# Proseminar/Seminar Kryptographie
# SS 2000

*Leiter:* Prof. Dr. K. Madlener
*Betreuer:* Robert Eschbach, Christoph Kögl
*Studenten:* Lidia Angelova, Klaus Dräger, Özgür Göceri, Christian Mathis, Frank Michel, Dirk Niebuhr, Christoph Thelen, Wei Zheng

AG Grundlagen der Informatik, FB Informatik, Universität Kaiserslautern, 67653 Kaiserslautern
*E-mail address*: {eschbach|koegl}@informatik.uni-kl.de

ZUSAMMENFASSUNG. Dieses Dokument enthält die Ausarbeitungen der Teilnehmer des Proseminars/Seminars Kryptographie, welches im SS 2000 von der AG 'Grundlagen der Informatik' unter der Leitung von Prof. Dr. K. Madlener durchgeführt wurde.
**Stand: 26. Oktober 2000**

# Inhaltsverzeichnis

KAPITEL 1

# Einführung

## 1. Thematik

Angesichts der immer weiter zunehmenden Vernetzung mit Computern erhält die Informationssicherheit und damit die Kryptographie eine immer größere Bedeutung. Die Entwicklung und Analyse von kryptographischen Protokollen wird ein immer wichtigerer Zweig der modernen Kryptographie.
In dem Proseminar/Seminar 'Kryptographie', welches im SS 2000 von der AG 'Grundlagen der Informatik' unter der Leitung von Prof. Dr. K. Madlener durchgeführt wurde, wurden grundlegende Protokolle (Diffie-Hellmann, ElGamal) und die so genannten Zero-Knowledge-Protokolle betrachtet, also Protokolle, die einen Anderen von der Existenz eines Geheimnisses überzeugen, ohne ihm dabei das Geringste zu verraten. Auch aktuelle Anwendungen wie elektronisches Geld, elektronische Wahlverfahren und Anonymität waren Gegenstand dieses Proseminars/Seminars.

## 2. Vortragsthemen

### 2.1. Monoalphabetische Chiffrierung: Substitution und Transposition.

(1) Literatur: [**Bau97**]: Kapitel 3, 4 und 6
(2) Referent: Dirk Niebuhr

### 2.2. Polyalphabetische Chiffrierung.

(1) Literatur: [**Bau97**]: Kapitel 7 und 8
(2) Referent: Özgür Göceri

### 2.3. Kryptanalyse: mono- und polyalphabetischer Verfahren.

(1) Literatur: [**Bau97**]: Kapitel 12,13,14
(2) Referent: Frank Michel

### 2.4. Kryptologische Grundlagen und grundlegende Protokolle.

(1) Literatur: [**BSW98**]: Kapitel 2 und 3, [**Sch96**]: Kapitel 3
(2) Referent: Lidia Angelova

### 2.5. Zero-Knowledge-Verfahren.

(1) Literatur: [**BSW98**]: Kapitel 4, [**Weg96**]: S. 287-304 (Beitrag, Uwe Schöning), [**Sch96**]: Kapitel 5
(2) Referent: Christian Mathis

### 2.6. Multi-Party Computation und Anonymität.

(1) Literatur: [**BSW98**]: Kapitel 5 und 6, [**Sch96**]: Kapitel 6
(2) Referent: Christoph Thelen

### 2.7. Oblivious Transfer.

(1) Literatur: [**BSW98**]: Kapitel 7, [**Sch96**]: Kapitel 5
(2) Referent: Wei Zheng

### 2.8. Kryptanalyse des Merkle-Hellman-Rucksacksystems.

(1) Literatur: [**Vau98**], [**Kux99**], [**SH95**], [**Bri83**], [**Bri84**]
(2) Referent: Klaus Dräger

KAPITEL 2

# Monoalphabetische Chiffrierung

**Dirk Niebuhr**

## 1. Einführung

Es folgt eine Erläuterung einiger simpler, monalphabetischer Chiffrierverfahren nach Bauer [**Bau97**].

## 2. Definitionen

**Kryptologie:** Als Kryptologie wird die Wissenschaft von den offenen Geheimschriften (Kryptographie), von ihrer unbefugten Entzifferung (Kryptanalyse) sowie den Vorschriften, die dazu dienen sollen, die unbefugte Entzifferung zu erschweren (Chiffriersicherheit) bezeichnet.

**Klartextvokabular/Klartextzeichenvorrat $V$:** Hiermit wird der „Klartext“ (also der unverschlüsselte Text) formuliert.

**Geheimtextvokabular/Geheimtextzeichenvorrat $W$:** Hiermit wird der Geheimtext formuliert. Kurz: Chiffre/Code, einzelne Zeichen aus $W$ können auch sogenannte Sigel sein (= Sonderzeichen).

**$V^*, W^*$:** Menge aller Wörter über $V, W$ (Klartextwort, Geheimtextwort).

**$\varepsilon$:** Leeres Wort.

**$Z^n$:** Menge aller Wörter der Länge $n$ über $Z$.

**$Z^{(n)}$:** $\{\varepsilon\} \cup Z^1 \cup Z^2 \cup \ldots \cup Z^n$.

**Chiffrierung:** Relation $X : V^* \to W^*$.

**Dechiffrierung:** Konverse Relation $X^{-1} : y \mapsto x$ gdw. $X : x \mapsto y$
Injektivität einer Chiffrierung ist von Vorteil, damit der Empfänger den Klartext eindeutig zurückgewinnen kann.
(*Injektivität:* $(x \mapsto z) \wedge (y \mapsto z) \Longrightarrow x = y$).

**Homophoner Text:** Geheimtextworte, die in der Relation dem selben Klartextwort zugeordnet sind. (Im funktionalen Fall nicht vorhanden, Chiffrierung *deterministisch*, eindeutige Abbildung von Klartext- in Geheimtextbereich. In der Regel gilt: $\varepsilon \mapsto \varepsilon$. Wird $\varepsilon$ auch auf andere Wörter abgebildet, so heißen diese Wörter „Blendtexte“.)

**Endliche Chiffrierung:** Die Menge aller in Relation stehenden Paare ist eine endliche Menge, für geeignete $n, m$ gilt: $X : V^{(n)} \to W^{(m)}$.

## 3. Angabe der Relation

**Chiffrierschritt-System $M$:** Eine nichtleere, in der Regel endliche Menge

$$\{\chi_0, \chi_1, \chi_2, \ldots, \chi_{\theta-1}\}$$

von (injektiven) Relationen $\chi_i : V_i^{(n)} \to W_i^{(m)}$. Jedes $\chi_i$ heißt Chiffrierschritt.

**Endlich erzeugte Chiffrierung $X = [\chi_{i_1}, \chi_{i_2}, \chi_{i_3}, \ldots]$:** wird durch (abbrechende oder unendliche) Folge $(\chi_{i_1}, \chi_{i_2}, \chi_{i_3}, \ldots)$ von Chiffrierschritten durch Konkatenation induziert, d.h. $x \mapsto y$ genau dann, wenn

$$x = x_1 \bigstar x_2 \bigstar x_3 \bigstar \ldots \bigstar x_k \text{ und } y = y_1 \bigstar y_2 \bigstar y_3 \bigstar \ldots \bigstar y_k$$

(wobei jedes Wort aus $V^*$ durch Anfügung von Zeichen geeignet aufgefüllt wird).

Beispiel: $V_i^{(n_i)} \to V_i^{(n_i)}$ zyklische Transposition von $n_i$ Elementen:
$n_1 = 3,\ n_2 = 5,\ n_0 = 2,\ n_3 = 6$

| e s w | a r s c h | o n | d u n k e l | | e s | w a r | s c h o n | d u n k e l |
|---|---|---|---|---|---|---|---|---|
| s w e | r s c h a | n o | u n k e l d | | s e | a r w | c h o n s | u n k e l d |
| $(\chi_1, \chi_2, \chi_0, \chi_3)$ | | | | | | | | $(\chi_0, \chi_1, \chi_2, \chi_3)$ |

**Kardinalität $\theta = |M|$:** Anzahl der Chiffrierschritte im Chiffrierschritt-System.

**Erzeugende Relation:** Ein Chiffrierschritt $\chi_i : V^{(n_i)} \to V^{(m_i)}$. $n_i$: (maximale) Chiffrierbreite, $m_i$: (maximale) Chiffratbreite.
*Achtung:* Relation $\chi_i$ kann nichtdeterministisch sein.

**Endomorph:** Ein Chiffrierschritt-System und die Chiffrierung heißen endomorph, falls $V = W$.

**Gespreizter Chiffrierschritt:** Geheimtextbereich enthält Worte verschiedener Länge.
Beispiel für nicht gegebene Injektivität: $a \mapsto \cdot -$; $i \mapsto \cdot\cdot$; $l \mapsto \cdot - \cdot\cdot$;
($V^1 \to W^{(4)}$ also injektiv), aber: in $V^* \to W^*$ : $ai \mapsto \cdot - \cdot\cdot$ und $l \mapsto \cdot - \cdot\cdot$!

**Codebuch, Satzbuch:** Auflistung eines Chiffrierschritts = Codierschritt.

**Chiffre, Code:** Elemente von $W_i^{(m)}$.

**Monoalphabetische Chiffrierung:** Nur ein einziger Chiffrierschritt, ansonsten polyalphabetisch.

**Monographische Chiffrierung:** $n_i = 1\ \forall i$, sonst polygraphisch.

**Blockchiffrierung:** Alle Chiffrierschritte aus $M$ sind von gleichem $n$ und gleichem $m$ und für alle $x_i$ aus $M$ gilt: $x_i : V^n \to W^m$ (Chiffrierschritte ungespreizt!).

**Block:** Wort aus $V^n$.

**Blockchiffrierung:** Im engeren Sinn: $V = W$ und $m = n$.

**Polyphonie:** Ein Geheimtextwort steht für mehrere Klartextworte (die möglichst keine ähnliche Bedeutung haben sollten ... — „illegitime“ polyphone Texte sollen keinen Sinn ergeben ... ). Erschwert die unbefugte Entzifferung. Auch die Unterdrückung von Wortzwischenräumen und Satzzeichen sowie Nichtunterscheidung von Groß- und Kleinschreibung verursacht Polyphonie:
„ernsthafte reformen“ $\Longleftrightarrow$ „ernsthaftere formen“
„Er beschloss, nicht in den Wald zu gehen“ $\Longleftrightarrow$ „Er beschloss nicht, in den Wald zu gehen“
„die gefangenen Fliegen“ $\Longleftrightarrow$ „die Gefangenen fliegen“
$N = |W|,\ N = 1$ uninteressant $\Longrightarrow N \geq 2$

**Alphabet:** Linear geordneter Zeichenvorrat, normalerweise (ab 1900)
$Z_{26}$ = {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}, aber:
Auf Hawaii gespr. Sprache verwendet $Z_{12}$ = {a,u,i,o,e,w,h,k,l,m,n,p}
(1536 (Porta): $Z_{20} = Z_{26}\backslash$ {j,k,u,w,x,y})
$Z_{10} = \{0, 1, 2, \ldots, 9\}$
$Z_6$ = {A,D,F,G,V,X} (da entspr. Morsezeichen $\cdot -, - \cdot\cdot, \cdot\cdot -\cdot, - - \cdot, \cdot\cdot\cdot -, - \cdot\cdot -$ gut unterscheidbar)
$Z_2$ = {0,L} (binär)

**Klartextzeichen:** Kleinbuchstaben

**Geheimtextzeichen:** GROßBUCHSTABEN

**Schlüssel:** Dienen der Auswahl der Chiffrierschritte aus einem Chiffrierschritt-System $M$, sowie zur Bildung von Chiffrierschritten. Schlüssel erlauben es, die Chiffrierung nach verabredeten Regeln zu wechseln (jeden Tag/jede Woche ... )

## 4. Transposition

Die Transposition wechselt nicht Alphabetzeichen aus, sondern permutiert lediglich ihre Plätze.

Einfachste Verfahren:

**Krebs:** Wortweise oder die ganze Nachricht rückwärts lesen:
April → Lirpa

**Ananyme:** (basierend auf gleicher Aussprache): Remarque $\leftrightarrow$ Kramer, (Ave $\leftrightarrow$ Eva)

**Palindrome:** Invariant zum Krebs, können also von vorne und von hinten gelesen werden (z.B.: Anna, Ein Neger mit Gazelle zagt im Regen nie).

**Schüttelreim:** Schwarzen Wein $\leftrightarrow$ Warzenschwein,
reine Sache $\leftrightarrow$ seine Rache

**Würfel:** Zeilenlänge wird gewählt und dann die Nachricht in Zeilen in den Würfel geschrieben und entlang der Spalten ausgelesen:

| | | | | | | |
|---|---|---|---|---|---|---|
| i | c | h | b | i | n | $\Longrightarrow$ IDTECEONHRRBBDEAIOIRNKST |
| d | e | r | d | o | k | Diagonalwürfel: ablesen entlang der Diagonalen: |
| t | o | r | e | i | s | $\Longrightarrow$ ETNDOBIERACRERHDITBOSIKN |
| e | n | b | a | r | t | |

weitere Variante: furchenwendig, aber in Spalten lesen(Schlangenlinienwürfel):
$\Longrightarrow$ IDTENOECHRRBAEDBIOIRTSKN

oder: Ablesen in einer Spirale (Schneckenwürfel):
$\Longrightarrow$ TSKNIBHCIDTENBARIODREORE

**Rösselsprungwürfel:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 4 | 53 | 18 | 55 | 6 | 43 | 20 |
| 52 | 17 | 2 | 5 | 38 | 19 | 56 | 7 |
| 3 | 64 | 15 | 54 | 31 | 42 | 21 | 44 |
| 16 | 51 | 28 | 39 | 34 | 37 | 8 | 57 |
| 63 | 14 | 35 | 32 | 41 | 30 | 45 | 22 |
| 50 | 27 | 40 | 29 | 36 | 33 | 58 | 9 |
| 13 | 62 | 25 | 48 | 11 | 60 | 23 | 46 |
| 26 | 49 | 12 | 61 | 24 | 47 | 10 | 59 |

(entsteht aufgrund der Bewegungsmöglichkeiten eines Pferdes beim Schach)

**Weitere Muster zur Transposition:** ($Z^{25}$):

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | | | | e | | | | i | | | | n | | | | r | | | | v | | | | z |
| | b | | | | d | | | | f | | | | h | | | | k | | | | m | | | |
| | | | o | | | | q | | | | s | | | | u | | | | w | | | | y | |
| | | c | | | | | | g | | | | | | l | | | | | | p | | | | |
| | | | t | | | | | | x | | | | | | | | | | | | | | | |

Zeilenweises Ablesen ergibt: AEINRVZBDFHKMOQSUWYCGLPTX

oder ($Z^{24}$):

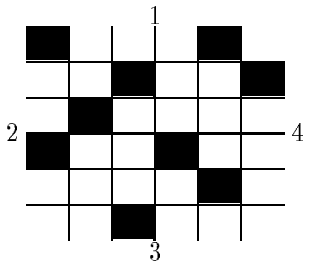
| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | b | | | f | | | k | | | o | | | s | | | w | | |
| a | | c | e | | | g | i | | | l | n | | | p | r | | | t | v | | | x | |
| | d | | | h | | | m | | | q | | | u | | | z | | |

Zeilenweise abgelesen: BFKOSWACEGILNPRTVXDHMQUZ

**Drehraster:** Verschiedene Fenster werden durch Drehung nacheinander sichtbar. Es gibt 2-zählige und 4-zählige Drehraster. Beispiel für ein 4-zähliges Drehraster:

| | | | | | |
|---|---|---|---|---|---|
| n | l | l | d | u | d |
| l | m | r | r | e | v |
| e | i | a | s | h | n |
| e | l | e | r | k | e |
| a | n | s | n | m | u |
| a | n | a | n | n | a |

Dechiffriert mit Muster:




$\Longrightarrow$ nurviermaldrehenundmankannalleslesen.

**Konstruktion** eines Drehrasters: Man nimmt ein Feld des Würfels, schreibt eine Zahl hinein, und schreibt dann in alle Felder, auf die dieses Feld rotiert wird die gleiche Zahl. Dann nimmt man das nächste Feld, welches noch nicht nummeriert ist und geht genauso vor, bis alle Felder nummeriert sind. Im Fall des 4-zähligen Rasters hat man dann jeweils viermal die gleiche Zahl. Nun sucht man sich von jeder 4er-Gruppe immer genau ein Feld aus und markiert es und erhält so das Raster (nimmt man das einer Zahl entsprechende Feld überhaupt nicht, kann man hier Blender einsetzen ... )!

Das Verfahren der Transposition allein stellt keine große Hürde für den unbefugten Entzifferer dar, wird aber wirksamer, wenn es mit Substitution kombiniert wird!

**Spalten-Transposition:** Die Nachricht wird in Zeilen der gewählten Länge geschrieben, die Spalten werden dann gemäß einer Permutation $\pi$ (hier $\pi$: 2 1 4 3) vertauscht:

eswarschondunkel

| 2 | 1 | 4 | 3 | | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|---|---|---|---|
| e | s | w | a | | S | E | A | W | |
| r | s | c | h | $\Longrightarrow$ | S | R | H | C | $\pi$ :2 1 4 3 |
| o | n | d | u | | N | O | U | D | |
| n | k | e | l | | K | N | L | E | |

Spaltenweise abgelesen: SSNKERONAHULWCDE

ebenfalls möglich: Ablesen in Zeilen(Blocktransposition genannt), ergibt: SEAWSRHCNOUDKNLE

**gemischte Zeilen-Spalten-Transposition:** Die Nachricht wird wie im vorangegangenen Beispiel in einen Würfel geschrieben, dann werden gemäß einer 1. Permutation $\pi_1$ die Zeilen vertauscht, anschließend gemäß einer 2. Permutation $\pi$ die Spalten vertauscht und schließlich spaltenweise abgelesen. (Wenn zeilenweise abgelesen wird, bezeichnet man es als „gemischte Zeilen-Block-Transposition“) Ein Vertauschen der Permutationen $\pi_1$ und $\pi$ ändert nichts!

**Doppelte Spalten-Transposition:** Auf das Ergebnis der Spalten-Transposition wird erneut Spalten-Transposition angewandt und zwar entweder mit der gleichen Permutation (auch Losung genannt) oder (besser) mit einer neuen Losung.

**Verfahren zur Gewinnung einer Losung:** Man merkt sich ein Kennwort und nummeriert dann die Buchstaben in der Reihenfolge des Auftretens im Alphabet durch (doppelte Buchstaben werden dabei der Reihenfolge nach nummeriert).

Beispiel:

| M | A | C | B | E | T | H |
|---|---|---|---|---|---|---|
| 6 | 1 | 3 | 2 | 4 | 7 | 5 |

$\Longrightarrow \pi$: 6 1 3 2 4 7 5

**Anagramme:** Beim Anagramm versucht man aus dem Zeichenhaufen die Nachricht zu rekonstruieren:
Huyghens schrieb:

$a^7c^5d^1e^5g^1h^1i^7l^4m^2n^9o^4p^2q^1r^2s^1t^5u^5$ kann interpretiert werden als:

$\Longrightarrow$ annulo cingitur tenui plano, nusquam cohaerente, ad eclipticam inclinato
([Saturn] ist umgeben von einem dünnen flachen Ring, der ihn nirgendwo berührt und zur Ekliptik geneigt ist)

Anagramme finden sich ebenfalls in Berufsrätseln bekannt aus Zeitschriften, z.B.: Emil Rest, Gera $\Longrightarrow$ Lagermeister.

Aber: Aus einem Zeichenhaufen können mehrere Nachrichten gewonnen werden, z.B.: permission $\leftrightarrow$ impression, denotation $\leftrightarrow$ detonation

## 5. Chiffrierschritte: Einfache Substitution

**Chiffrierschritte $\chi_i : V^{(1)} \rightarrow W^{(m_i)}$:** (Chiffrierschritte monographisch!).
Im Monoalphabetischen Fall wird ein beliebiges $\chi_s$ ausgewählt und mit der Chiffrierung $X = [\chi_s, \chi_s, \chi_s, \ldots]$ gearbeitet (Chiffrierschrittsystem $M$ also einelementig).
Zuerst: $m_i = 1 \; \forall i$

**$V^{(1)} \rightarrow W^{(1)}$(ohne Homophone und Blender):** Meist wird für $W$ ein Alphabet seltsam geformter, unüblicher Zeichen verwendet.

**$V^{(1)} \rightarrow W^{(1)}$(mit Homophonen und Blendern):** Zusätzliche Zeichen werden eingeführt, um z.B. häufig vorkommende Buchstaben auf unterschiedliche Zeichen (Homophone) abbilden zu können - Blender werden zusätzlich verwendet, um ein Erkennen von Wortmustern zu erschweren.

**Spezialfälle: $V \leftrightarrow W$(eineindeutig):** „umgeordnetes“ Alphabet $W$ von $N$ Klartextzeichen, das einem Standard-Alphabet $V$ von $N$ Geheimtextzeichen gegenübersteht (realisierbar über Umstecken von Leitungsverbindungen).
Paarungen von Klartext- und Geheimtextzeichen müssen notiert werden.
$\Longrightarrow$ für Chiffrierer günstiger, wenn Klartextzeichen alphabetisch geordnet, für Dechiffrierer Geheimtextzeichen ...

