
kommutativ, 29

- Hashfunktion, 105
- kryptographische, 108
- Identifikation, 117
- Integrität, 104
- irreduzibel, 31
- Körper, 30
 - endlich, 30
- Klartext, 6
- kleiner Fermatscher Satz, 59
- Kollision, 107
- kollisionsresistent, 107
- Kryptoanalyse
 - Chosen Ciphertext, 10
 - Chosen Plaintext, 10
 - Ciphertext only, 10
 - Known plaintext, 10
- Kryptosystem, 6
 - asymmetrisch, 9
 - homomorph, 155
 - Private-Key, 9
 - Public-Key, 9, 56
 - symmetrisch, 9
 - vollständig homomorph, 156
- Legendre-Symbol, 93
- Man-in-the-middle-Angriffe, 84
- Miller-Rabin-Primzahltest, 66, 165, 172
- Millionärsproblem, 144
- MR-Lügner für n , 169
- MR-Zeuge für n , 169
- MRL_n , 174
- multiplikative Gruppe der Ordnung $\varphi(n)$, 57
- Nullstellen, 57
- One-Time-Pad, 19
- Ordnung von x , 59
- Padded RSA, 73
- perfekt sicher, 50
- perfekt sicher bezüglich einer Verteilung, 50
- perfekte Sicherheit, 45
- Plain RSA, 72
- Polynomring, 31
- Primivwurzel, 75
- Primzahltest, 66
- probabilistischer Algorithmus, 2
- quadratischer Nichtrest modulo n , 90
- quadratischer Rest modulo n , 90
- Quadratwurzel von 1 modulo n , 168
- randomisierte Polynomialzeit, 166
- RSA-Kryptosystem, 60
- Satz von

Bayes, 47
Shannon, 51
Satz von Euler, 58
Satz von Lagrange, 59
Schlüssel
 öffentlicher, 9, 56
 privater, 9, 56
Schlüsselaustauschproblem, 55
Schlüsselgenerator, 6
Schlüsselpaar, 6
Secret Sharing, 140
Secret-Sharing-Protokoll, 140
Secure Circuit Evaluation, 147
Secure Multiparty Computation, 138
semantisch sicher, 101
Sicherheitsparameter, 115
Sophie-Germain-Primzahl, 80
Square and Multiply, 69
Substitutionschiffre, 14

unabhängig, 46

Verbindlichkeit, 104
Vernam, 52
Vernam-Chiffre, 20
Verschiebungschiffe, 13
Verschlüsselungsalgorithmus, 6
Verteilung, 46
Vigenère-Chiffre, 17

Wahrscheinlichkeit