## 6. Permutation

Eineindeutige Substitution $V \leftrightarrow V$ heißt Permutation. Für Permutation wird auch Zyklenschreibweise verwendet.
Beispiel: (a b c d) (e g) (f), d.h.: a $\mapsto$ B, b $\mapsto$ C, ..., d $\mapsto$ A, ..., f $\mapsto$ F
Beim Chiffrierschritt geht man jeweils zum nachfolgenden Buchstaben, vom letzten des Zyklus wieder zum ersten - Einerzyklen brauchen nicht notiert zu werden! - Dechiffrierung erfordert denselben Schritt rückwärts!

Naheliegende Abbildungen hierzu:

$V \leftrightarrow V$ : $\updownarrow$ 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

$V \leftrightarrow V$ : $\updownarrow$ k r y p t o l g i e a b c / z x w v u s q n m j h f d

(„furchenwendig mit Merkwort (Kryptologie)“)

Dies ist eine „involutorische“ Abbildung (Chiffrierschritt = Dechiffrierschritt, in diesem Fall gegeben, da ausschließlich 2er-Zyklen vorhanden sind!).

„Echt“ involutorisch: keine Einerzyklen
Angewandt auf binäres Alphabet: einzige (nichttriviale) Permutation ist echt involutorisch:

$V \leftrightarrow V$ : $\updownarrow$ 0 / L bzw. geläufiger: $V \leftrightarrow V$ : $\updownarrow$ 0 / 1 entspricht 'xor'-Operation mit 1

**Voll zyklische Permutation:** genau ein Zyklus
(z.B.: (a b c d e f g h i l m n o p q r s t v x) (Def. auf $Z_20$) oder
(a d g l o r v b e h m p s x c f i n q t) - ebenfalls auf $Z_20$) die 2. („Verschiebung um 3 Buchstaben“) soll Julius Caesar angeblich verwendet haben, sein Nachfolger Augustus die 1 ...

**permutiertes Alphabet:** Auch für nicht-involutorische, nicht zyklisches $V \leftrightarrow V$ wird das Alphabet in Substitutionsschreibweise notiert:

$V \leftrightarrow V$ :

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

Weitere permutierte Alphabete können durch (zyklische) Verschiebung einer der beiden Zeilen gewonnen werden („verschobene“ permutierte Alphabete).

**Potenziertes, permutiertes Alphabet:** Wird durch Mehrfachausführung des Chiffrierschritts erhalten, im obigen Beispiel ergeben sich für die 2. Potenz die Zyklen:
(a n y w q f)(b r g o)(c)(d p)(e m t j)(h x v l i s)(k u)(z),
in Substitutionsschreibweise:

$V \leftrightarrow V$ :

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

Durch Potenzierung oder Verschiebung erhält man jeweils bis zu $N$ Alphabete, die sich jedoch nicht entsprechen (müssen).

Konstruktion wird „sicherer“, wenn die beiden Alphabete unterschiedlich notiert werden:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | E | C | U | R | I | T | Y | a | e | i | l | o | r | u | x |
| A | B | D | F | G | H | J | K | b | f | j | m | p | s | v | y |
| L | M | N | O | P | Q | V | W | c | g | k | n | q | t | w | z |
| X | Z | | | | | | | d | h | | | | | | |

$\Longrightarrow$

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

Weiteres Verfahren: Klartextbuchstaben in Reihenfolge der Schlüsselwortbuchstaben in die Spalten füllen:
Schlüsselwortmethode kann auch zur Gewinnung von Zyklen genutzt werden:
(s e c u r i t y a b d f g h j k l m n o p q v w x y), je mehr verschiedene Buchstaben das Schlüsselwort beinhaltet, desto besser.

**Technische Realisierungen von Substitutionen:**
Klar- und Geheimtextzeichen auf 2 Linealen gegenüberstellen (auf einem Lineal muß das Alphabet wiederholt werden). Andere Möglichkeit: Chiffrierscheibe: eine Scheibe, auf der das Klartextalphabet innen im Kreis und das Geheimtextalphabet außen im Kreis notiert war. Diese beiden Kreisteile konnten gegeneinander verschoben werden (verschieben/verdrehen führt zu verschobenen Alphabeten).

**Bipartite einfache Substitution:** $M = 2$, also: $V^{(1)} \to W^{(2)}$

**Quinärchiffrierung:**

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | a | b | c | d | e |
| 2 | f | g | h | i | k |
| 3 | l | m | n | o | p |
| 4 | q | r | s | t | u |
| 5 | v | w | x | y | z |

oder

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | a | f | l | q | v |
| 2 | b | g | m | r | w |
| 3 | c | h | n | s | x |
| 4 | d | i | o | t | y |
| 5 | e | k | p | u | z |

c entspricht nach linker Vorschrift „13“, nach rechter „31“. Diese Chiffrierung $Z_{25} \to Z_5 \times Z_5$ wird in Haftanstalten bis heute verwendet und „internationaler Klopf-Code“ genannt.

Einführung von Homophonen, um das Entschlüsseln schwieriger zu machen:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Häufigkeit |
|---|---|---|---|---|---|---|---|---|---|---|
| 4,5,6,7,8,9,0 | e | t | a | o | n | i | r | s | h | 71,09% |
| 2,3 | b | c | d | f | g | j | k | l | m | 19,46% |
| 1 | p | q | u | v | w | x | y | z | | 9,45% |

Analog kann man sich tripartite, quinpartite oder gar oktopatite Substitution vorstellen.

**$V^{(1)} \to W^{(m)}$: Spreizen:** Als Beispiel: $m = 2$, also $V \to W^1 \cup W^2$:
Spionage-Chiffren:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| | s | i | o | e | r | a | t | n | | |
| 8 | c | x | u | d | j | p | z | b | k | q |
| 9 | . | w | f | l | / | g | m | y | h | v |

**Einelementige Chiffren:** Erste Zeile ; $Z_{28}$ : 2 Sonderzeichen: „." für Stop, / für „Buchstaben-Ziffernwechsel“ (nicht weiter betrachtet: hängt mit anschließend vorgenommener „Überchiffrierung“ zusammen!). Möglichkeiten zur Gewinnung weiterer Chiffren: Kennworte, andere Nummerierung (aber beachten: „rechts oben“ und in der linken Spalte gleiche Ziffern, ansonsten kommt es zu Polyphonie!)

Weiteres Beispiel:
1.Kennwort: „subway“, 2. Kennsatz zur „Nummerierung“: „a sin to err“
$\Longrightarrow$

| s | u | b | w | a | y |
|---|---|---|---|---|---|
| 0 | 82 | 87 | 91 | 5 | 97 |
| c | d | e | f | g | h |
| 80 | 83 | 3 | 92 | 95 | 98 |
| i | j | k | l | m | n |
| 1 | 84 | 88 | 93 | 96 | 7 |
| o | p | q | r | t | v |
| 2 | 85 | 89 | 4 | 6 | 99 |
| x | z | . | / | | |
| 81 | 86 | 90 | 94 | | |

Erläuterung zur Nummerierung:
Es wird spaltenweise nummeriert, und zwar die Zeichen des 2. Kennsatzes nach Reihenfolge des Auftretens mit 0-7, sämtliche anderen Zeichen mit 80-99.

## 7. Polygraphische Substitution und Codierung

**Polygraphische Substitution: $V^{(n)} \to W^{(m)}$ mit $n > 1$**

**Bigramm-Substitutionen:** $V^2 \to W^{(m)}$
Zur Darstellung wird meist eine Matrix verwendet.
**Bigramm Permutation:** $V^2 \to V^2$
Beispiel: involutorische Bigrammpermutation

| | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | XZ | KJ | YJ | HP | PL | EL | VB | CI | DW | XN | ZL | YP | VN | HH | CC | ... |
| b | LP | QT | HE | RS | UR | CR | ZH | GV | WC | HL | YN | KT | WT | MC | KH | ... |
| c | DX | MN | AO | NH | SF | GI | WL | MN | AH | GR | BZ | HS | ZU | YM | WU | ... |
| d | KM | YZ | RY | FP | TR | CR | XE | JK | NY | PO | GJ | JR | PE | MO | VB | ... |
| e | QU | HP | QG | JQ | YQ | OB | SA | NL | PX | OP | VS | AF | XK | XR | UQ | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋱ |

Involutorisch: kann man eigentlich nur anhand der kompletten Tabelle zeigen, lässt sich aber anhand dieses Ausschnitts bereits erahnen, denn:
(ah $\to$ CI, ci $\to$ AH), außerdem (ao $\to$ CC, cc $\to$ AO) und (af $\to$ EL, el $\to$ AO)

Weitere Chiffren können wieder durch Kennworte erhalten werden, die in der 1. Zeile und 1. Spalte eingetragen werden. (sind dann aber nicht mehr involutorisch, für den Dechiffrierer muß also eine 2. Matrix hergestellt werden, oder er muß „mühsam“ dechiffrieren).

Ideale Matrix: in jeder Spalte und Zeile kommt jeder Buchstabe genau einmal als erster und einmal als letzter Buchstabe vor (möglich für alle $N > 2$ außer 6):

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AB | BC | CA | | AC | BA | CB | DD | | AA | BB | CC | DD | EE |
| CC | AA | BB | oder | BD | AB | DA | CC | oder | BC | CD | DE | EA | AB |
| BA | CB | AC | | DB | CD | BC | AA | | CD | DA | EB | AC | BD |
| | | | | CA | DC | AD | BB | | DB | EC | AD | BE | CA |
| | | | | | | | | | ED | AE | BA | CB | DC |

Ungeeignete Matrix:

| | | | | |
|---|---|---|---|---|
| AA | AB | AC | AD | ... |
| BA | BB | BC | BD | ... |
| CA | CB | CC | CD | ... |
| DA | DB | DC | DD | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋱ |

z.B.:

| | b | a | d | c |
|---|---|---|---|---|
| d | AA | AB | AC | AD |
| c | BA | BB | BC | BD |
| b | CA | CB | CC | CD |
| a | DA | DB | DC | DD |

Ungeeignet, da:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ab | ca | dd | aa | ac | cc | da | db | bb | bc | ↓ |
| DA | BB | AC | DB | DD | BD | AB | AA | CA | CD | |

Entspricht einer monographischen Chiffrierung nach 2 Alphabeten (abwechselnd), denn z.B. a $\mapsto$ D, wenn a „1. Zeichen“, a $\mapsto$ B, wenn a „2. Zeichen“ usw.

**Tripartite Bigrammsubstitution:** $V^2 \to W^3$

Beispiel:

| | a | b | c | d | ... |
|---|---|---|---|---|---|
| a | 148 | 287 | 089 | 623 | ... |
| b | 243 | 127 | 500 | 321 | ... |
| c | 044 | 237 | 174 | 520 | ... |
| d | 143 | 537 | 257 | 347 | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋱ |

Dies ist möglich, da $262 = 676 < 1000$. Die unbenutzten 224 Zahlen können für Homophone und Blender genutzt werden.

**Spezielle bipartite Bigrammsubstitution „Playfair“**, erfunden von Charles Wheatstone 1854, Playfair empfahl sie dem Militär und hohen Regierungsstellen:
Ein aus einem Kennwort gewonnenes permutiertes Alphabet ($Z^{25}$) wurde in ein $Z^5 \times Z^5$ eingeschrieben:

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

(1) Beide Buchstaben des Bigramms in derselben Zeile/Spalte: jeweils der rechte/untere Nachbar wird als Chiffrat gewählt:
am $\mapsto$ LE, bzw. oq $\mapsto$ FY

(2) Ist das nicht der Fall nimmt man statt des ersten Buchstabens den in der selben Zeile liegenden, aber in der Zeile des 2. Buchstabens und umgekehrt:
ag $\mapsto$ EC

(3) Um Bigramme mit Doppelzeichen zu vermeiden, wird jeweils ein x eingeschoben: aus ba ll oo n wird ba lx lo on, steht am Ende nur noch ein Zeichen, so wird ebenfalls ein x angehängt: se ve n $\mapsto$ se ve nx

Sehr einfach durchzuführende Chiffrierung.

Dechiffrierung: Zeichen in gleicher Zeile/Spalte: der dechiffrierte Text steht jeweils links bzw. über den Geheimtextzeichen. Ansonsten muß der „Rechteckschritt“ ($\to$ 2.) umgekehrt werden. Außerdem müssen anschließend eingefügte 'x' entfernt werden.

**2-Tafel-Playfair:**
Zwei 5 x 5-Quadrate, z.B.

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

(zufällig gebildet)

Bigramm-Chiffrierschritte wie ah $\mapsto$ YA, wenn Klartextzeichen in selber Zeile, sonst Überkreuzschritt: xe $\mapsto$ WF, or $\mapsto$ NM, bx $\mapsto$ YR
Klartext wurde zusätzlich in Gruppen einer vorgegebenen Länge zerschnitten, die Nachricht

*„anxobergruppenfuehrerxvonxdemxbachxkiewxbittexdreitausendxschussxpatronenxschickenxstop“*

(x = Wortzwischenraum, hier nicht unterdrückt!) wurde z.B. in Gruppen von 17 Zeichen zerlegt und wie folgt chiffriert:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | n | x | o | b | e | r | g | r | u | p | p | e | n | f | u | e |
| h | r | e | r | x | v | o | n | x | d | e | m | x | b | a | c | h |

YA PK WF NM YR SZ WC EM YM VN ET GR AN PI AZ CO HW

Einfügen von x bei Doppelzeichen ist hier unnötig!

Dechiffrierung analog zum 1-Tafel-Playfair!

**Verfahren von Delastelle:**

Hin- und Rückübersetzen zum Chiffrieren:
Beispiel:

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | B | O | R | D | E |
| 2 | A | U | X | C | F |
| 3 | G | H | I | J | K |
| 4 | L | M | N | P | Q |
| 5 | S | T | V | Y | Z |

o 12 14 D

n 43 23 X

oder

| o | n | |
|---|---|---|
| 1 | 4 | D |
| 2 | 3 | X |

**Trigramm-Substitution:** $V^3 \to W^{(m)}$

Bereitet bereits technische Schwierigkeiten, da Papier nicht dreidimensional ist, so daß die Auflistung schwierig ist.

## 8. Der allgemeine Fall $V^{(n)} \to W^{(m)}$: Codes

Wörter werden in einem Codebuch (inkl. Zugeordnetem Geheimtext) gesammelt, je mehr desto besser, zusätzlich sollten alle Buchstaben aus V auch im Codebuch verzeichnet sein, damit selten vorkommende Wörter wenigstens buchstabenweise codiert werden können. Ein gutes Codebuch ordnet Zahlen/Buchstaben nicht in alphabetischer Reihenfolge den Klartextworten zu sondern möglichst wahllos und gibt außerdem mehrere Homophone für häufig verwendete Worte an.
Eigenarten von Personen gefährdeten solche Chiffrierungen, z.B. versah Leutnant Jäger im 1. Weltkrieg Nachrichten stets mit seinem (buchstabenweise) chiffriertem Namen. Für eine „schnelle“ Chiffrierung/Dechiffrierung müssen zwei verschiedene Codebücher existieren: eines, welches die Klartextworte alphabetisch auflistet (für den Chiffrierer), eines welches die Geheimtextworte alphabetisch/numerisch auflistet (für den Dechiffrierer).

KAPITEL 3

# Polyalphabetische Chiffrierung

**Özgür Göceri**

## 1. Einführung

Wir stellen einige klassische Verfahren zur polyalphabetischen Chiffrierung vor.
Diese Ausarbeitung stützt sich auf das Buch [**Bau97**]. Als eine weitere Quelle für den interessierten Leser sei das Werk [**Wob98**] genannt.

## 2. Kryptographie

Ursprünglich Geheimwissenschaft von den Geheimschriften und ihrem Gebrauch (Schwerpunkt: Vertraulichkeit von Nachrichten). Mittlerweile Wissenschaft von den algorithmischen, insbesondere mathematischen zur Gewährleistung von

- vertraulicher Kommunikation
- Authentizität und Integrität der Nachrichten:
  Der Empfänger einer Nachricht soll sicher sein, dass diese vom angegebenen Absender stammt und von Dritten nicht verändert worden ist.
- Authentizität der Teilnehmer bzw. der Einheiten:
  Die Kommunikationspartner sollen die Identität des jeweiligen Gegenüber überprüfen können, gleich ob es sich um menschliche Teilnehmer oder um Chipkarte und Chipkartenmaterial handelt.
- Anonymität bei der rechner- und netzgestützten Abwicklung von Alltagsgeschäften — von der elektronischen Geldbörse bis zu notariellen Kaufverträgen.

## 3. Polyalphabetische Chiffrierung: Begleitende Alphabete

Monoalphabetische Chiffrierung benutzt irgendeinen Chiffrierschritt immer wieder.
Eine echte polyalphabetische Chiffrierung erfordert, dass das Chiffrierschrittsystem $M$ mindestens die Kardinalität $k = 2$ hat, d. h. dass die Menge $M$ der verfügbaren Chiffrierschritte mindestens zweielementig ist. Die einzelnen Chiffrierschritte können verschiedenster Natur sein; so könnte das Chiffriersystem $M$ beispielsweise eine Anzahl einfacher Substitutionen und eine Anzahl Transpositionen der Breite 4 umfassen. In der Regel verlangt man aber, dass alle Chiffrierschritte ein und der selben Klasse angehören — beispielsweise der Klasse der Substitutionen, oder der linearen Substitutionen, oder der Transpositionen. Oft wird auch verlangt, dass alle Chiffrierschritte gleiche Chiffrierbreite besitzen, was zur Blockchiffrierung führt.
Das Hauptproblem ist, auf einfache Weise viele verschiedene Chiffrierschritte festzulegen oder, wie man sagt, viele verschiedene Alphabete zu erzeugen.

**3.1. Potenzierung.** Es liegt nahe, das Chiffrierschrittsystem dadurch aufzubauen, dass man aus einem Grundschritt andere Chiffrierschritte ableitet. Wir beschränken uns dabei auf den endomorphen Fall $V = W$, $m = n$.
Sei allgemein für $S : Q^n \to Q^n$

$$\{S^i : i \in \mathbb{N}\}$$

die Gruppe der potenzierten $S$-Alphabete. Mit festen Substitutionen

$$P_1 : V \longleftrightarrow Q^n \text{ und } P_2 : Q^n \longleftrightarrow W$$

bieten sich an die Mengen von Chiffrierschritten.

- $$\{P_1 S^i P_2 : i \in \mathbb{N}\}, \text{ wo } P_1 S^i P_2 : V \longleftrightarrow W$$
  Mit einer weiteren festen Substitution $R : Q^n \longleftrightarrow Q^n$
  $$\{S^{-i} R S^i : i \in \mathbb{N}\}$$
  gibt es die Gruppe der $S$-Ähnlichkeiten von $R$.
- $$\{S^{-i} R S^i P_2 : i \in \mathbb{N}\}, \text{ wo } P_1 S^{-i} R S^i P_2 : V \longleftrightarrow W$$
  Die Mengen sind jedenfalls endlich, denn $Q^n \longleftrightarrow Q^n$ mit endlichem $|Q| = N$ umfasst nur $(N^n)!$ verschiedene Permutationen.

### 3.2. Verschobene und rotierte Standardalphabete.

$$\{\rho^i : i \in \mathbb{N}\} = \{\rho^i \rho : i \in \mathbb{N}\}$$

Menge der verschobenen Standardalphabete

- Menge der horizontal verschobenen $P$-Alphabete
  $$\{\rho^i P : i \in \mathbb{N}\}$$
- Menge der vertikal verschobenen $P$-Alphabete
  $$\{P\rho^i : i \in \mathbb{N}\}$$
- Menge der $R$-rotierten Standardalphabete
  $$\{\rho^{-i} R \rho^i : i \in \mathbb{N}\}$$

Die Bezeichnungen werden klar, wenn man sich die Tafeln für die Familien von Substitutionen betrachtet. Mit $V = Q = W = Z_{26}$ und $N = 26$, $q = 1$, für das durch das Merkwort NEWYORK-CITY erzeugte Referenzalphabet $P$,

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

ergibt die Menge $\{\rho^i \rho : i \in \mathbb{N}\}$ die Tafel

| i | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | N | E | W | Y | O | R | K | C | I | T | A | B | D | F | G | H | J | L | M | P | Q | S | U | V | X | Z |
| 1 | E | W | Y | O | R | K | C | I | T | A | B | D | F | G | H | J | L | M | P | Q | S | U | V | X | Z | N |
| 2 | W | Y | O | R | K | C | I | T | A | B | D | F | G | H | J | L | M | P | Q | S | U | V | X | Z | N | E |
| 3 | Y | O | R | K | C | I | T | A | B | D | F | G | H | J | L | M | P | Q | S | U | V | X | Z | N | E | W |
| 4 | O | R | K | C | I | T | A | B | D | F | G | H | J | L | M | P | Q | S | U | V | X | Z | N | E | W | Y |
| 5 | R | K | C | I | T | A | B | D | F | G | H | J | L | M | P | Q | S | U | V | X | Z | N | E | W | Y | O |
| ⋮ | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 25 | Z | N | E | W | Y | O | R | K | C | I | T | A | B | D | F | G | H | J | L | M | P | Q | S | U | V | X |

während die Menge $\{P\rho^i : i \in \mathbb{N}\}$ die folgende Tafel hat

| i | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | N | E | W | Y | O | R | K | C | I | T | A | B | D | F | G | H | J | L | M | P | Q | S | U | V | X | Z |
| 1 | O | F | X | Z | P | S | L | D | J | U | B | C | E | G | H | I | K | M | N | Q | R | T | V | W | Y | A |
| 2 | P | G | Y | A | Q | T | M | E | K | V | C | D | F | H | I | J | L | N | O | R | S | U | W | X | Z | B |
| 3 | Q | H | Z | B | R | U | N | F | L | W | D | E | G | I | J | K | M | O | P | S | T | V | X | Y | A | C |
| 4 | R | I | A | C | S | V | O | G | M | X | E | F | H | J | K | L | N | P | Q | T | U | W | Y | Z | B | D |
| 5 | S | J | B | D | T | W | P | H | N | Y | F | G | I | K | L | M | O | Q | R | U | V | X | Z | A | C | E |
| ⋮ | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 24 | L | C | U | W | M | P | I | A | G | R | Y | Z | B | D | E | F | H | J | K | N | O | Q | S | T | V | X |
| 25 | M | D | V | X | N | Q | J | B | H | S | Z | A | C | E | F | G | I | K | L | O | P | R | T | U | W | Y |

Bei den *horizontal* verschobenen $P$-Alphabeten tritt das $P$-Alphabet in jeder Zeile in Erscheinung, von Zeile zu Zeile um eine Stelle nach links verschoben; bei den *vertikal* verschobenen $P$-Alphabeten ist das Referenzalphabet $P$ nur in der ersten Zeile erkennbar, dafür tritt das Standard-Alphabet vertikal auf.
$\{\rho^{-i} R \rho^i : i \in \mathbb{N}\}$ ist die Menge der $R$**−rotierten Standardalphabete**

Nimmt man für $R$ das zum selben Merkwort NEWYORKCITY gehörige Referenzalphabet wie oben, so ergibt die Menge $\{\rho^{-i}R\rho^i : i \in \mathbb{N}\}$ die Tafel

| i | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | N | E | W | Y | O | R | K | C | I | T | A | B | D | F | G | H | J | L | M | P | Q | S | U | V | X | Z |
| 1 | A | O | F | X | Z | P | S | I | D | J | U | B | C | E | G | H | I | K | M | N | Q | R | T | V | W | Y |
| 2 | Z | B | P | G | Y | A | Q | T | J | E | K | V | C | D | F | H | I | J | L | N | O | R | S | U | W | X |
| 3 | Y | A | C | Q | H | Z | B | R | U | K | F | L | W | D | E | G | I | J | K | M | O | P | S | T | V | X |
| 4 | Y | Z | B | D | R | I | A | C | S | V | L | G | M | X | E | F | H | J | K | L | N | P | Q | T | U | W |
| 5 | X | Z | A | C | E | S | J | B | D | T | W | M | H | N | Y | F | G | I | K | L | M | O | Q | R | U | V |
| ⋮ | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 21 | M | F | X | D | O | V | W | Y | A | B | C | E | G | H | K | L | N | P | Q | S | U | I | Z | R | T | J |
| 22 | K | N | G | Y | E | P | W | X | Z | B | C | D | F | H | I | L | M | O | Q | R | T | V | J | A | S | U |
| 23 | V | L | O | H | Z | F | Q | X | Y | A | C | D | E | G | I | J | M | N | P | R | S | U | W | K | B | T |
| 24 | U | W | M | P | I | A | G | R | Y | Z | B | D | E | F | H | J | K | N | O | Q | S | T | V | X | L | C |
| 25 | D | V | X | N | Q | J | B | H | S | Z | A | C | E | F | G | I | K | L | O | P | R | T | U | W | Y | M |

Bei den *R-rotierten* $P$-Alphabeten tritt das Referenzalphabet $R$ nur in der ersten Zeile auf, und wird *längs* der *Diagonalen* in der Reihenfolge von $P$ fortgesetzt.

**3.3. Zyklenzerlegung.** Die Zyklenzerlegung der begleitenden Alphabete ist interessant.
**Beispiel:**
Für

$$Q = \begin{pmatrix} a & b & c & d & e \\ B & A & D & E & C \end{pmatrix} = (ab)(cde) \text{ und } \rho = (abcde)$$

ergibt sich

$$\rho Q = \begin{pmatrix} a & b & c & d & e \\ b & c & d & e & a \end{pmatrix}\begin{pmatrix} b & c & d & e & a \\ A & D & E & C & B \end{pmatrix} = \begin{pmatrix} a & b & c & d & e \\ A & D & E & C & B \end{pmatrix}$$

$$Q\rho = \begin{pmatrix} a & b & c & d & e \\ B & A & D & E & C \end{pmatrix}\begin{pmatrix} B & A & D & E & C \\ C & B & E & A & D \end{pmatrix} = \begin{pmatrix} a & b & c & d & e \\ C & B & E & A & D \end{pmatrix}$$

$$\rho^{-1}Q\rho = \begin{pmatrix} a & b & c & d & e \\ e & a & b & c & d \end{pmatrix}\begin{pmatrix} e & a & b & c & d \\ D & C & B & E & A \end{pmatrix}$$

$$= \begin{pmatrix} a & b & c & d & e \\ D & C & B & E & A \end{pmatrix}$$

In der Substitutionsschreibweise lauten die Alphabete in diesem Beispiel:

| i | a | b | c | d | e |
|---|---|---|---|---|---|
| 0 | B | A | D | E | C |
| 1 | A | D | E | C | B |
| 2 | D | E | C | B | A |
| 3 | E | C | B | A | D |
| 4 | C | B | A | D | E |

| i | a | b | c | d | e |
|---|---|---|---|---|---|
| 0 | B | A | D | E | C |
| 1 | C | B | E | A | D |
| 2 | D | C | A | B | E |
| 3 | E | D | B | C | A |
| 4 | A | E | C | D | B |

| i | a | b | c | d | e |
|---|---|---|---|---|---|
| 0 | B | A | D | E | C |
| 1 | D | C | B | E | A |
| 2 | B | E | D | C | A |
| 3 | B | C | A | E | D |
| 4 | E | C | D | B | A |

In der Zyklenschreibweise erhält man

- als Menge der horizontal verschobenen $P$-Alphabete {(ab)(cde), (a)(bdce), (adbe)(c), (aed)(bc), (ac)(b)(d)(e)}
- als Menge der vertikal verschobenen $P$-Alphabete {(ab)(cde), (aced)(b), (adbc)(e), (ae)(bdc), (a)(be)(c)(d)}
- als Menge der $P$-rotierten Alphabete {(ab)(cde), (bc)(dea), (cd)(eab), (de)(abc), (ea)(bcd)}

Aus der Gruppentheorie weiß man, dass eine Ähnlichkeitstransformation $\rho^{-i}P\rho^i$ die Länge der Einzelzyklen einer Permutation invariant lässt. Alle Substitutionen aus der Familie der begleitenden rotierten $P$-Alphabete besitzen also ein und die selbe Zyklenzerlegung. Man hat das den *Hauptsatz der Rotor-Chiffrierung* genannt. Für die verschobenen $P$-Alphabete trifft solches nicht zu.

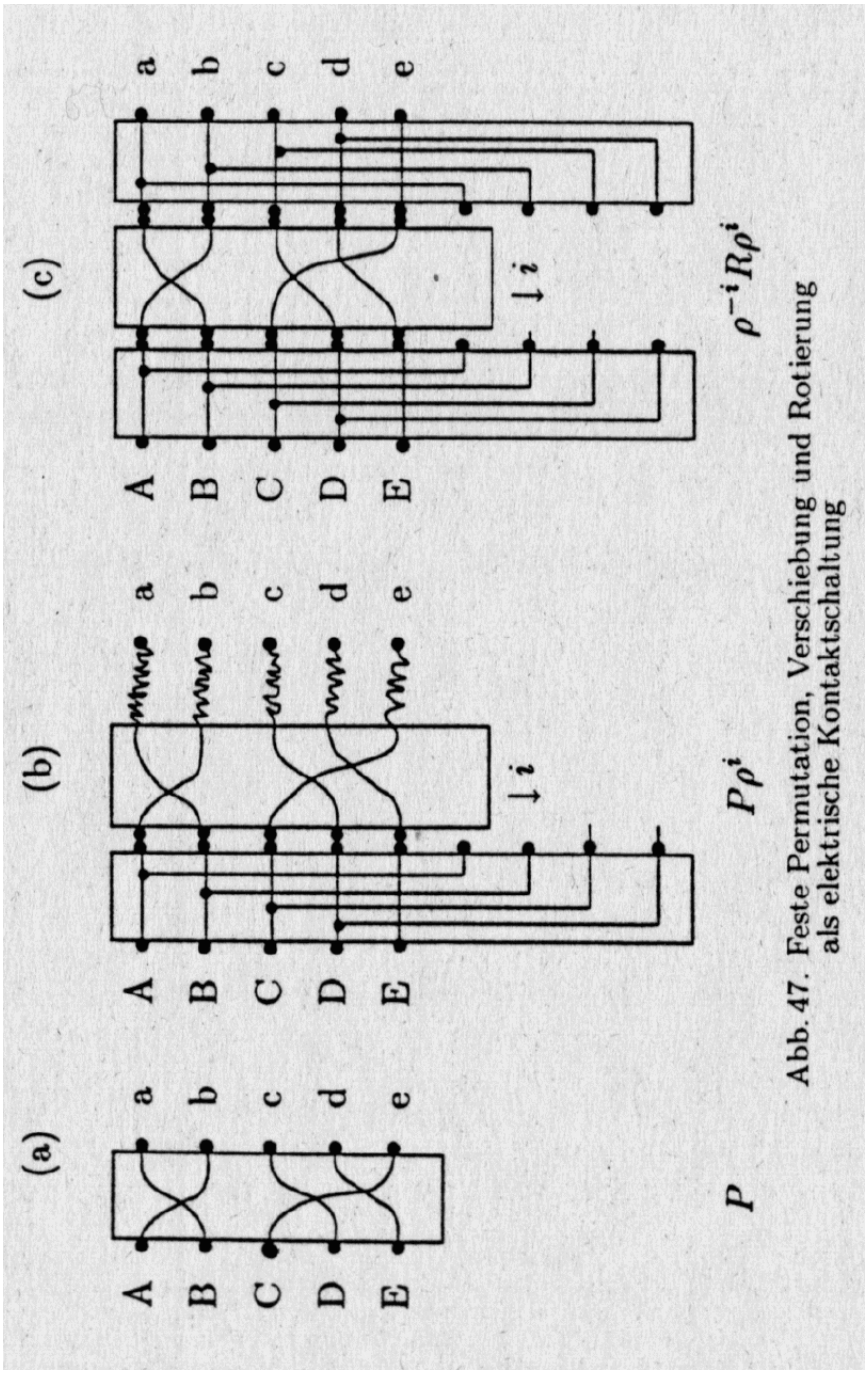



Abb. 47. Feste Permutation, Verschiebung und Rotierung als elektrische Kontaktschaltung

**3.4. Rotor-Maschinen.** Zur elektrischen Kontaktrealisierung einer festen Substitution $P$ dient ein Schaltkasten $P$ mit $N$ Eingangsbuchsen für die Klartextzeichen und $N$ Ausgangsbuchsen für die Geheimtextzeichen, intern verbunden durch $N$ Leitungsdrähte, Abb. 47(a).
Zum Zwecke einer elektrischen Kontaktrealisierung von begleitenden verschobenen $P$-Alphabeten $P\rho^i$ kann man hinter die ausgangsseitigen Buchsen des Schaltkastens $P$ eine verschiebbare Kontaktleiste, oder den Schaltkasten $P$ verschiebbar machen, Abb. 47(b). In jedem Fall muss man flexible Leitungsverbindungen vorsehen, was zu einem Abnutzungsproblem führt.
Dies vermeidet man, wenn man zunächst je eine verschiebbare Kontaktleiste eingangsseitig und ausgangsseitig anbringt und beide starr koppelt. Es ist stattdessen auch möglich, auf flexible

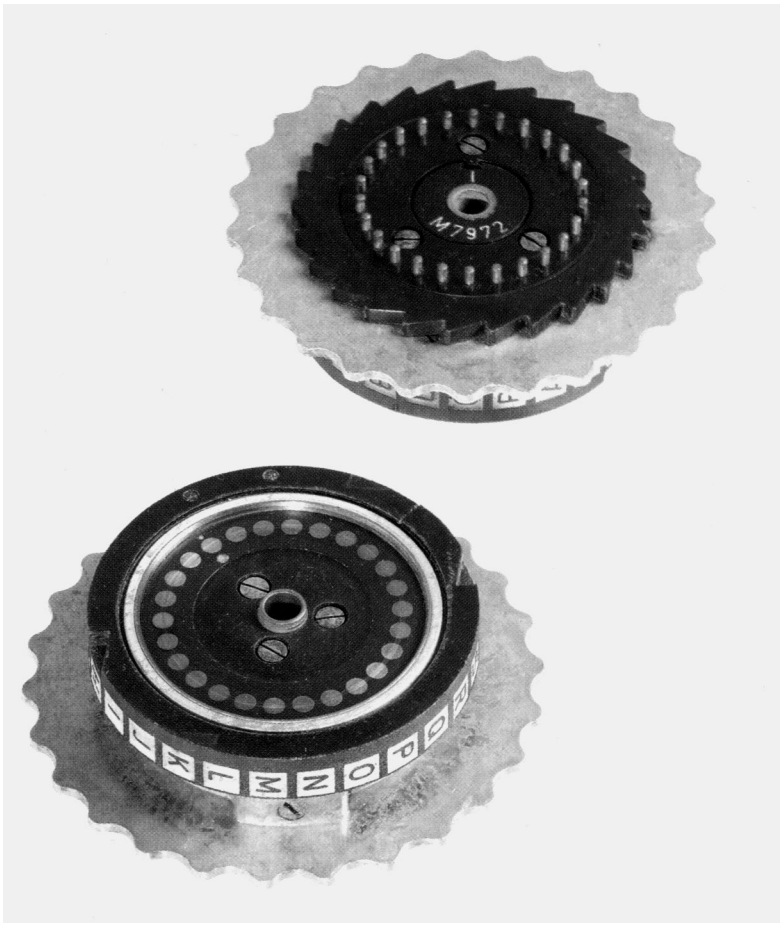



ABBILDUNG 1. Halbrotor

Leitungsverbindungen zu verzichten, indem man $P$ verschiebt, Abb. 47(c). Ohne Duplikation der ein- und ausgangsseitigen Kontakte kommt man aus, wenn man eine drehbare Kontaktscheibe, einen **Rotor**, benutzt. Man erhält dadurch eine elektrische Kontaktrealisierung von begleitenden, rotierten P - Alphabeten $\rho^{-i}P\rho^{i}$. Der Rotor (s. Abbildung 1) ist dabei eine Scheibe. Auf ihren gegenüberliegenden Kreisflächen sind ringförmig je 26 Kontakte angebracht. Jeder Kontakt auf der linken Seite ist mit genau einem Kontakt auf der rechten Seite verbunden. Dies entspricht einer Substitution.
Die linken Kontaktflächen werden nun von 26 Schleifkontakten abgegriffen ebenso die rechten. Die Schleifkontakte entsprechen den Buchstaben des Alphabets. Legt man an einen der linken Schleifkontakte per Tastendruck eine Spannung an, kommt diese bei einem anderen rechten Schleifkontakt an. Das Verfahren ist nichts anderes wie eine einfache Substitution. Es lag nahe, mehrere solcher Scheiben nebeneinander zu verwenden, so war auch die **Enigma** (s. Abbildungen 2 und 3), eine Chiffriermaschine, die im zweiten Weltkrieg benutzt wurde, aufgebaut. Zwischen je zwei Scheiben werden Schleifkontakte angebracht, die immer eine rechte Kontaktfläche der linken Scheibe mit der gegenüberliegenden linken Kontaktfläche der rechten Scheibe verbindet. Neben den Rotoren gibt es auch noch die Umkehrwalze. Die Umkehrwalze besitzt nur auf einer Seite eine Kontaktfläche. Die Kontaktfläche sind auch untereinander verbunden, und zwar so, dass jede Fläche genau mit einer anderen verbunden ist.
Die Spannung, die nun an einen Buchstaben ganz rechts angelegt wird, pflanzt sich durch die Rotoren nach links fort, wird in der Umkehrwalze umgelenkt und durchwandert die Rotoren wieder von links nach rechts auf einem ganz anderen Pfad (s. Abbildung 49).

ABBILDUNG 2. Enigma von außen

Stromlauf der elektrischen Realisierung für das Klartextzeichen e und das zugehörige Geheimtextzeichen m. Dabei entsteht die Familie $\{P_{(i_1,i_2,i_3,i_4)}\}$ von echten Involutionen, wo

$$P_{(i_1,i_2,i_3,i_4)} = S_{(i_1,i_2,i_3,i_4)} U S^{-1}_{(i_1,i_2,i_3,i_4)} \text{ und } P_{(i_1,i_2,i_3,i_4)} = \rho^{-i_1} R_N \rho^{i_1-i_2} R_M \rho^{i_2-i_3} R_L \rho^{i_3-i_4}$$

Die Rotoren bewegen sich dabei wie in einem Zählwerk fort. Hat ein Rotor eine bestimmte Stellung erreicht, so wird der nächste Rotor durch eine Nase um einen Schritt weiter gedreht.

**3.5. Verschobene Standardalphabete: Vigenère und Beaufort.** Ein Vigenère-Chiffrierschritt bewirkt eine einfache lineare Substitution: Der Zyklus $\rho$ legt als zyklische lineare Ordnung von $V^n$ die Addition modulo $N^n fest$; einem Chiffrierschritt $\rho^i : i \in \mathbb{N}$ entspricht die Addition $A_i : i \in \mathbb{Z}_{\mathbb{N}^n}$ einer Verschiebungzahl $i$ :

$$A_i : A_i(x) \overset{N^n}{\simeq} x + i$$

Der Dechiffrierschritt $A_i^{-1}$ bedeutet eine Subtraktion der Verschiebungszahl:

$$A_i^{-1} : A_i^{-1}(y) \overset{N^n}{\simeq} y - i$$

**Beispiel:**

$$A_5 : A_5(b) \overset{6}{\simeq} b + 5 \to 1 + 5 \equiv 0 \pmod 6$$

$$A_5^{-1} : A_5^{-1}(a) \overset{6}{\simeq} a - 5 \to 0 - 5 \equiv 1 \pmod 6$$

Multiplikation mit einem regulären Faktor $q$ ergibt auch eine Familie

$$\{C_q : q \in \mathbb{Z}_{\mathbb{N}^n} \text{ und } ggt(q, N^n) = 1\}$$

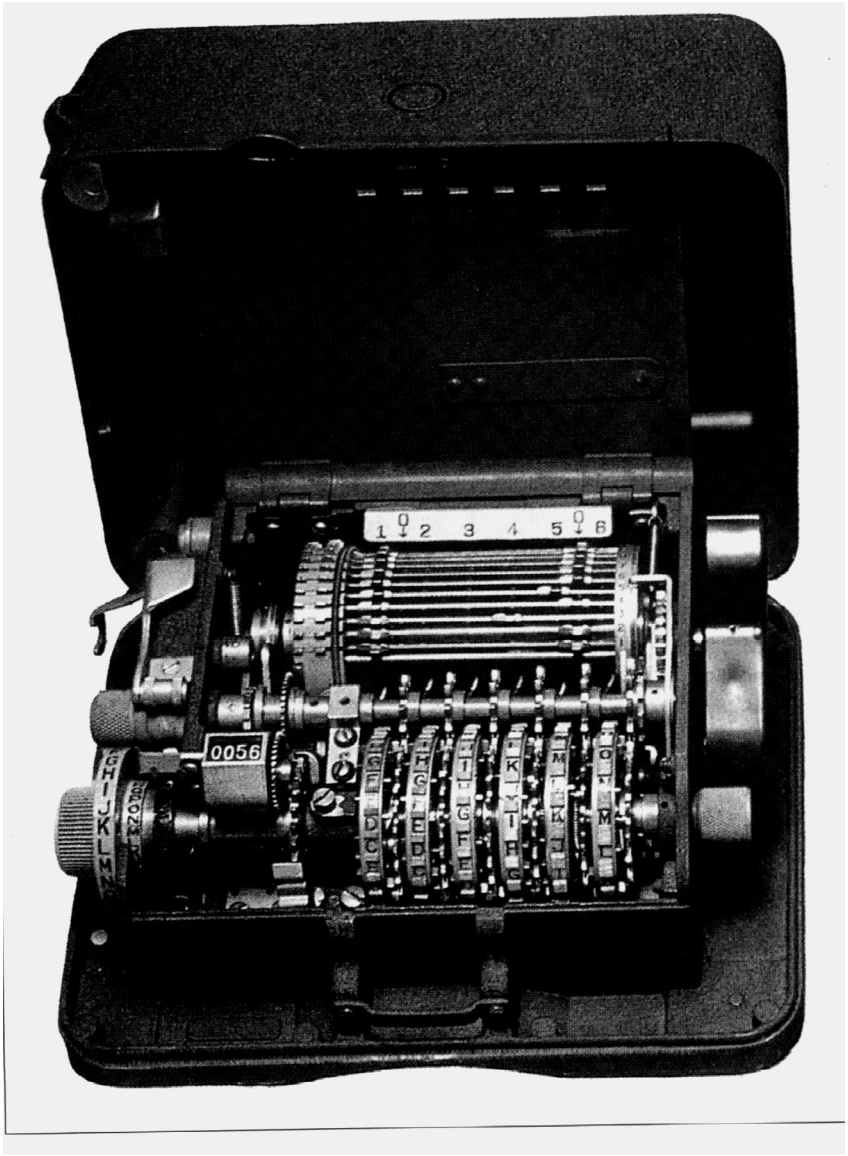



ABBILDUNG 3. Das Innenleben der Enigma

begleitender Chiffrierschritte:

$$C_q : C_q(x) \overset{N^n}{\simeq} q * x$$

Der Dechiffrierschritt ist

$$C_q^{-1} : C_q^{-1}(x) \overset{N^n}{\simeq} q' * x, \text{ wo } q * q' \overset{N^n}{\simeq} 1$$

**Beispiel :**

$$C_5 : C_5(c) \overset{6}{\simeq} 5 * c \to 5 * 2 \equiv 4 \pmod 6$$

$$C_5^{-1} : C_5^{-1}(c) \overset{6}{\simeq} q' * c \to 5 * 2 \equiv 4 \pmod 6$$

Der Fall, dass $q$ fest und zwar gleich $N^n - 1$ gewählt wird, ergibt die Familie $\{B_i : i \in \mathbb{Z}_{N^n}\}$ mit

$$B_i : B_i(x) \overset{N^n}{\simeq} i - x \quad (\textbf{Beaufort-}\text{Chiffrierschritt})$$

Der Dechiffrierschritt ist

$$B_i^{-1} : B_i^{-1}(y) \overset{N^n}{\simeq} i - y$$

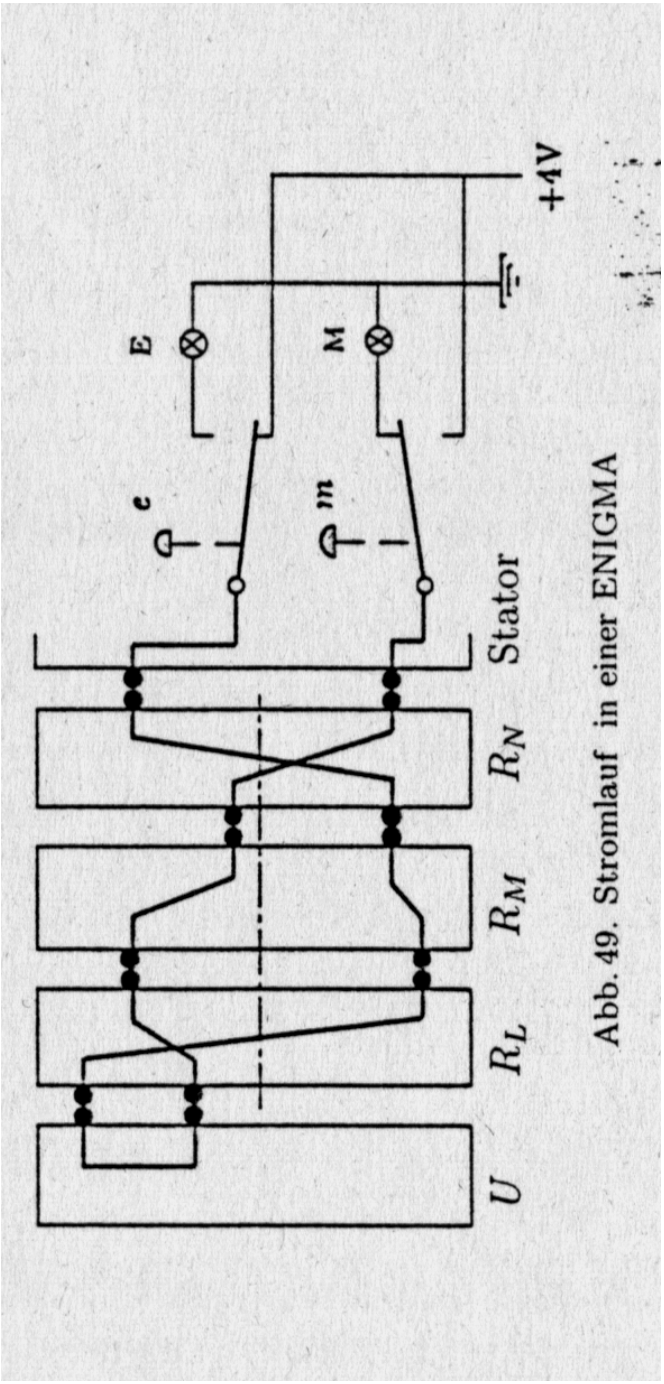



Abb. 49. Stromlauf in einer ENIGMA

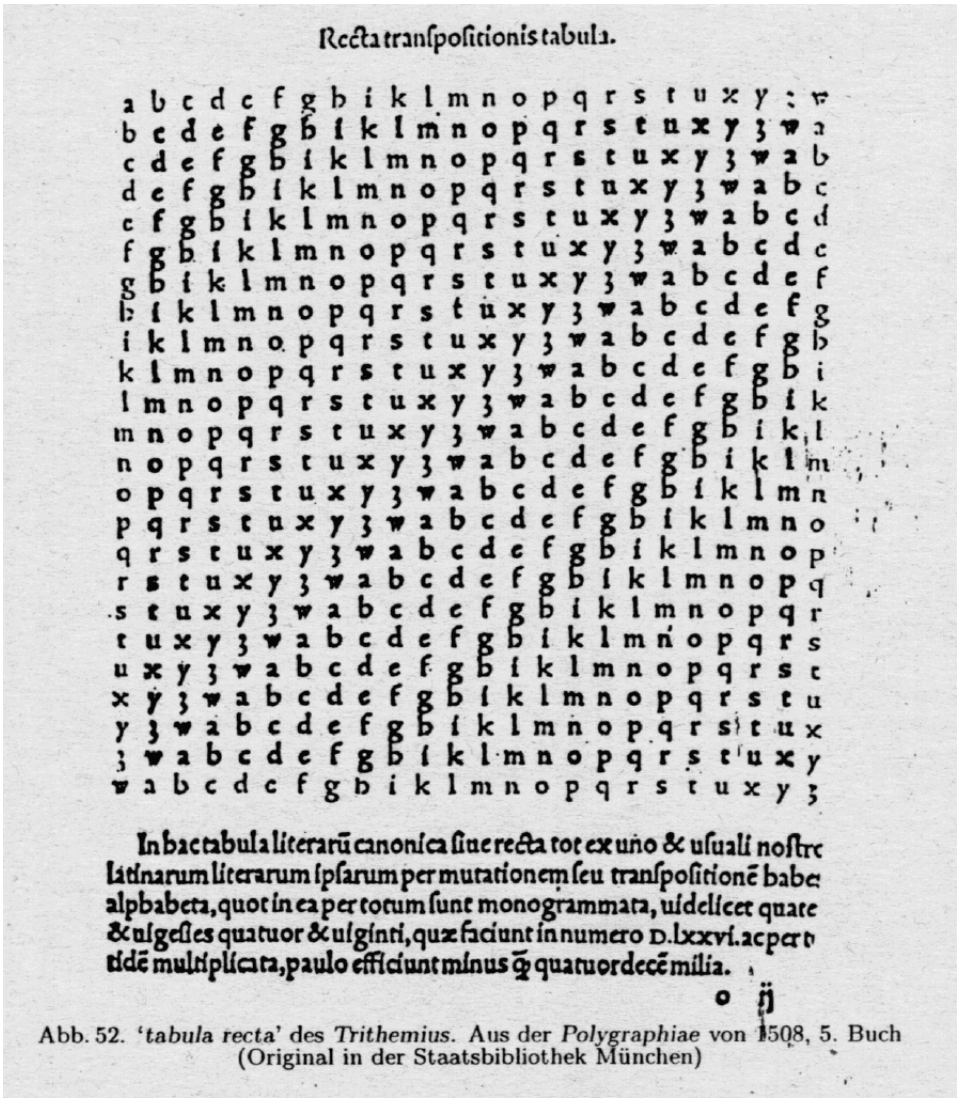
Eine triviale Variante der Vigenère-Chiffrierung

$$E_i : E_i(x) \overset{N^n}{\simeq} -i + x$$

wird als „Rückwärts"-Vigenère bezeichnet.

## 4. Polyalphabetische Chiffrierung: Schlüssel

**4.1. Frühe Verfahren mit periodischen Schlüsseln.** Leon Battista Alberti (1404-1472). Aufbauend auf seinen Überlegungen zur Kryptanalyse erkannte Alberti **1466**, dass es beim Gebrauch einer einfachen Substitution nicht ausreichte, sie von Zeit zu Zeit zu wechseln. Er schlug



Recta tranſpoſitionis tabula.

a b c d e f g h i k l m n o p q r s t u x y z w
b c d e f g h i k l m n o p q r s t u x y z w a
c d e f g h i k l m n o p q r s t u x y z w a b
d e f g h i k l m n o p q r s t u x y z w a b c
e f g h i k l m n o p q r s t u x y z w a b c d
f g h i k l m n o p q r s t u x y z w a b c d e
g h i k l m n o p q r s t u x y z w a b c d e f
h i k l m n o p q r s t u x y z w a b c d e f g
i k l m n o p q r s t u x y z w a b c d e f g h
k l m n o p q r s t u x y z w a b c d e f g h i
l m n o p q r s t u x y z w a b c d e f g h i k
m n o p q r s t u x y z w a b c d e f g h i k l
n o p q r s t u x y z w a b c d e f g h i k l m
o p q r s t u x y z w a b c d e f g h i k l m n
p q r s t u x y z w a b c d e f g h i k l m n o
q r s t u x y z w a b c d e f g h i k l m n o p
r s t u x y z w a b c d e f g h i k l m n o p q
s t u x y z w a b c d e f g h i k l m n o p q r
t u x y z w a b c d e f g h i k l m n o p q r s
u x y z w a b c d e f g h i k l m n o p q r s t
x y z w a b c d e f g h i k l m n o p q r s t u
y z w a b c d e f g h i k l m n o p q r s t u x
z w a b c d e f g h i k l m n o p q r s t u x y
w a b c d e f g h i k l m n o p q r s t u x y z

In hac tabula literarū canonica ſiue recta tot ex uno & uſuali noſtre latinarum literarum ipſarum permutationem ſeu tranſpoſitionē habe alphabeta, quot in ea per totum ſunt monogrammata, uidelicet quate & uigeſies quatuor & uiginti, quæ faciunt in numero D.lxxvi. ac per tidē multiplicata, paulo efficiunt minus q quatuordecē milia.

o ij

Abb. 52. '*tabula recta*' des *Trithemius*. Aus der *Polygraphiae* von 1508, 5. Buch (Original in der Staatsbibliothek München)

vor, nach jeweils drei oder vier Worten zu einem anderen Alphabet überzugehen (Alberti Scheibe). **1518** schlug Trithemius vor, nach jedem Buchstaben zum nächsten Alphabet überzugehen, und so nach einer festen Progressionsvorschrift alle verfügbaren Alphabete zu benutzen, bevor ein Alphabet zum zweiten mal benutzt wird. Trithemius gab auch die erste Chiffriertafel an („tabula recta").

**1553** erfand Giovanni Battista Belaso die Chiffrierung durch Angabe eines **Schlüsselwortes** zur sukzessiven Verdrehung der Scheibe oder der Auswahl der Zeile. Es handelt sich dabei um einen periodischen, d.h. endlichen wiederholten Schlüssel, der eine polyalphabetische Chiffrierung ergibt. Die kombinatorische Komplexität $Z$ des Verfahrens ist $N^{n*d}$. **1563** verband Porta (1535-1615) Albertis Gebrauch eines permutierten Alphabets $P$ mit Belasos Gebrauch eines Schlüssels zur Bestimmung der Verdrehung der Scheibe. Da auch ein Kennwort zur Festlegung des permutierten Alphabets als Schlüssel dienen kann, sprach man von „doppelter Chiffrierung". Die Komplexität des Verfahrens $Z$ ist $(N^n - 1)! * N^{n*d}$.

**4.2. Vernam-Chiffrierung.** Ein besonders einfacher Fall polyalphabetischer Chiffrierung ist das bitweise Vigenère-Verfahren, dass die rechnerfreundliche Umsetzung des zeichenweise Verfahrens darstellt. Bisher haben wir nur 26 Buchstaben betrachtet und modulo 26 addiert. Heute

arbeiten wir also mit Bits und Bytes. Ein Bit ist nichts weiter als ein Buchstabe in einem zweielementigem Alphabet. Die Addition modulo 2 in diesem Alphabet entspricht gerade dem bitweisen *xor*. Ist eine Chiffrierung $\mathbb{Z}_2 \to \mathbb{Z}_2$ definal, so ist sie eine Permutation der beiden Elemente **O** und **L**, es gibt nur die **Identität**

$$O : \mathbf{O} \mapsto \mathbf{O} \ , \ \mathbf{L} \mapsto \mathbf{L}$$

und die **Spiegelung**

$$L : \mathbf{O} \mapsto \mathbf{L} \ , \ \mathbf{L} \mapsto \mathbf{O}$$

als Chiffrierschritte (Vernam-Chiffrierschritte). Die Chiffrierung ist notwendigerweise involutorisch, aber nicht echt involutorisch. Es ist $|M| = 2$ der kleinste Wert, der eine polyalphabetische Chiffrierung erlaubt. Als Schlüssel dient eine endliche (**O,L**)-Folge, die periodisch wiederholt wird, oder eine unendliche (**O,L**)-Folge.

**4.3. Quasi-nichtperiodische Schlüssel.** Bis ins 19. Jahrhundert galt die Polyalphabetische Chiffrierung als unbrechbar. Ein echter Einbruch gelang nur, wenn Standardalphabete mit Verschiebung verwendet wurden, dann konnten Worte aus dem Klartext erraten und der Schlüssel rekonstruiert werden, oder es war möglich das Schlüsselwort zu erraten. Mit der im 19. Jahrhundert aufkommenden systematischen Lösung der periodischen polyalphabetischen Chiffrierung änderte sich das. Wollte man Angriffsmöglichkeiten ausschließen, so musste man zu einer Periodenlänge greifen, die deutlich größer war als die gesamte Nachricht (quasi-nichtperiodischer Schlüssel) oder man geht zu fortlaufenden Schlüsseln über. Wenn man aber doch periodisch chiffrieren will, dann sollte aufgrund der Sicherheit mehr Alphabete verwendet werden als man Schlüsselbuchstaben hat. Die Alphabete sollten dabei auch unregelmäßig verwendet werden. Die progressive Chiffrierung ist eine polyalphabetische Chiffrierung, bei der kein Alphabet benutzt wird, bevor jedes andere Alphabet benutzt wurde. Eine progressive Chiffrierung ist also periodisch mit einer Periode gleich der Kardinalität $k$ der Menge der Chiffrierschritte. Eine quasi-nichtperiodische Chiffrierung entsteht, wenn die Nachricht kürzer ist als $k$.

**4.4. Nichtperiodische Schlüssel.** Eine nichtperiodische Chiffrierung erfordert $k \geq 2$ und eine nichtperiodische Folge $(X_{i_1}, X_{i_2}, X_{i_3}...)$ von Chiffrierschritten. Sie wird charakterisiert durch die Indexfolge $(i_1, i_2, i_3...)$ mit $0 \leq i_\mu < k$.
Eine nichtperiodische Chiffrierung wie etwa (für $k = 2$) eine mit der unendlichen Indexfolge

$$(1101000100000001...)$$

d.h.

$$i_\mu = \begin{cases} 1 & \text{falls } \mu = 2^k \\ 0 & \text{sonst} \end{cases}$$

bringt gegenüber einer periodischen Chiffrierung keinen Vorteil.
Aber auch eine fortlaufende Chiffrierung (10010110011010010110...) hat ein einfach zu durchschauendes Bildungsgesetz des Schlüssels, das eine rekursive Berechnung erlaubt. Und die fraktale Wortfolge

$a_0 \widehat{=} (0)$
$a_1 \widehat{=} (1)$
$a_2 \widehat{=} (01)$
$a_3 \widehat{=} (101)$
$a_4 \widehat{=} (01101)$
$a_5 \widehat{=} (10101101)$
$a_6 \widehat{=} (0110110101101)$
$a_7 \widehat{=} (101011010110110101101)$
$\vdots$

die durch das Lindenmayer-Ersetzungssystem

$$\begin{cases} 0 \to 1 \\ 1 \to 01 \end{cases}$$

erzeugt wird, hat ebenfalls ein durchschaubares Bildungsgesetz: es ist für

$$i \geq 2 \qquad a_i = a_{i-2} \circ a_{i-1}.$$

**4.5. Wie erhält man eine „unregelmäßigen" nichtperiodischen Indexfolge?** Der entscheidende Schritt kommt von Geronimo Cardano (1501-1576). Nachdem Belaso polyalphabetische Substitution eingeführt hat, leitet Cardano aus dem Klartext den Schlüssel ab. Alphabet $Z_{20} \cup \{x, y\}$

| a | b | c | d | e | f | g | h | i | l | m | n | o | p | q | r | s | t | v | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| s | i | c | | e | r | g | o | | e | l | e | m | e | n | t | i | s | | | | |
| S | I | C | | S | I | C | E | | S | I | C | E | R | G | O | E | L | | | | |
| N | T | F | | Z | C | L | T | | Z | V | H | R | Y | V | I | P | E | | | | |

Die Chiffrierung des Startwortes mit sich selbst jedoch als Schlüssel ist nicht eindeutig umkehrbar. s und S wie auch f und F ergeben N. Vigenère hatte die Idee einen frei wählbaren Schlüssel ins Spiel zu bringen. Er wählte den ersten Buchstaben des Schlüssels frei und nahm als weitere Schlüsselbuchstaben entweder die des Klartextes oder die des Geheimtextes.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | u | n | o | m | d | e | l | e | t | e | r | n | e | l |
| D | A | U | N | O | M | D | E | L | E | T | E | R | N | E |
| X | I | A | H | G | U | P | T | M | L | S | H | I | X | T |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | u | n | o | m | d | e | l | e | t | e | r | n | e | l |
| D | X | H | E | E | C | O | U | M | X | G | N | A | B | Q |
| X | H | E | E | C | O | U | M | X | G | N | A | B | Q | O |

Die zweite Art ist allerdings unbrauchbar, da der Schlüssel vollständig exponiert ist, kann die ganze Nachricht bis auf das erste Zeichen sofort entschlüsselt werden. Die erste Art des rekurrenten Verfahrens ist kaum besser. Man muss den Anfangsbuchstaben des Schlüssels kennen, um weiterzukommen, jedoch sind die zwölf Möglichkeiten schnell durchprobiert.

**4.6. Gibt es überhaupt eine sichere Chiffriermethode?** Nachdem die einfachen rekurrenten Chiffrierverfahren nichts einbringen, gibt es noch die Möglichkeit so genannte **One Time Pads**, auch individueller Schlüssel genannt, zu benutzen. Dabei handelt es sich um eine polyalphabetische Chiffrierung mit unendlicher Länge. Der entscheidende Punkt bei dieser Methode ist, dass der betroffene Abschnitt des Schlüssels nur ein einziges mal verwendet werden darf. Problem hierbei jedoch ist die Erzeugung eines „echt zufälligen" Schlüssels. Man könnte meinen,dass ist mittels eines Rechners erzeugbar wäre, dem ist es aber nicht so. Denn die Rechnerausgabe genügt Gesetzmäßigkeiten und kann höchstens deshalb nur „pseudo-zufällig" sein. Als Fazit kann man aus der ganzen Sache herausziehen, dass eine wesentliche Schwäche aller Algorithmen ist, dass immer ganze Zeichen verschlüsselt werden. Auch wenn der Algorithmus noch so gut ist, bleiben meist statistische Abhängigkeiten erhalten.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *A* | *B* | ↕ | a | b | c | d | e | f | g | h | i | l |
| | | | m | n | o | p | q | r | s | t | u | x |
| *C* | *D* | ↕ | a | b | c | d | e | f | g | h | i | l |
| | | | x | m | n | o | p | q | r | s | t | u |
| *E* | *F* | ↕ | a | b | c | d | e | f | g | h | i | l |
| | | | u | x | m | n | o | p | q | r | s | t |
| *G* | *H* | ↕ | a | b | c | d | e | f | g | h | i | l |
| | | | t | u | x | m | n | o | p | q | r | s |
| *I* | *L* | ↕ | a | b | c | d | e | f | g | h | i | l |
| | | | s | t | u | x | m | n | o | p | q | r |
| *M* | *N* | ↕ | a | b | c | d | e | f | g | h | i | l |
| | | | r | s | t | u | x | m | n | o | p | q |
| *O* | *P* | ↕ | a | b | c | d | e | f | g | h | i | l |
| | | | q | r | s | t | u | x | m | n | o | p |
| *Q* | *R* | ↕ | a | b | c | d | e | f | g | h | i | l |
| | | | p | q | r | s | t | u | x | m | n | o |
| *S* | *T* | ↕ | a | b | c | d | e | f | g | h | i | l |
| | | | o | p | q | r | s | t | u | x | m | n |
| *U* | *X* | ↕ | a | b | c | d | e | f | g | h | i | l |
| | | | n | o | p | q | r | s | t | u | x | m |

Abb. 69. Involutorische polyalphabetische Chiffrierung von *Vigenère*

KAPITEL 4

# Kryptanalyse mono- und polyalphabetischer Verfahren

**Frank Michel**

## 1. Einleitung

Diese Dokument wurde im Rahmen des Proseminars Kryptographie im SS2000 an der Universität Kaiserslautern erstellt. Es behandelt die Kryptanalyse mono- und polyalphabetischer Chiffrierverfahren.
Es werden zuerst einige kurze aber wichtige Begriffe eingeführt. Unter anderem wird eine kurze Einführung in das Prinzip der Exhaustionsmethode geliefert, die in den meisten kryptanalytischen Verfahren — wie wir später sehen werden — Verwendung findet, wenn man den Suchraum nur genügend einschränkt.
Im nächsten Abschnitt befassen wir uns mit Dechiffrierverfahren zur Brechung monoalphabetischer Chiffrierungen. Hier betrachten wir zuerst die Anatomie der Sprache, nämlich Muster.Desweiteren betrachten wir die Möglichkeit mit Hilfe wahrscheinlicher Wörter einen Einstieg in die Chiffrierung zu finden.
Im letzten Abschnitt kommen wir zu den polyalphabetischen Chiffrierungen und Verfahren sie zu Brechen. Auch dort werden wir auf Methoden eingehen, die mittels wahrscheinlicher Wörter und Mustersuche arbeiten.
Abschließend sehen wir eine Zusammenfassung und einen kurzen Ausblick.

## 2. Einführung

In diesem Kapitel werden der Begriff der kombinatorischen Komplexität und der Unizitätslänge eingeführt. Das grundlegende Verfahren der Exhaustion wird vorgestellt und auf deren praktische Durchführung eingegangen.

**2.1. Die kombinatorische Komplexität.** Die Mächtigkeit eines Verfahrens bzw. einer ganzen Klasse von Chiffrierungen ist ein Maß für die kombinatorische Komplexität der Chiffrierung. Dieser Mächtigkeit entspricht genau die Anzahl der zu einem Verfahren gehörigen Schlüssel. Sie gibt die obere Schranke für die Sicherheit gegen unbefugtes Entschlüsseln an, d.h. den maximalen Arbeitsaufwand der *Exhaustion* bei bekanntem Verfahren. In Tabelle 1 sehen wir eine Zusammenstellung einiger Werte für verschiedene Chiffrierverfahren. Hierbei ist $\boldsymbol{N}$ die Mächtigkeit des Alphabetes, also die Anzahl der verschiedenen Zeichen, $\boldsymbol{Z}$ ist die kombinatorische Komplexität des Verfahrens, $\boldsymbol{n}$ gibt die Chiffrierbreite an und $\boldsymbol{d}$ ist die Anzahl der benutzten Alphabete bei polyalphabetischen Chiffrierungen.

**2.2. Die Exhaustionsmethode.** Die eben eingeführte kombinatorische Komplexität ist ein Maß für eine ganz bestimmte, aber sehr allgemeine Entzifferungsmethode, die Exhaustion.
Das Prinzip der Exhaustionsmethode ist, alle möglichen Klartexte, die nach dem vermuteten Verfahren zum empfangenen Geheimtext passen, zu konstruieren, und dann den „richtigen“ Klartext herauszulesen. Man sieht sofort, dass dies nur für Verfahren praktisch durchgeführt werden kann, deren Anzahl an Varianten nicht riesengroß ist. Um nicht unnötige Varianten aufschreiben zu müssen, schreibt man für längere Nachrichten nicht jeden möglichen Klartext auf, sondern jeweils ein Stück (im Extremfall jeweils ein Zeichen), um dann unmögliche Buchstabenkombinationen auszuschließen.

[1] $(N-1)!! := (N-1)*(N-3)*...*5*3*1$

| Verfahren | Kombin. Komplexität $Z$ | Art |
|---|---|---|
| Einfache Substitution | $N!$ | 1 |
| Vollzyklische einfache Substitution | $(N-1)!$ | 1 |
| Echt involutorische einfache Substitution | $(N-1)!!$ [1] | 1 |
| CAESAR-Addition | N | 1 |
| Polygraphische Substitution | $(N^n)!$ | 2 |
| Polygraphische Addition | $N^n$ | 2 |
| Transposition | $n!$ | 3 |
| VIGENÈRE/BEAUFORT | $N^d$ | 3 |
| ALBERTI | $(N-1)! * N^d$ | 3 |

TABELLE 1. Kombinatorische Komplexitäten im Überblick

Falls keine „richtige“ Nachricht herausgelesen werden kann liegt entweder ein Chiffrierfehler vor, oder es wurde nicht das vermutete Verfahren benutzt.
Auch kann es vorkommen, dass mehr als eine „richtige“ Nachricht herausgelesen werden kann, d.h. die Entzifferung ist nicht eindeutig. Dies hängt mit der *Unizitätslänge* des benutzten Chiffrierverfahrens zusammen.

**2.3. Unizitätslänge.** Als Unizitätslänge $\boldsymbol{U}$ bezeichnet man die Anzahl der Zeichen bis zu der Stelle, an der man die Entscheidung für genau einen Klartext deutlich fällen kann. Falls die Anzahl der Zeichen im empfangene Text die Unizitätslänge des vermuteten Verfahrens unterschreitet, ist keine eindeutige Entschlüsselung möglich.
Empirische Versuche ergaben, dass die Unizitätslänge lediglich von der kombinatorischen Komplexität $Z$ des Verfahrens abhängt, und für nicht zu kleine $Z$ Proportional $ldZ$ ist.
Falls eine Unizitätslänge existiert, so ist zu erwarten, dass auch durch andere geeignete Methoden als durch Exhaustion das Brechen einer Chiffrierung mit wachsender Länge des Chiffrats weniger unsicher und ab einer gewissen, in der Nähe der Unizitätslänge liegenden Länge des Chiffrats unproblematisch wird, sobald ein hinreichender Aufwand getrieben wird [**Bau97**].

**2.4. Praktische Durchführung der Exhaustion.** Bei der praktischen Durchführung der Exhaustion vergrößert man schrittweise die Länge der zu betrachtenden Stücke und entfernt „unmögliche“ Varianten. Von den 676 verfügbaren Bigrammen sind zum Beispiel nur 50 % möglich, von den 17 576 Trigrammen nur ca. 500 Stück. Während der Durchführung vermindert sich die Anzahl der zu betrachtenden Varianten ganz drastisch, wie auch in Tabelle 2 leicht ersichtlich ist. Dieses Verfahren ist leicht mechanisierbar und mit Rechenanlagen interaktiv durchführbar, falls die Anzahl der möglichen Varianten im Bereich von mehreren zehntausenden liegt. Dies zeigt auch welche (beschränkte) Tragweite die Exhaustion eigentlich hat, denn für allgemeine monoalphabetische Chiffrierungen mit $Z \approx 10^{25}$, ist sie praktisch nicht mehr durchführbar. Wenn man allerdings die noch so hoch erscheinende Komplexität durch geeignete Verfahren genügend einschränken kann, kommt die Exhaustion wieder zum Einsatz.
Mit anderen Worten:

> *Die Exhaustionsmethode, obschon für sich allein ziemlich bedeutungslos, ist in Kombination mit anderen Verfahren die Grundmethode der Kryptanalyse.*
> ([**Bau97**])

## 3. Monoalphabetische Verfahren

Im folgenden Kapitel werden Verfahren besprochen, die mit Hilfe eines Musters bzw. eines wahrscheinlichen Wortes einen Einstieg in monoalphabetische Verfahren Chiffrierverfahren versuchen. Es wird auch auf die Möglichkeit der maschinellen Exhaustion einer Belegung eines Musters eingegangen.

**3.1. Anatomie der Sprache: Muster.** Sprache enthält ein schwer ausrottbares Gerüst von Gesetzmäßigkeiten. Von besonderer Wichtigkeit sind dabei Wiederholungsmuster.

| V | VF | VFK | VFKR | VFKRQ | VFKRQG | VFKRQGX | VFKRQGXQ |
|---|---|---|---|---|---|---|---|
| V | VF | VFK? | | | | | |
| W | WG | WGL | WGLS? | | | | |
| X | XH | XHM? | | | | | |
| Y | YI | YIN? | | | | | |
| Z | ZJ | ZJO | ZJOV | ZJOVU? | | | |
| A | AK | AKP? | | | | | |
| B | BL | BLQ? | | | | | |
| C | CM | CMR? | | | | | |
| D | DN | DNS? | | | | | |
| E | EO | EOT | EOTA | EOTAZ? | | | |
| F | FP | FPU | FPUB | FPUBA? | | | |
| G | GQ | GQV? | | | | | |
| H | HR | HRW? | | | | | |
| I | IS | ISX | ISXE | ISXED? | | | |
| J | JT? | | | | | | |
| K | KU | KUZ | KUZG? | | | | |
| L | LV? | | | | | | |
| M | LV | LVA | LVAH | LVAHG? | | | |
| N | MW | MWB? | | | | | |
| O | OY | OYD | OYDK? | | | | |
| P | PZ | PZE | PZEL | PZELK | PZELKA | PZELKAR? | |
| Q | QA? | | | | | | |
| R | RB | RBG | RBGN | RBGNM? | | | |
| S | SC | SCH | SCHO | SCHON | SCHOND | SCHONDU | SCHONDUN |
| T | TD | TDI | TDIP | TDIPO | TDIPOE | TDIPOEV? | |
| U | UE | UEJ? | | | | | |

TABELLE 2. Exhaustion

3.1.1. *Invarianzsatz und Muster.*

> **Invarianzsatz** Für alle monoalphabetischen, funktionalen einfachen Substitutionen, insbesondere auch für alle linearen monoalphabetischen einfachen Substitutionen gilt: *Wiederholungsmuster der Einzelzeichen innerhalb des Textes bleiben erhalten.* ([**Bau97**])

Wie wir in Tabelle 3 sehen, bleibt das Wiederholungsmuster des Buchstaben **e** immer erhalten. Auch für monoalphabetische polygraphische Substitutionen bleiben unter Beachtung der Polygrammfugen die Muster der Polygrammwiederholungen erhalten. Diese Muster werden allerdings durch Transpositionen oder homophone und polyalphabetische Substitutionen zerstört.
Muster bezeichnet man in Normalform mit Ziffernfolgen, in denen jede erstmals auftretende Ziffer keine größere zur linken hat. So wird z.B. aus dem Muster **NRGRN** die Ziffernfolge **12321**. Je länger das Muster wird desto weniger Möglichkeiten der Belgung gibt es, so ist im Englischen das Muster **12132435** nur mit dem Wort **fiftieth** zu belegen. Falls diese Muster also in einem Geheimtext vorkommen sollte, und man das Chiffrierverfahren richtig vermutet hat, wären in diesem Fall schon 5 Buchstaben dechiffriert und ein guter Angriffspunkt gegeben. Deshalb sollten Worte und Phrasen mit auffälligen Muster durch den Chiffrierer vermieden bzw. umschrieben werden.

3.1.2. *Verwendung des Invarianzsatzes.* Der Invarianzsatz kann sowohl negativ als auch positiv genutzt werden.
Negativ angewendet kann man monoalphabetische, funktionale einfache Substitutionen ausschließen, nämlich genau dann wenn keine gängigen Muster im Chiffrat vorkommen.
Wenn man allerdigs Grund zur Annahme hat, dass genau ein solches Verfahren benutzt wurde, kann man den Invarianzsatz positiv nutzen um nach Mustern zu suchen und damit einen Einstieg

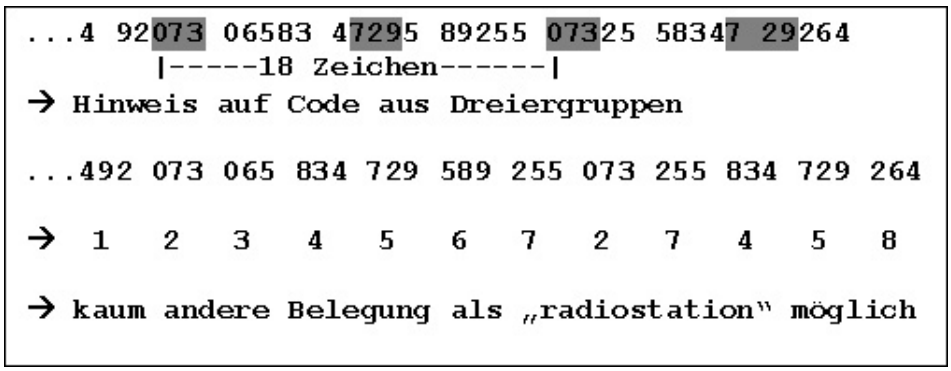



ABBILDUNG 1. Beispiel zur Mustersuche

| Klartext | w | i | n | t | **e** | r | s | **e** | m | **e** | s | t | **e** | r |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CAESAR chiffriert | Z | L | Q | W | **H** | U | V | **H** | P | **H** | V | W | **H** | U |
| revertiertes Alphabet | D | R | M | G | **V** | I | H | **V** | N | **V** | H | G | **V** | I |
| permutiertes Alphabet | V | A | H | O | **R** | M | N | **R** | G | **R** | N | O | **R** | M |

TABELLE 3. Beispiel zum Invarianzsatz

in die Chiffrierung zu erhalten. Dies wird auch an Abbildung 1 gezeigt. In diesem Beispiel hatte der Dechiffrierer allerdings den glänzenden Einfall „radiostation" herauszulesen, worauf man nicht unbedingt kommen muss.

**3.2. Die Methode des wahrscheinlichen Wortes.** Das Prinzip dieser Methode ist, dass man im Chiffrat nicht nach einem auffälligen Muster sucht, sondern sich ein Wort auswählt das „wahrscheinlich" im Klartext vorkommt. Von diesem Wort bildet man nun das Muster und durchsucht das ganze Chiffrat nach Stellen an denen dieses Muster vorkommt. Anstatt eines einzelnen Wortes kann man auch nach ganzen Phrasen suchen. Häufig benutzte Phrasen sind zum Beispiel: „Hochachtungsvoll Ihr" oder im 2.Weltkrieg „Oberkommando der Wehrmacht".
Um wahrscheinliche Wörter zu finden ist es sehr von Vorteil einen Einblick und ein gutes Einfühlungsvermögen in die gegnerische Situation zu haben. So ist z.B. nach einem Angriff oder einem Feuergefecht eine Nachricht zu erwarten, in der wahrscheinlich das Wort 'Angriff' oder 'Gefecht' vorkommt. Somit hätte man schon zwei wahrscheinliche Wörter nach denen man suchen könnte.

**3.3. Maschinelle Exhaustion der Belegung eines Musters.** Auch bei der Mustersuche und der Methode des wahrscheinlichen Wortes ist eine Rechnerunterstützung sehr angebracht. Zum einen kann ein Rechner sehr viel schneller ein Chiffrat nach dem Muster eines wahrscheinlichen Wortes durchsuchen und alle möglichen Stellen durchprobieren. Zum anderen kann, wenn kein wahrscheinliches Wort vorliegt, unter hinzuziehen maschinell erstellter Listen mit Wörtern gleichen Musters (z. B. aus Lexika) sehr schnell nach seltenen oder wiederholten Mustern gesucht werden. Im zweiten Fall ist ein systematisches Vorgehen gefragt:
Zuerst sucht man nach Mustern die nur wenige oder eine einzige Belegungen erlauben und wendet dann die Methode der Exhaustion auf diese Stellen an.
Beim Auffinden mehrerer hintereinander liegenden Muster schließen sich häufig einige Belegungen aus und der Suchraum wird dadurch eingeschränkt. Man sollte erwarten:

> In einem monoalphabetisch chiffrierten Geheimtext ist die Anzahl der vorkommenden Muster proportional zur Länge des Textes. Die Kopplung der Muster wächst jedoch mindestens quadratisch mit der Länge, so dass die aus der Kopplung der Muster hervorgehenden Einschränkungen rasch zunehmen und der Suchraum der Exhaustion entsprechend eingeengt wird. ([**Bau97**])

| WORTLÄNGE | Anzahl möglicher Pangramme |
|---|---|
| 11 | einige Hundert |
| 12 | einige Dutzend |
| 13 | einige wenige |
| 14 | 1 ⇒ "ambidextrously" |

TABELLE 4. Anzahl der Pangramme für bestimmte Wortlängen (im Englischen)

| Wortlänge n | Trefferwahrscheinlichkeit P (in %) |
|---|---|
| 1 | 96,15 |
| 2 | 92,45 |
| 3 | 88,90 |
| 4 | 85,48 |
| : | : |
| 8 | 73,07 |
| 16 | 53,39 |
| 32 | 28,51 |
| 64 | 8,13 |
| 128 | 0,66 |

TABELLE 5. Gesamt-Trefferwahrscheinlichkeit der Negativ-Mustersuche (N=26)

```
manyo rgani zatio nsrel yonco mpute rsand datac …
GRSUZ TLDSZ NKWNE RDPFB OVVQN OBKYI QNJRC QLIFR …
```

ABBILDUNG 2. ENIGMA chiffrierter Text

**3.4. Pangramme.** Als guter Einstieg in eine Chiffrat sind Listen mit langen Wörtern in denen jeder Buchstabe nur einmal vorkommt. Solche Wörter nennt man auch Pangramme. Falls Pangramme durch den Chiffrierer nicht entfernt wurden, ist nur eine Exhaustion in kleinem Umfang nötig, um einen Einstieg in das gegnerische System zu erlangen, da die Anzahl der Pangramme mit wachsender länge stark abnimmt(siehe Tabelle 4).

## 4. Polyalphabetische Verfahren

Dieses Kapitel über Verfahren zur Brechung polyalphabetischer Verfahren behandelt am Anfang wieder wahrscheinliche Wörter und Mustersuche. Es werden auch Verfahren zur Dechiffrierung polyalphabetischer Chiffrierungen mit nichtperiodischen Schlüsseln und ENIGMA chiffrierter Texte angesprochen.

**4.1. Wahrscheinliche Wörter.** Da die Übereinstimmung von Mustern notwendigerweise auf monoalphabetische Verfahren beschränkt ist geht man im Falle einer vermuteten polyalphabetischen Chiffrierung einen anderen Weg.

4.1.1. *Nichtübereinstimmung.* Für eine weite Klasse von polyalphabetischen, endomorphen Chiffrierungen, nämlich genau diejenigen, die die Bedingung erfüllen, dass in jedem Alphabet kein Zeichen durch sich selbst chiffriert wird, ist eine Negativsuche nach einem Muster möglich. Dies sind z. B. PORTA und MULTIPLEX Verfahren. D. h. man schließt alle Lagen innerhalb des Chiffrats aus, an denen für das wahrscheinliche Wort diese Regel verletzt wird, und erhält damit alle möglichen Lagen, die dann wieder exhaustiv zu untersuchen sind ob es Treffer oder Fehltreffer sind.

4.1.2. *Negativ-Mustersuche.* Die Gesamttrefferwahrscheinlichkeit der Negativsuche nach einem Wort der Länge $n$ ist

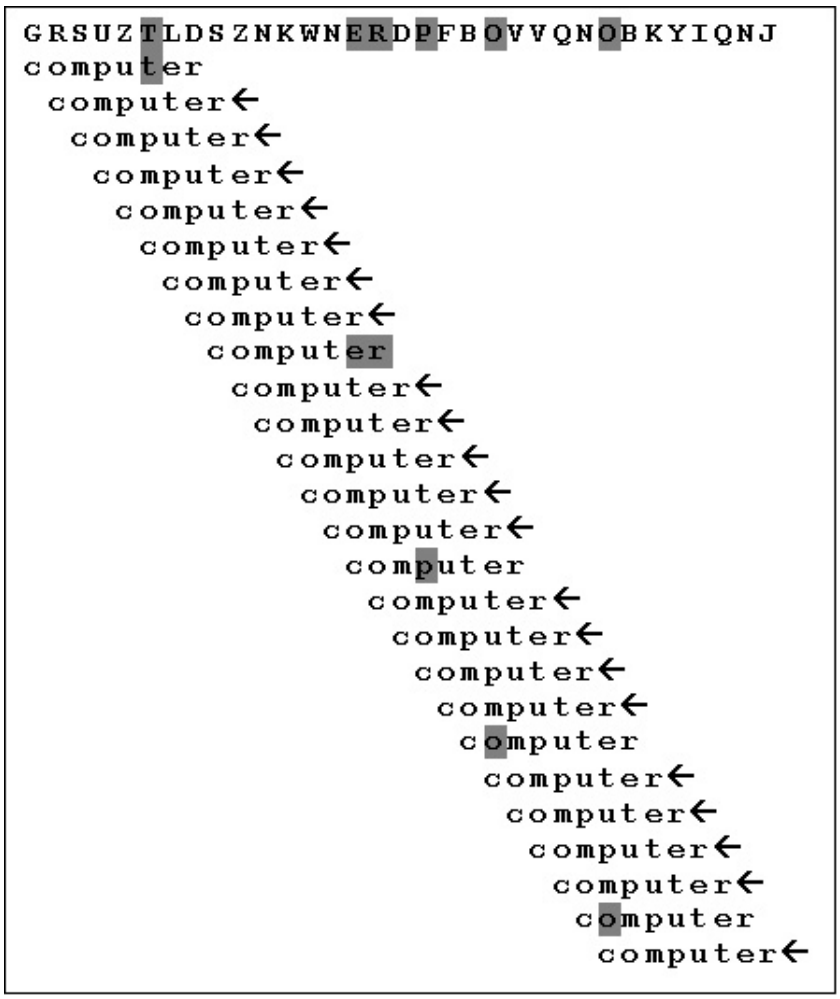



ABBILDUNG 3. Beispiel zur Negativ-Mustersuche

```
manyo rgani zatio nsrel yonco mpute rsa…
PRSBF IOZGP LYCNE EDGWS ATVQD OBKJQ CMP…
```

ABBILDUNG 4. PORTA chiffrierter Text

$$P = (1 - \frac{1}{N})^n.$$

Dies ist in Tabelle 5 für einige $n$ dargestellt.
Für kurze Wörter erhält man natürlich viele mögliche Lagen, und somit auch viele Fehltreffer (siehe Abbildung 3). Hier liegt der ENIGMA chiffrierte Text aus Abbildung 2 zugrunde. Es wurde eine Negativsuche nach dem vermuteten 8-Zeichen-Wort *computer* durchgeführt. Aufgrund von Tabelle 5 sind für die ersten 26 Positionen ungefähr $19 = 26 * 0.7307$ mögliche Lagen zu erwarten. Tatsächlich sind aber 21 Lagen nicht auszuschließen und man erhält somit 20 Fehltreffer.

4.1.3. *Binäre negative Mustersuche.* Falls es sich bei der Chiffrierung um eine (polyalphabetische) Involution(z. B. PORTA) handelt, d.h.eine Hälfte des Alphabets wird auf die andere Hälfte abgebildet und umgekehrt, bietet sich die Möglichkeit der binären Mustersuche an.

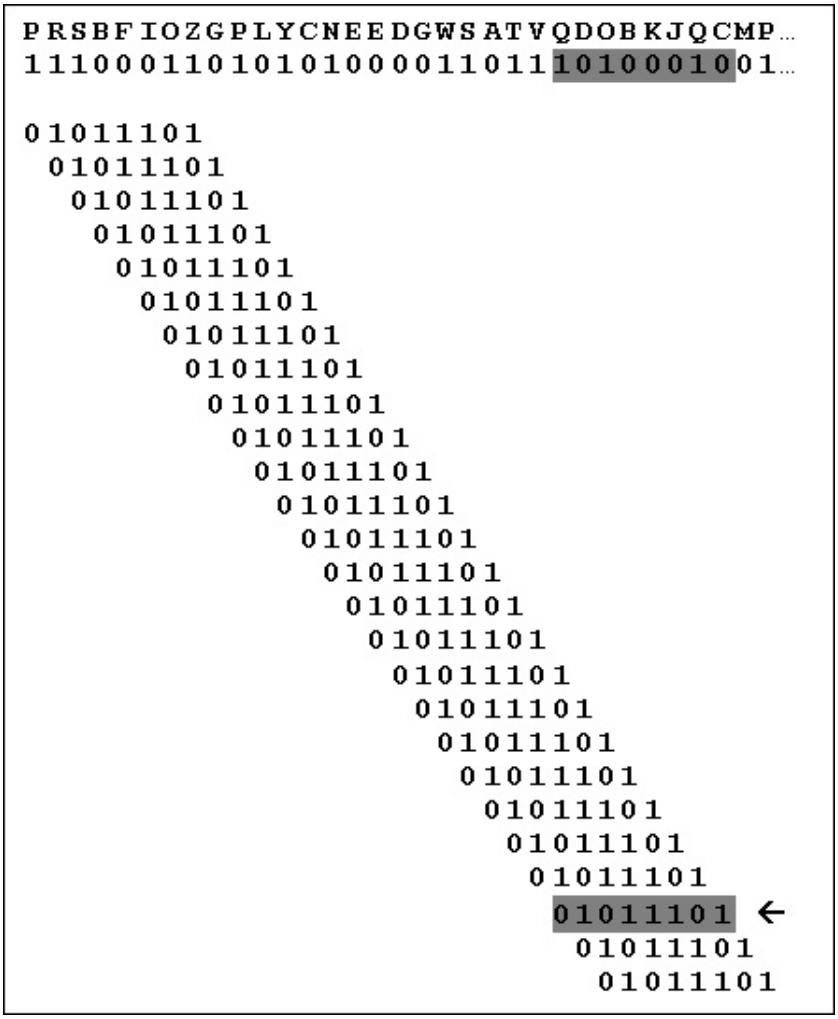



ABBILDUNG 5. Beispiel zur binären Negativ-Mustersuche

Das binäre Muster eines Wortes ergibt sich wie folgt:

0 $\Leftrightarrow$ das Zeichen ist aus der ersten Hälfte des Alphabets
1 $\Leftrightarrow$ das Zeichen ist aus der zweiten Hälfte des Alphabets

Die binäre Mustersuche verläuft dann wie in Abbildung 5. Hier liegt derselbe Text wie in Abbildung 2 zugrunde, allerdings diesmal mittels PORTA chiffriert(siehe Abbildung 4). Wie man sieht führt dieses Verfahren nur zu einem Treffer, nämlich genau der Position des Wortes im Klartext.

**4.2. Mustersuche bei bekannten Alphabeten - DeViaris.** Diese allgemeine Methode zur Mustersuche bei Schieber- und Zylindergeräten setzt nun nicht mehr voraus, dass die Alphabete voll zyklisch oder echt involutorisch sind, allerdings ist jetzt der Besitz des Chiffriergerätes nötig. Für diesen speziellen Falll der Schieber- und Zylindergeräte kommt die scheinbare Komplikation der Entzifferung durch die Polyphonie des Verfahrens hinzu.
Man nimmt nun an, dass die Chiffre in der k-ten Zeile nach dem Klartextstück (in der k-ten-Generatrix) abgelesen wurde. Ausserdem sei das wahrscheinliche Wort kurz genug um nicht zerrissen zu werden. Man wird mit kleinem k beginnen und hat höchstens zwei Dutzend Fälle durchzuprobieren.
Bei diesem Verfahren wird für ein wahrscheinliches Wort und ein gegebenes festes k festgestellt, welche Zeichen in der k-ten Generatrix auf der i-ten Scheibe/Lineal stehen. Mit dieser Information

fährt man nun am Chiffrat vorbei und kann jetzt wieder mögliche und unmögliche Lagen bestimmen. Auch hierbei kann es bei kurzem wahrscheinlichen Wort wieder mehrere mögliche Lagen geben, die wieder exhaustiv zu untersuchen sind.
Ist das gewählte wahrscheinliche Wort lang, so besteht auch die Möglichkeit keine passende Lage zu finden. In diesem Falle ist zu einer anderen Generatrix überzugehen.
Gelingt es für keine Generatrix eine mögliche Lage zu bestimmen, so kommt das gewählte wahrscheinliche Wort entweder nicht vor, oder es wurde zerrissen. Dies macht man so lange, bis man eine korrekte mögliche Lage gefunden hat.
Dies kann man überprüfen, indem man die selbe Dechiffrierung im Abstand von n Zeichen, wobei n die Anzahl der Lineale/Zylinder ist, noch einmal anwendet und sich dort wieder eine Entzifferung ergibt. Somit hat man die Lage der ersten Lineale/Zylinder gefunden.
Aus dieser teilweisen Entzifferung ergeben sich nun Bruchstücke von Wörtern, die man logisch ergänzen kann und somit immer neue wahrscheinliche Wörter erhält, aus denen immer mehr Lagen eindeutig besimmt werden können. Zum Schluß hat man dann alle Lagen für alle Lineale/Zylinder gefunden und kann den ganzen Text dechiffrieren.

**4.3. Zick-Zack-Exhaustion.** Wird bei einem polyalphabetischen Verfahren ein Schlüssel in verständlicher Sprache verwendet, darf man erwarten, dass es darin wahrscheinliche Wörter gibt. Sind bei diesem Verfahren das Referenzalphabet oder die unabhängigen Alphabete, und somit die Chiffrier- bzw. Dechiffrierschritte bekannt, und besteht Kenntnis welche Schlüsselzeichen diese Alphabete kennzeichnen, so kann man eine teilweise Dechiffrierung mit Hilfe der Exhaustion versuchen.
Dies würde z.B für eine VIGENÈRE Chiffrierung mit

$$c_j = p_j + k_j$$

über $\mathbb{Z}_{26}$ die Dechiffrierung

$$p_j = c_j - k_j$$

ergeben.
Wobei $p_j$ das Klartextzeichen, $c_j$ das Geheimtextzeichen und $k_j$ das Schlüsselzeichen ist.
Wenn es sich jetzt aber auch um eine Shannonsche Chiffrierung handelt, d.h. wenn Geheimtext - und Klartextzeichen das Schlüsselzeichen eindeutig bestimmen, kann das Verfahren auch umgekehrt anwenden. Dies ist bei linearen Substitutionen mit bekanntem Referenzalphabet ebenfalls der Fall. Dies ergibt nun

$$k_i = c_i - p_i.$$

Aber weder das eine noch das andere Verfahren führt selten für sich alleine zum Erfolg. Wenn man nun beide Verfahren abwechselnd in einem Zick-Zack-Wechselspiel anwendet, erhält man ein leistungsfähiges Verfahren, die Zick-Zack-Exhaustion. Selbst nichtperiodische Schlüssel verhindern die Zick-Zack-Exhaustion nicht, denn es kommt nur darauf an, das der Schlüsseltext genügend Redundanz mit dem Klartext aufweist.

**4.4. Isomorphiemethode.** Um ROTOR-Chiffrierungen wie die der kommerziellen Enigma zu brechen verwendet man die schon 1937 von Alfred Dillwyn Knox verwendete Isomorphie-Methode die allerdings bei neueren Enigma's mit Steckerbrett nicht mehr anwendbar ist. Die ROTOR-Chiffrierungen haben nämlich eine spezifische Regelmäßigkeit, die einen Angriffsweg eröffnet.
Sei die polyalphabetische Substitution von der Form

$$c_i = p_i S_i U S_i^{-1},$$

mit den bekannten Alphabeten $S_i$ (was man bei einer kommerziellen Enigma voraussetzen kann) , und die Reihenfolge der Alphabete sei bekannt. Der unbekannte Schlüssel besteht aus dem Anfangsindex i der Folge und höchstens noch U. Wenn man nun die isomorphen Folgen $c_i S_i$ und

$p_iS_i$bildet gilt

$$c_iS_i = p_iS_i \cdot U$$

und die Folge $(c_iS_i)$ ist monoalphabetisches Bild der Folge $(p_iS_i)$ unter der Substitution U. Diese kann mit Hilfe einer vollständigen Klartext-Geheimtext-Kompromittierung genügend langer Texte aufgedeckt werden.
Zur Kryptanalyse ist es jetzt nur noch nötig für ein wahrscheinliches Wort einen passenden Anfangsindex i zu finden, so dass für das Wort $(p_i, p_{i+1}, ..., p_{i+k})$ gilt:

$$c_iS_i, c_{i+1}S_{i+1}, ..., c_{i+k}S_{i+k} \textit{ ist isomorph zu } p_iS_i, p_{i+1}S_{i+1}, ..., p_{i+k}S_{i+k}.$$

Unter den gefundenen Indizes ist nun sicher auch der Richtige, wenn das wahrscheinliche Wort phasenrichtig zum Geheimtext steht. Hier kommt im übrigen wieder die Exhaustion zum Zuge, denn mit ihr muss die Phasenlage des wahrscheinlichen Wortes bestimmt werden.
Die oben gemachten Voraussetzungen liegen insbesondere im Fall der rotierten Standardalphabete mit $\{\rho^{-i}R\rho^i : i \in \mathbb{N}\}$ vor.
Dies gilt nun auch für die kommerzielle ENIGMA ohne Steckerbrett, denn alle Rotoren sind aufgeklärt und für die Walzenlage einer 3-Rotoren ENIGMA gibt es genau sechs Möglichkeiten. Desweiteren ist die Isomorphie, aufgrund der Bauweise der Umkehrwalze, auf involutorisches U beschränkt, was die Möglichkeit für Fehltreffer verringert.
Ein etwas längeres Beispiel findet sich in [**Bau97**] S. 268ff.

## 5. Verdeckte Klartext-Geheimtext-Kompromittierung

Falls eine Klartext-Geheimtext-Kompromittierung vorliegt, also Klar- und Geheimtext in die Hände des Dechiffrierers fallen, kann in glücklichen Fällen der Schlüssel gefunden werden und damit ein tiefer Einbruch in das gegnerische System erzielt werden.
Eine verdeckte Klartext-Geheimtext-Kompromittierung entsteht, wenn der selbe Klartext zwei unterschiedlichen Chiffriersystemen unterworfen und verschickt wird. Gerät der Dechiffrierer nun in Besitz beider Chiffrate und kann eines der beiden Systeme (meistens das ältere) brechen, liegt eine normale Klartext-Geheimtext-Kompromittierung mit all ihren Folgen vor.
Damit kann das ander System (meistens das neuere) vollständig aufgedeckt werden.
Dazu sagte *Hüttenhain*:

> *Es dürfen also keine Chiffrierverfahren verwendet werden, die gegen Klar-Geheim-Kompromisse anfällig sind.*

Diese Aussage hat zur Folge, dass viele klassiche und etliche moderne Verfahren aufgegeben werden müssen und man neue bessere Verfahren entwickeln muss.

## 6. Zusammenfassung und Ausblick

Fast alle der hier vorgestellten Dechiffrierverfahren basieren auf der Schaffung von Keimen die zu Inseln werden und schließlich zu einer kompletten Lösung zusammenfließen. Auch ist bei allen Verfahren gewisses „Glück“ von nöten, so zum Beispiel bei der Suche nach einem wahrscheinlichen Wort oder der Belegung eines Musters, allerdings kann man zusammenfassend sagen, dass diese „alten“ Verfahren mit den heutigen technischen Mittel leicht bzw. in annehmbarer Zeit gebrochen werden können.
Insbesondere bei der Mustersuche und dem durchprobieren aller möglichen Lagen spielen Computer mit ihrer Geschwindigkeit eine große Rolle.
Auch die Tatsache, dass diese Chiffrierverfahren heute genauestens dokumentiert sind und jeder diese Unterlagen einsehen kann, spricht nicht für ihre Sicherheit. Auch die Auswahl von Schlüsseln war bei diesen Verfahren nicht sehr sicher.
Dies hat sich aber bis heute schon verändert, man benutzt bei vielen modernen Chiffrierverfahren öffentliche und private Schlüssel, sowie Schlüssel die ein Produkt aus zwei großen Primzahlen sind, wie sicher oder unsicher diese sind werden wir in späteren Kapiteln sehen.

KAPITEL 5

# Kryptologische Grundlagen und grundlegende Protokolle

**Lidia Angelova**

## 1. Einführung

Dieses Dokument stellt neben einigen grundlegenden kryptographischen Mechanismen auch wesentliche Protokolle der modernen Kryptographie mit ihren unterschiedlichen Anwendungsgebieten vor. Hierzu gehören neben der Authentifikation, die Schlüsselvereinbarung, das Verschlüsseln sowie das Erzeugen blinder Signaturen. Die mathematische Grundlage, die ihnen obliegt, wird anhand einiger Beispiele demonstriert. Zudem werden noch die Angriffsmöglichkeiten auf die einzelnen Protokolle behandelt.

## 2. Kryptologische Grundlagen

In diesem Kapitel werden grundlegende kryptologische Mechanismen dargestellt, die ihre Verwendung in komplexen Protokollen finden. Der älteste Zweig der Kryptographie beschäftigt sich mit der Geheimhaltung von Nachrichten durch Verschlüsselung. Ziel hierbei ist die Lesbarkeit der Nachricht einzig und allein durch den berechtigten Empfänger. Zur Realisierung muß der Empfänger dem Angreifer eine Information, welche geheimzuhalten ist, voraushaben. Diese wird **Schlüssel** genannt; mit seiner Hilfe kann der Empfänger den empfangenen Geheimtext **entschlüsseln**. In der Kryptographie unterscheidet man zwei Arten.

**2.1. Symmetrische Verschlüsselung.** In der klassischen Kryptographie wird der Nachrichtenaustausch über einem dem Sender und Empfänger bekannten Schlüssel vollzogen. Verfahren mit dieser Eigenschaft nennt man **symmetrische** Verschlüsselungsverfahren. Im Alltag kann man sich den Vorgang so vorstellen:

- Sender legt die Nachricht in einen „Tresor“und verschließt diesen mit Hilfe seines Schlüssels.
- Sender schickt Tresor samt Inhalt an den Empfänger.
- Nur dieser hat einen Zweitschlüssel, um den Tresor zu öffnen und die Nachricht zu lesen

Realisiert wird es aber durch wesentlich sichere und kostengünstige mathematische Methoden. Genauer gesagt: durch einen symmetrischen Verschlüsselungsalgorithmus, der aus einer **Funktion f** mit den Eingabewerten, dem **Schlüssel k** und dem **Klartext m** , und einer Ausgabe, dem **Geheimtext c** besteht.
Schreibweise: $c = f(k, m)$ alternativ: $c := f_k(m)$
Anforderungen:
Die Funktion f muß **umkehrbar** sein, d.h. $\exists \mathbf{f}^\star$ mit $\mathbf{f}^\star(\mathbf{k}, \mathbf{c}) = \mathbf{m}$.

2.1.1. *Sicherheit.* Da grundsätzlich das **Kerkhoffsche Prinzip** gilt, welches besagt, daß sowohl Ver- als auch Entschlüsselungsfunktion bekannt sind, muß eine Verschlüsselungsfunktion folgende Angriffe überstehen, um als **sicher** zu gelten:

- **Ciphertext-only attack** Der Angreifer kennt eine begrenzte Anzahl von Geheimtexten und möchte daraus die zugehörigen Klartexte bzw. den verwendeten Schlüssel berechnen.
- **Known-plaintext attack** Der Angreifer kennt eine begrenzte Anzahl von Geheimtexten mit den zugehörigen Klartexten und möchte daraus den verwendeten Schlüssel bzw. den Klartext zu einem weiteren Chiffretext berechnen.
- **Chosen-plaintext attack** Der Angreifer hat sich Zugang zu der mit dem Schlüssel $k$ parametrisierten Verschlüsselungsfunktion $f$ verschafft. Er kann den Schlüssel $k$ zwar nicht auslesen, aber bestimmte von ihm ausgewählte Klartexte (z.B. spezielle Klartexte wie

**100000000**) mit Hilfe von $f_k$ verschlüsseln. Mit seiner Hilfe möchte er andere Geheimtexte entschlüsseln bzw. den Schlüssel $k$ berechnen. (Dieser Angriff erscheint zunächst etwas akademisch, er stellt aber bei der Public-Key Kryptographie eine echte Bedrohung dar.)

- **Chosen-ciphertext attack** Der Angreifer hat sich Zugang zu der mit dem Schlüssel $k$ parametrisierten Entschlüsselungsfunktion $f^\star$ verschafft. Er kann den Schlüssel $k$ zwar nicht auslesen, aber bestimmte von ihm ausgewählte Geheimtexte mit Hilfe von $f_k^\star$ entschlüsseln. Auf diese Art versucht er den Schlüssel zu berechnen.

Die Algorithmen zur symmetrischen Verschlüsselung von Daten unterteilt man in **Blockchiffre** und **Stromchiffre**.

2.1.2. *Blockchiffre.* Bei einer Blockchiffrierung wird der Klartext in Blöcke $m_1, m_2, m_3, \ldots$ fester Länge eingeteilt (eine typische Zahl ist 64 Bit), und jeder Block $m_i$ wird unter Verwendung des Schlüssels einzeln verschlüsselt:

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

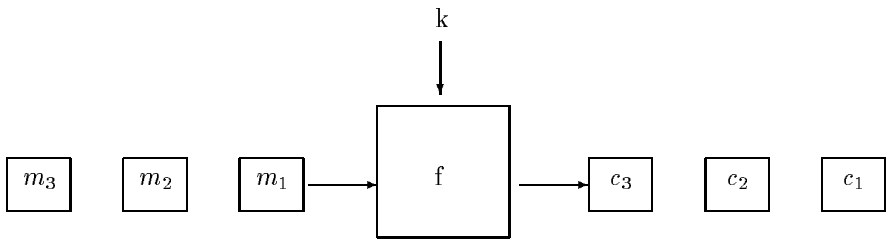



ABBILDUNG 1. Blockchiffre

Beispiele für Blockchiffrierungen

- *DES* (Data Encryption Standard)
  Blocklänge: 64 Bit
  Schlüssellänge: 56 Bit
  [Am 19. Januar 1999 wurde ein DES-Schlüssel mit einer known-plaintext attack durch vollständig Suche innerhalb von 22 Stunden und 15 Minuten berechnet. Ein Nachfolgealgorithmus wird derzeit standardisiert.]
- *IDEA*(International Data Encryption Algorithm) Blocklänge: 64 Bit
  Schlüssellänge: 128 Bit

2.1.3. *Stromchiffre.* Bei einer Stromchiffrierung wird ein Schlüsselstrom erzeugt, der die gleiche Länge wie der Klartext hat, so daß jedes Geheimtextzeichen aus einer Verknüpfung eines Klartextzeichens mit einem Schlüsselzeichen hervorgeht.
Der Prototyp aller Stromchiffren ist das **One-time-pad**. Hierbei liegen sowohl Nachricht als auch der zufällig gewählte Schlüssel als Folge von Bits vor.
*Verschlüsselung*: Entsprechende Klartext- und Schlüsselbits werden modulo 2 addiert.

$$0 \oplus 0 = 0, 0 \oplus 1 = 1 \oplus 0 = 1, 1 \oplus 1 = 0$$

In diesem Fall verlaufen Ver- und Entschlüsselung identisch, es ist also $f^\star = f$. Bei einmaliger Verwendung ist dieses Verfahren absolut sicher („perfekt“). Es hat allerdings den Nachteil, daß der Schlüssel genauso lang sein muß wie der Klartext. Mehrfaches Benutzen desselben Schlüssels führt zu einer Sicherheitsminderung - das System kann per Known-Plaintext-Angriff gebrochen werden.

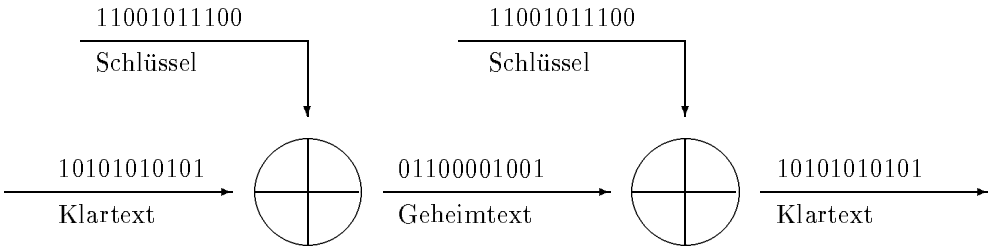



ABBILDUNG 2. One-time-pad

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

**2.2. Asymmetrische Verschlüsselung.** Jahrtausendelang ging man davon aus, daß auch der Sender einen geheimen Schlüssel, und zwar den gleichen wie der Empfänger, benötigt.
Die Geburtsstunde der **asymmetrischen Kryptographie** (oder **Public-Key-Kryptographie)** schlug und die daran anschließenden Überlegungen führten W. Diffie und M. Hellman 1976 zum Konzept des asymmetrischen Verschlüsselungsverfahrens.
Das Verfahren ist mit einem Briefkasten assoziierbar. Jeder kann einem Empfänger eine verschlüsselte Nachricht schicken, ohne irgendeine Geheiminformation zu besitzen. Die Entschlüsselung ist aber nur durch den Empfänger möglich.
**Vorteile:**

- **kein Schlüsselaustausch** (ideal für offene Kommunikation)
- **weniger Schlüssel:** jeder Teilnehmer erhält zwei Schlüssel (bei der symmetrischen Verschlüsselung: $\frac{n\cdot(n-1)}{2}$ bei n Teilnehmer)
- **Problemloses Hinzufügen** neuer Teilnehmer
- nur **Entschlüsselung** verläuft in **sicherer Umgebung**
- Möglichkeit zur **elektronischen Unterschrift**

Jeder Teilnehmer des Systems erhält:

- **privaten Schlüssel** $d = d_T$ oftmals auch **geheimer Schlüssel** bezeichnet
- **öffentlicher Schlüssel** $e = e_T$ ist möglichst vielen Personen zugänglich

Der Algorithmus $f$ ordnet unter einem öffentlichen Schlüssel $e$ jedem Klartext $m$ einen Geheimtext $c = f_e(m)$
zu. Umgekehrt ordnet $f$ unter jedem privaten Schlüssel $d$ jedem Geheimtext $c$ einen Klartext

$$m' = f_d(c)$$

zu. Dabei müssen folgende Eigenschaften erfüllt sein:
Korrekte Entschlüsselung: Reproduktion des korrektes Klartextes für alle Klartexte, d.h.

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

Public-Key-Eigenschaft: Kein Rückschluß durch den öffentlichen Schlüssel $e$ auf den privaten Schlüssel $d$ möglich

Möchte man einem Teilnehmer $T$ eine Nachricht senden, ermittelt man zuerst seinen öffentlichen Schlüssel $e = e_T$. Anschließend wird die Funktion $f_e$ auf die zu verschlüsselnde Nachricht $m$ angewandt und der so erhaltene Geheimtext $c = f_e(m)$ an T geschickt. Nur dieser kann mit Hilfe seines privaten Schlüssels $d = d_T$ die Funktion $f_d$ bilden und damit $m$ entschlüsseln:

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

.

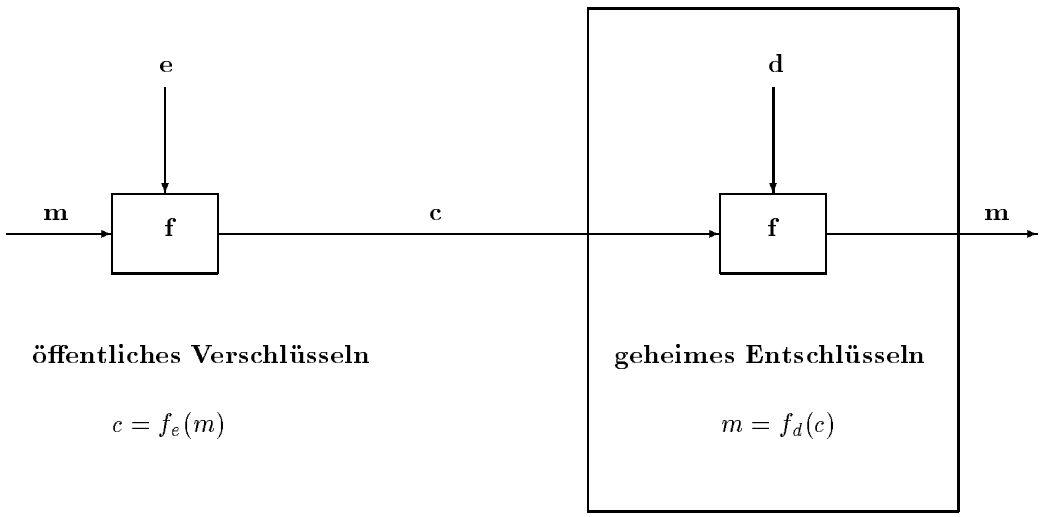



ABBILDUNG 3. Funktionsschema der asymmetrischen Verschlüsselung

### 2.3. Einwegfunktionen und ihre Verwendungen.

2.3.1. *Definitionen.* Eine **Einwegfunktion** ist eine Abbildung $f$ einer Menge $X$ in eine Menge $Y$, so daß $f(x)$ für jedes Element von $X$ leicht zu berechnen ist, während es für (fast) jedes $y$ aus $Y$ extrem schwer ist, ein Urbild $x$ zu finden bzw. überhaupt nicht. Eine Einwegfunktion ist eine Funktion, die einfach auszuführen, aber schwer - praktisch unmöglich zu invertieren ist.
Ist eine Einwegfunktion bijektiv (Permutation), dann spricht man von einer **Einwegpermutation**.
Eine Einwegfunktion heißt **kollisionsfrei**, wenn

$$\boxed{\nexists x, x' \in X, x \neq x' f(x) \neq f(x')}$$

Einwegfunktionen spielen sowohl in der theoretischen als auch in der praktischen Kryptographie eine entscheidende Rolle. Zum Beispiel lassen sich fast alle kryptographische Begriffe auf den Begriff der Einwegfunktion zurückführen. Bis heute weiß man nicht, ob es Einwegfunktionen überhaupt gibt, allerdings gilt:

$$\boxed{\textbf{Einwegfunktionen existieren} \Longleftrightarrow \textbf{P} \neq \textbf{NP}}$$

Diese aus der Komplexitätstheorie stammende Vermutung, ist noch unbewiesen. Nach heutigem Wissensstand sind die diskreten Exponentialfunktionen ebenso wie das Quadrieren modulo n (mit n = pq) Einwegfunktionen.
**Klasse P**

In diese Klasse fallen diejenigen Probleme, die mit einem polynomialem *Zeit*aufwand gelöst werden können.

**Klasse NP** („nichtdeterministisch polynomial“)

Die Klasse **NP** besteht aus denjenigen Problemen, bei denen die *Verifizierung* einer gegebenen Lösung mit polynomialem Zeitaufwand möglich ist. Ein wesentlicher Begriff ist hier die NP-Vollständigkeit. Ist ein Problem L in der Klasse NP und sind alle übrigen Probleme M derselben Klasse polynomial auf L reduzierbar, dann heißt das Problem L NP-vollständig. Kann ein NP-vollständiges Problem L schnell, also in polynomialer Zeit, gelöst werden, dann gilt sogar $P = NP$. NP-vollständige Probleme stellen also in gewisser Weise die schwierigsten Probleme aus NP dar.

2.3.2. *Kryptographische Hashfunktion.* Hashfunktionen dienen in der Kryptograhie zur Erstellung eines unmanipulierbaren „Fingerabdrucks“von Nachrichten. Eine **Einweg-Hashfunktion** ist eine kollisionsfreie Einwegfunktion, die Nachrichten beliebiger Länge auf einen Hashwert einer festen Länge (typischer Wert: 128 Bit) komprimiert. Einweg-Hashfunktionen werden auch **kryptographische Hashfunktionen** genannt.

2.3.3. *Trapdoor-Einwegfunktionen.* Indem Einwegfunktionen nur auf Berechnungen, die von *allen* Beteiligten ausgeführt werden dürfen, beschränkt sind, benötigt man in der modernen Kryptographie noch ein weiteres Konzept, nämlich das der Trapdoor-Einwegfunktion. Eine **Trapdoor-Einwegfunktion** ist eine Einwegfunktion, zu der es eine *Geheiminformation* („Geheimtür“, englisch „trapdoor“) gibt, wodurch die Funktion leicht invertierbar wird.

Zum Beispiel ist die *Potenzfunktion*

$$x \mapsto x^e \bmod n$$

für $n = pq$ eine Trapdoor-Einwegfunktion, denn ohne Kenntnis der Faktorisierung von $n$ ist es praktisch unmöglich, diese Funktion zu invertieren. Folglich ist die Trapdoorinformation die Faktorisierung von $n$. Man kann diese Funktionen auch mit anderen mathematischen Strukturen konstruieren. Im ElGamal-Verschlüsselungsverfahren (Abschnitt 2.2.2) muß der Empfänger einer verschlüsselten Nachricht den diskreten Logarithmus $t$ seines öffentlichen Schlüssels $\tau$ kennen, um den Schlüssel $k$ zur Entzifferung der Nachricht berechnen zu können.

2.3.4. *Commitment und Bit-Commitment.* Das Prinzip des *Commitments* (englisch für „Festlegung“) ist, daß eine Person $A$ eine Nachricht so hinterlegen kann, daß sie:

(1) von niemandem gelesen werden kann und
(2) von niemandem, insbesondere nicht von $A$, verändert werden kann.

Ein Commitment-Protokoll besteht aus den beiden Phasen des **Festlegens** und des **Öffnens** des Commitments. Eine Partei $A$ legt sich $B$ gegenüber auf einen Datensatz $m$ fest, indem sie $c = f(m)$ berechnet und $c$ an $B$ sendet. $A$ öffnet das Commitment, indem sie $B$ das Urbild $m$ von $c$ mitteilt. Man stelle sich folgende Situation vor: Auf ein Ausschreiben hin reichen mehrere Firmen ein Angebot ein, das auch einen Kostenvoranschlag enthält. Die Firmen befürchten aber, daß das erste Angebot ihren Konkurrenten bekannt wird und diese ihren Preis anschließend entsprechend anpassen. Daher möchten sie ihren Kostenvoranschlag geheim halten; andererseits müssen sie sich natürlich auch auf einen bestimmten Betrag festlegen .

Hier kommt die Eigenschaft der Einwegfunktion zu tragen, denn der naheliegenste Ansatz, die Nachricht einfach verschlüsselt zu hinterlegen, funktioniert nicht ohne weiteres. $A$ könnte nämlich seine Angaben nachträglich verfälschen.

Sonderfall: Bit-Commitment

In diesem Fall ist die zu verschlüsselnde Nachricht kurz, im Extremfall nur aus einem Bit bestehend. Das **Bit-Commitment-Problem** einer fehlenden Einwegfunktion auf die Menge $\{0, 1\}$ kann wie folgt gelöst werden: Unter Hinzunahme einer hinreichend großen Zufallszahl $r$ wird eine Einwegfunktion auf beide angewandt und der Funktionswert $f(b, r)$ veröffentlicht. Dabei muß die Kollisionsfreiheit nur für das erste Argument garantiert sein, d.h. für alle Zufallszahlen $r, s$ muß gelten: $f(0, r) \neq f(1, s)$. Eine bekannte Realisierung einer solchen Funktion $f$ ist die folgende:

$$f(m, r) := y^b r^2 \bmod n$$

wobei $n = pq$ das Produkt zweier großer Primzahlen und $y$ ein fester quadratischer Nichtrest modulo n ist.

**2.4. Digitale Signatur.** Das Ziel einer **digitalen Signatur** oder **elektronischen Unterschrift** liegt in der Realisierung wesentlicher Eigenschaften der handschriftlichen Unterschrift in elektronischer Form. Grundsätzlich unterscheidet man dabei folgende Eigenschaften:

- **Echtheitseigenschaft** Gewährleistung, daß das unterschriebene Dokument tatsächlich vom Unterschreibenden stammt, indem Unterschrift und die zu unterschreibende Erklärung auf einem Blatt stehen.
- **Identitätseigenschaft** Die Persönlichkeit einer digitalen Signatur basiert auf der Schwierigkeit diese zu fälschen, da sie nur von einer einzigen Person ausgestellt werden kann.
- **Abschlußeigenschaft** Sie signalisiert die Vollendung der Erklärung, indem die Unterschrift am Ende der Erklärung steht.
- **Verifikationseigenschaft** Verifikation einer erhaltenen Unterschrift durch Unterschriftenvergleich.
- **Warneigenschaft** (kryptographisch nicht realisierbar) Diese soll den Unterzeichner vor einem übereilten Signieren bewahren, indem die handschriftliche Unterschrift nicht nur aus einem Zeichen, z.B. Kreuz, besteht, sondern doch etwas komplexer ist.

Ein **Signaturschema** wird wie folgt definiert:
Jeder Teilnehmer $T$ des Systems erhält:

(1) Signaturfunktion $\mathbf{s_T}$ **geheim**
(2) Verifikationsfunktion $\mathbf{v_T}$ **öffentlich**

$T$ unterschreibt eine Nachricht $m$, indem er seine Signaturfunktion anwendet und daraus die *digitale Signatur*

$$sig = s_T(m)$$

erhält. Sowohl $m$ als auch $sig$ werden an einen beliebigen Empfänger gesandt. Mit Hilfe der Verifikationsfunktion $v_T$ ist dieser in der Lage, aus $m$ und $sig$ die Signatur auf ihre Korrektheit zu überprüfen. Bei diesem Verfahren ist die Verifikationsfunktion die Umkehrung der Signaturfunktion, es gilt also

$$v_T(sig) = m.$$

Aus den folgenden Gründen kommt in der praktischen Anwendung von Signaturfunktionen den Hashfunktionen eine wichtige Bedeutung zuteil: Zum einen sind die heutigen Signaturverfahren sehr langsam und zum anderen ist die digitale Signatur stets etwa so lang wie das unterschriebene Dokument. Allerdings ist eine Signatur fester Länge gewünscht. Dazu wird das Dokument durch eine öffentliche Hashfunktion $h$ auf einen Wert $h(m)$ fester Länge komprimiert und anschließend der Signaturfunktion unterworfen: $sig = s_T(h(m))$.
Bei der Verifikation bildet man ebenfalls zuerst $h(m)$ und überzeugt sich dann, daß $sig$ die zu $h(m)$ gehörende Signatur ist.

**2.5. Der RSA-Algorithmus.** Der **RSA-Algorithmus** wurde 1978 von R.Rivest, A.Shamir und L.Adleman erfunden, als sie zu zeigen versuchten, daß Public-Key-Kryptographie unmöglich sei. Seine Verwendungsmöglichkeiten liegen in der Verschüsselung als auch in der Erzeugung digitaler Signaturen.
Folgender mathematischer Sachverhalt wird dabei ausgenutzt:

$$\boxed{m^{k(p-1)(q-1)+1} \bmod n = m}$$

Dabei gilt $n = pq$ und $m \leq n$.
Zur Realisierung des RSA-Algorithmus wählt man zwei natürliche Zahlen $e$ und $d$, deren Produkt von der Form

$$e \cdot d = k(p-1)(q-1) + 1, k \in \mathbb{N}$$

ist. Somit gilt

$$(m^e)^d \bmod n = m \text{und} (m^d)^e \bmod n = m$$

Folgendermaßen geht man bei der Schlüsselerzeugung vor:
Jedem Teilnehmer werden zwei verschiedene große Primzahlen $p$ und $q$ zugeteilt, deren Produkt $n = pq$ und die Zahl $\varphi(n) = (p-1)(q-1)$ *(Eulersche Phi-Funtkion)* berechnet.
Dann wählt man eine natürliche Zahl $e$ mit der Eigenschaft $\text{ggT}(e, \varphi(n)) = 1$ und berechnet mit Hilfe des erweiterten euklidischen Algorithmus ein Zahl $d$ mit $e \cdot d \bmod \varphi(n) = 1$, also $e \cdot d = 1 + k(p-1)(q-1)$ für eine natürliche Zahl $k$. Somit bildet $d$ den geheimen Schlüssel und $e$ zusammen mit $n$ den öffentlichen Schlüssel des Teilnehmers. Die restlichen Zahlen $p$,$q$ und $\varphi(n)$ sind entweder ebenfalls geheimzuhalten oder können gegebenenfalls vernichtet werden, da sie während des Protokolls nicht genutzt werden.

2.5.1. *Der RSA als asymmetrisches Verschlüsselungsverfahren. Korrektes Entschlüsseln:* Die Korrektheit wird durch den Satz von Euler garantiert

| | |
|---|---|
| Verschlüsselung: | $f_e(m) := m^e \bmod n.$ |
| Entschlüsselung: | $f_d(c) := c^d \bmod n.$ |
| Korrektheit: | $f_d(f_e(m)) = (m^e)^d \bmod n = m.$ |

*Public-Key-Eigenschaft:* Mit der Kenntnis der Zahl $n$(allerdings ohne Kenntnis der Faktoren $p$ und $d$ oder der Zahl $\varphi(n) = (p-1)(q-1)$) läßt sich $d$ nicht aus $e$ berechnen.

2.5.2. *Der RSA als Signaturverfahren. Signaturfunktion:* Erstellen einer Signatur, indem auf die (eventuell gehashte) Nachricht $m$ der geheime Schlüssel $d$ des Unterschreibenden $T$ angewandt wird.

$$sig = s_T(m) := m^d \bmod n.$$

*Verifikationsfunktion:* Verifikation der Signatur *sig* durch Anwendung des öffentlichen Schlüssels $e$ des Unterschreibenden $T$ auf sie und anschließendem Vergleichen der Nachricht $m$ mit dem Ergebnis

$$v_T(sig) := sig^e \bmod n \stackrel{?}{=} m.$$

2.5.3. *Attacken auf den RSA-Algorithmus.* Aufgrund seiner enormen Bedeutung für die moderne Kryptographie sollen an dieser Stelle noch einige Attacken auf den RSA-Algorithmus erklärt werden. Genaugenommen sind die zahlreichen Angriffe auf den RSA-Algorithmus immer nur auf Spezialfälle anwendbar, i.a. ist der RSA-Algorithmus ungebrochen.
**Die Homomorphie-Eigenschaft von RSA:** Für den öffentlichen Schlüssel $(e, n)$ eines RSA-Systems gilt:

$$m_1{}^e m_2{}^e \bmod n = (m_1 m_2)^e \bmod n.$$

Ein Angreifer kann ohne Kenntnis von $m_1$ und $m_2$ die Nachricht $m_1 m_2$ verschlüsseln. Die Problematik beim Signaturverfahren ist analog.
Einem Angriff kann durch Einsatz eines Fingerabdrucks entgegengewirkt werden. Dabei wird auf die Nachrichten $m_1$ und $m_2$ eine gehashte Funktion angewandt; bei der Multiplikation von Seiten des Angreifers geht die Eigenschaft des Fingerabdrucks verloren, wodurch der Angriff bemerkbar wird. Ergeben die Nachrichten $m_1$ und $m_2$ einen Sinn, dann aber nicht $m_1 m_2$.
**Generieren eines gültigen Nachricht-Signatur-Paares:** Man kann mit dem RSA-Verfahren leicht aus einer Signatur eine Nachricht erzeugen, die diese Signatur besitzt. Dazu wählt man *sig*aus und bildet dazu $m := sig^e \bmod n$.
Bei der Überprüfung der Signatur $m^d = sig^{e\,d} = sig \bmod n$ wird ein korrektes Ergebnis geliefert.
**Verwendung kleiner Exponenten:** Aus Effizienzgründen wird die Verwendung von $e = 3$ als Exponenten des öffentlichen Schlüssels für verschiedene Moduli vorgeschlagen. Treten in einem solchen System drei Teilnehmer auf, die dieselbe Nachricht verschlüsseln wollen, ist aus den Kryptogrammen der Klartext einfach zu berechnen:
Seien

$$y_1 = m^3 \bmod n_1,$$
$$y_2 = m^3 \bmod n_2,$$
$$y_3 = m^3 \bmod n_3$$

die Chiffretexte der einzelnen Teilnehmer. Nach dem chinesischen Restsatz gibt es die eindeutige Lösung

$$y = m^3 \bmod n_1 n_2 n_3$$

Wegen der Voraussetzung, daß $m$ kleiner als die einzelnen Moduli sein muß, ist demnach $m^3$ kleiner als $n_1 n_2 n_3$. Dies hat aber zur Folge, daß

$$y = m^3 \bmod n_1 n_2 n_3 = m^3$$

ist, d.h. durch das Ziehen der dritten Wurzel in $\mathbb{Z}$ kann $m$ ganz einfach aus $y$ berechnet werden.
**Gleicher Modul für mehrere Benutzer:** Besitzen innerhalb eines Systems zwei Teilnehmer $A$ und $B$ den gleichen RSA-Modul $n = pq$, so ist jede an $B$ gerichtete Nachricht für $A$ lesbar und umgekehrt.
Dem Teilnehmer $B$ sind bekannt:$e_B, d_B, e_A$; ($e_A$ ist nämlich der öffentliche Schlüssel von $A$)
B berechnet:

$$h = \frac{e_B d_B - 1}{ggT(e_A, e_B d_B - 1)} = \frac{kgV(e_A, e_B d_B - 1)}{e_A}$$

wobei $h$ ein Vielfaches von $\varphi(n) = (p-1)(q-1)$ und zu $e_A$ teilerfremd ist. Dadurch erhält $B$ unter Verwendung des erweiterten Euklidischen Algorithmus, die Vielfachsummendarstellung

$$c \cdot h + d \cdot e_A = 1.$$

Somit hat $B$ ein $d$ gefunden, für das die Eigenschaft $d \cdot e_A \bmod \varphi(n) = 1$ erfüllt ist. Durch Potenzierung mit $d$ werden alle Nachrichten von $A$ für $B$ lesbar. Dem Teilnehmer $A$ sind dieselben Möglichkeiten gegeben.
**Verschlüsselung von Nachrichten, die in algebraischer Beziehung zueinander stehen:** Diese Attacke gehört zu den aktuelleren Angriffen auf den RSA-Algorithmus. Es wird verdeutlicht, wie einfach ein Rückschluß auf $m$ ist, wenn die Chiffretexte von $m$ und $m+1$ bekannt sind und zudem der Exponent $e = 3$ benutzt wird.
Sei also $c_1 = m^3 \bmod n$ und $c_2 = (m+1)^3 \bmod n$. Dann gilt

$$\frac{c_2 + 2c_1 - 1}{c_2 - c_1 + 2} = \frac{(m+1)^3 + 2 - 1}{(m+1)^3 - m^3 + 2} = \frac{3m^3 + 2m^3 + 3m}{3m^2 + 3m + 3} = m$$

## 3. Grundlegende Protokolle

In diesem Kapitel werden einige grundlegende Protokolle vorgestellt, zu deren sinnvollen Nutzung mindestens die beiden folgenden Bedingungen erfüllt sein müssen:

- **Durchführbarkeit:** Das Protokoll liefert bei korrektem Verhalten aller im System vorhandenen Teilnehmer mit beliebig hoher Wahrscheinlichkeit das gewünschte Ergebnis
- **Korrektheit:** Bei einem Betrugsversuch von Seiten eines Teilnehmers kann dieser mit beliebig hoher Wahrscheinlichkeit erkannt werden

**3.1. Authentifikationsverfahren.** Bei den Authentifikationsverfahren lassen sich zwei verschiedene Mechanismen unterscheiden. Zum einen die *unidirektionalen*, bei denen die für die Authentifikation relevanten Informationen nur in einer Richtung übermittelt werden und die *bidirektionalen*, bei denen Nachrichten in beide Richtungen fließen.

3.1.1. *unidirektional.* Zu den bekanntesten unidirekionalen Authentifizierungsverfahren gehören die **Paßwortverfahren**. Solche Verfahren werden in der Regel in Situationen eingesetzt, in denen sich Personen gegenüber einer zentralen Stelle authentifizieren müssen. Jeder Teilnehmer ist in Besitz eines individuellen Geheimnisses (seines **Paßwortes** oder seiner **PIN**, Persönliche Identifizierungs Nummer), das der zentralen Stelle bekannt ist. Während des Authentifikationsprozesses übermittelt der einzelne Teilnehmer der Zentrale mit seiner Identität auch sein Geheimnis. Die Zentrale überprüft die Richtigkeit der Identität zu dem übermittelten Geheimnis.
Eine grundlegende Schwäche von Paßwortverfahren besteht darin, daß nicht nur das Geheimnis statisch ist, sondern daß auch die zur Authentikation eines Benutzers gesendeten Nachrichten immer gleich sind.
Alle weiteren vorgestellten Verfahren zur Authentikation zweier Instanzen haben das Ziel, bei festem Geheimnis die übermittelten Nachrichten variabel und möglichst unvorhersagbar zu gestalten.
Einfache Verfahren, die dies leisten, sind die **Wechselcodeverfahren**: Der Teilnehmer berechnet einen Authentikationscode, der nach einem vorher vereinbarten Verfahren aus seinem Geheimnis und einem anderen veränderlichen Wert erstellt wird. Dieser wird an die Zentrale gesendet, die die Berechnung des Teilnehmers wiederholen und die beiden Ergebnisse miteinander vergleichen kann.
Ein anderes Beispiel, das mit einer Verschlüsselungsfunktion $f$ realisiert wird: Will sich ein Partei $A$ gegenüber $B$ authentisieren, einigt man sich vorher auf eine natürliche Zahl, den *Anfangszählerstand* $z$ und auf einen gemeinsamen geheimen Schlüssel $k$.

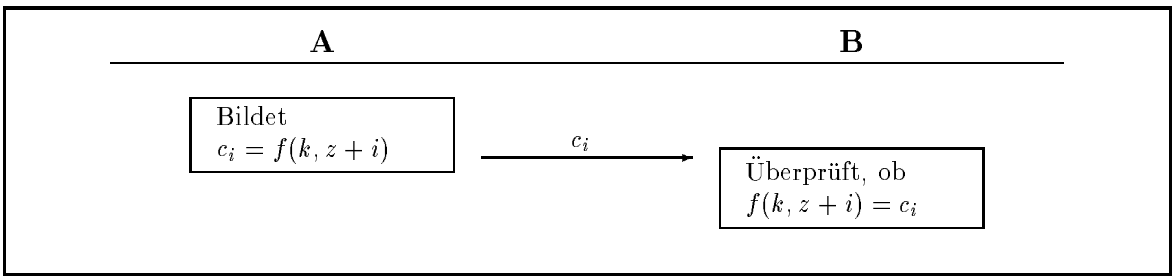



ABBILDUNG 4. Wechselcodeverfahren

3.1.2. *bidirektional.* Das im vorigen Abschnitt erwähnte Wechselcodeverfahren hat den Nachteil, daß man Authentifikationsnachrichten „vorproduzieren"kann. Beim Verfahren Challenge-and-Response ist ein solcher Angriff nicht ausführbar, da die Nachrichten, die $B$ erwartet, von $A$ jeweils „$frisch$" produziert sein müssen.
Man kann sich dieses Verfahren anhand eines Telefonats verdeutlichen. Einer stellt eine Frage, die nur die Person am anderen Ende der Leitung beantwortet kann. In technischen Anwendungen geht man folgendermaßen vor: Beide Seiten einigen sich auf eine Einwegfunktion $f_k$, die von einem Schlüssel $k$ abhängt. Dabei entspricht der unvorhersagbaren Frage, die $B$ stellt, eine Zufallszahl $RAND$(die „challenge").
Statt eines symmetrischen Algorithmus $f$ kann man auch ein Signaturverfahren verwenden, was den Vorteil aufweist, daß $B$ das Geheimnis von $A$ nicht zu kennen braucht.

**3.2. Protokolle.** In der klassischen Kryptographie konnten die Mechanismen und Algorithmen erst nach dem Austausch eines geheimen Schlüssels greifen. Geheimnisse wurden kryptographisch ungesichert ausgetauscht; die Sicherheit des Systems belief sich ausschließlich auf einen **geheimen Kanal**, der für den Austausch der Geheimnisse benötigt wurde. Dieser kann physikalisch (technisch abhörsichere Übertragung) oder organisatorisch (Übertragung durch einen vertrauenswürdigen Boten) realisiert werden.
Die moderne Kryptographie distanziert sich von diesem Konzept; man benötigt keine geheimen Kanäle mehr: Geheime Schlüssel können über nicht-geheime, also öffentliche Kanäle vereinbart werden.

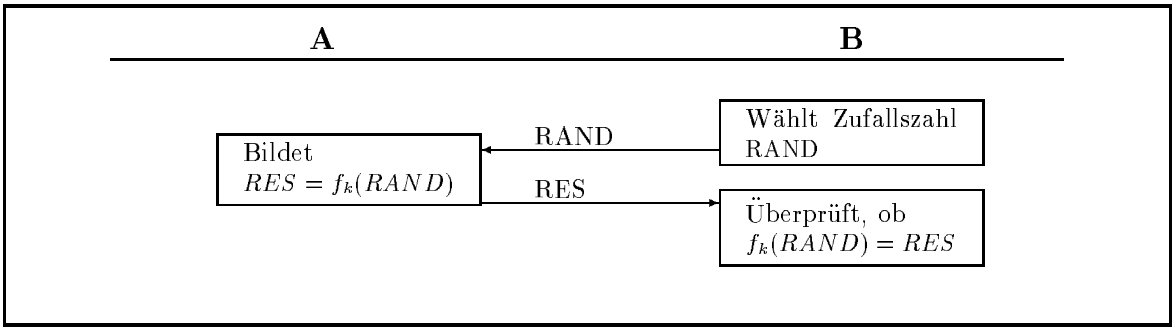



ABBILDUNG 5. Challenge-and-Response mit schlüsselabhängiger Einwegfunktion

Zunächst sollte man das Protokoll von Diffie und Hellman zur Schlüsselvereinbarung erwähnen und danach weitere Protokolle, z.B. welche digitale Signaturen nutzen.

3.2.1. *Diffie-Hellman-Schlüsselvereinbarung.* Die Diffie-Hellman-Schlüsselvereinbarung ist ein Protokoll, welches

- offene Unterhaltung zuläßt
- jedem Teilnehmer das gleiche Geheimnis liefert
- einen Lauschangriff abwehrt

Dem Protokoll liegt die *diskrete Exponentialfunktion*

$$\boxed{a \mapsto \alpha := g^a \bmod p} \qquad (a \in \mathbb{N})$$

zugrunde, die neben ihrer Einwegeigenschaft noch die Kommutativität der Exponenten mitbringt, die für die Durchführbarkeit des Protokolls wichtig ist.
Wollen zwei Personen $A$ und $B$ miteinander kommunizieren, einigen sie sich zuerst auf ein $e$ Primzahl $p$ und eine natürliche Zahl $g$. Die Zahlen können öffentlich sein. Zunächst wählt sich jeder eine beliebige Zufallszahl, die den privaten Schlüssel darstellt und jeder potenziert $g$ mit seiner Zahl modulo p. Die erhaltenen Ergebnisse werden ausgetauscht, und wiederum mit der eigenen Zufallszahl potenziert.

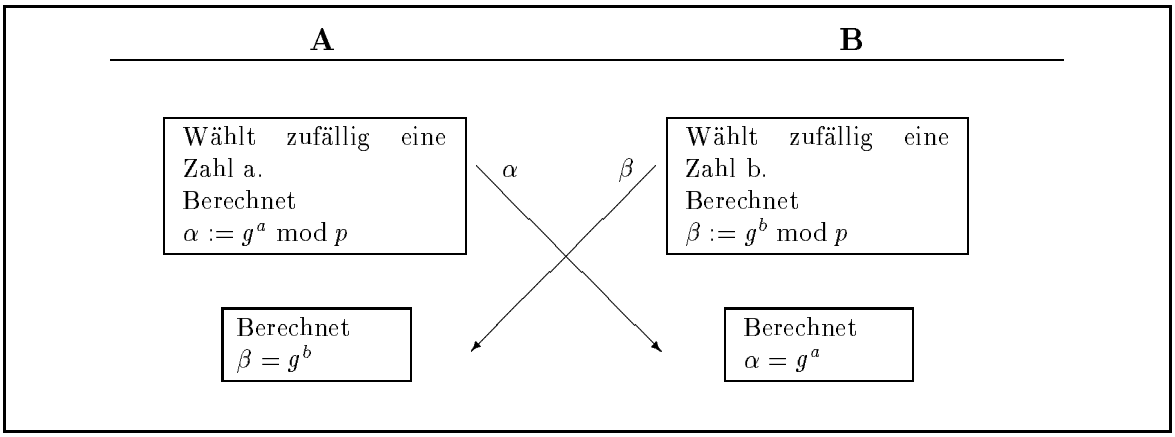



ABBILDUNG 6. Diffie-Hellman-Schlüsselvereinbarung

$A$ und $B$ erhalten den gemeinsamen Wert $k = \beta^a \bmod p = \alpha^b \bmod p$ Wegen der Kommutativitätseigenschaft, gilt

$$\beta^a \bmod p = (g^b)^a \bmod p = g^{ba} \bmod p = g^{ab} \bmod p = (g^a)^b \bmod p = \alpha^b \bmod p$$

Ein Angreifer, dem die beiden Zahlen $p$ und $g$ bekannt sind, könnte sich durch Abhören der Unterhaltung auch $\beta$ und $\alpha$ verschaffen, allerdings kann er dadurch nicht auf $k$ schließen. Ein möglicher Angriff besteht darin, den Wert $k$ auf dieselbe Weise wie $A$, das heißt also $k = \beta^a \bmod p$ zu berechnen. Hierfür müßte dem Angreifer $a$ bekannt sein, also müßte er den diskreten Logarithmus von $\alpha$ zur Basis $g$ berechnen können.

3.2.2. *Das ElGamal-Verschlüsselungsverfahren.* Der folgende asymmetrische Verschlüsselungsalgorithmus, der auf Taher ElGamal zurückgeht, entsteht durch Variation des Diffie-Hellman Schlüsselvereinbarungsprotokolls. Das „Senden der Teilschlüssel"wird hier zeitlich entkoppelt.

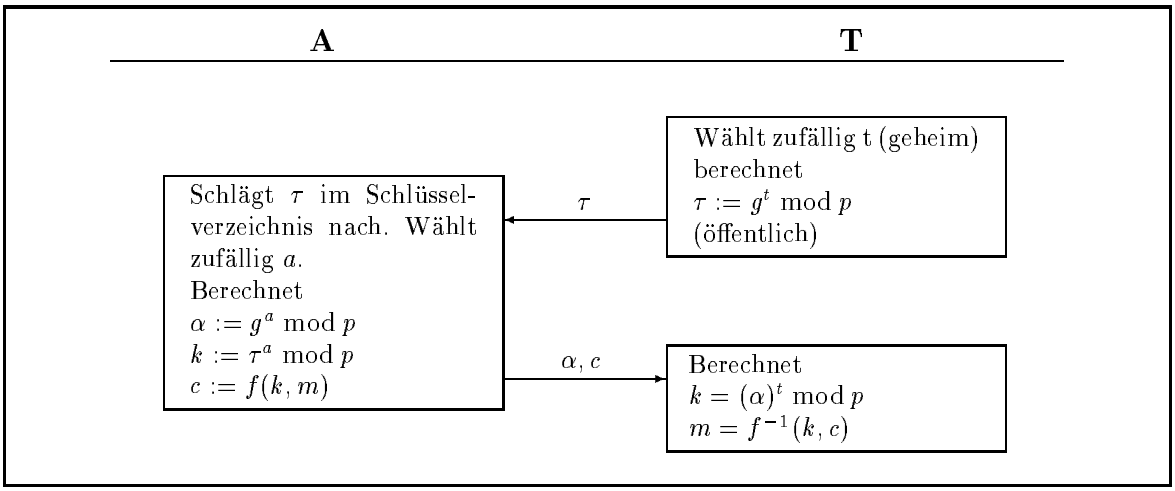



ABBILDUNG 7. Das ElGamal-Verschlüsselungsverfahren

Auch hier einigt man sich zuvor auf eine Primzahl $p$ und eine natürliche Zahl $g$, die öffentlich sind. Jeder Teilnehmer $T$ wählt sich eine geheime Zahl $t$ als privaten Schlüssel, potenziert $g$ mit diesem modulo $p$ und veröffentlicht das Ergebnis als seinen öffentlichen Schlüssel. Um an den Teilnehmer $T$ eine verschlüsselte Nachricht zu schicken, benötigt der Sender dessen öffentlichen Schlüssel und berechnet die beiden Werte $\alpha$ und $k$. Dann verschlüsselt er die Nachricht $m$ unter den Schlüssel $k$ mit Hilfe eines beiden bekannten symmetrischen Algorithmus $f$ und sendet den Geheimtext $c = f(k, m)$ zusammen mit $\alpha$ an $T$. Der Empfänger wendet seinen privaten Schlüssel $t$ auf $\alpha$ an, erhält dadurch $k$, wodurch er dann die Nachricht berechnen kann (vgl. Abbildung 7).

3.2.3. *Das ElGamal-Signaturverfahren.* Die im ersten Kapitel erwähnte Modellierung einer digitalen Signatur findet in dem folgenden Verfahren Verwendung. Zur Erzeugung und Verifikation der Signatur wird erneut der gleiche private Schlüssel $t$ und der gleiche öffentliche Schlüssel $\tau = g^t \bmod p$ wie beim ElGamal-Verschlüsselungsverfahren genutzt.
Folgendermaßen geht Teilnehmer $T$ vor, wenn er eine Nachricht $m$ signieren möchte: Zunächst wählt er eine Zahl $r$ mit der Eigenschaft $ggT(r, \varphi(p)) = 1$ und bildet $k = g^r \bmod p$. Danach berechnet er eine Lösung $s$ der Kongruenz $t \cdot k + r \cdot s \equiv m \bmod (\varphi(p))$. Die beiden Zahlen $k$ und $s$ entsprechen zusammen der Signaturfunktion. Der Empfänger bildet zur Verifikation der Unterschrift $g^m \bmod p$ und $\tau^k \cdot k^s \bmod p$ und vergleicht, ob diese Zahlen identisch sind. Die Durchführbarkeit des Verfahrens wird durch den Satz von Fermat garantiert:

$$\boxed{g^m \equiv g^{t \cdot k + r \cdot s} \equiv \tau^k \cdot k^s (\mathrm{mod} p)}$$

Dieses Verfahren ist auch korrekt, denn bislang ist noch kein effizienter Angriff gefunden.

3.2.4. *Shamirs No-Key-Protokoll.* Bei den bisher beschriebenen Protokollen mußte ein Teilnehmer stets den öffentlichen Schlüssel seines Gegenübers kennen bzw. vorher einen gemeinsamen Schlüssel vereinbaren. Nun stellt sich natürlich die Frage, ob ein Nachrichtenaustausch auch ohne jegliche Schlüsselkenntnis stattfinden kann. Die überraschende Idee zur Lösung dieses Problems geht auf eine unveröffentlichte Arbeit von A.Shamir zurück.

Möchte eine Person $A$ an $B$ eine geheime Nachricht $m$ schicken, so steckt sie $m$ in eine Kiste und verschließt diese mit einem Vorhängeschloß, zu dem nur sie einen Schlüssel besitzt. Sie schickt die Kiste an $B$, der seinerseits sein Vorhängeschloß anbringt. $B$ schickt die Kiste an $A$ zurück, die ihr Schloß öffnet und die Kiste wieder an $B$ schickt. Dieser kann nach Entfernen des letzten Schlüssels die Kiste öffnen und die Nachricht lesen.
Mathematische Grundlage ist die diskrete Exponentialfunktion, die als *symmetrische Verschlüsselungsfunktion* eingesetzt wird. Dafür einigen sich beide Teilnehmer auf eine große Primzahl $p$. $A$ erzeugt ein Paar von Zahlen $(a, a')$; entsprechend erzeugt $B$ ein Paar $(b, b')$, wobei $a'$ bzw. $b'$ die multiplikativen Inversen von $a$ bzw. $b$ *modulop* $-1$ sind. Das Potenzieren mit $a$ entspricht dem Verschließen und $a'$ dem Entfernen des Schlosses.
Das Protokoll läuft wie in Abbildung 8 beschrieben ab.

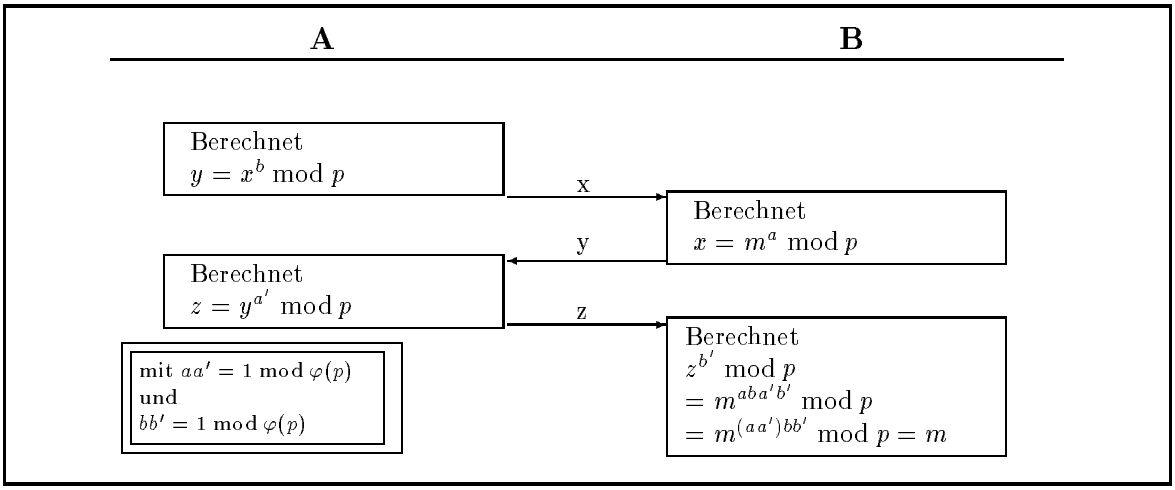



ABBILDUNG 8. Das No-Key-Protokoll mit der diskreten Exponentialfunktion

3.2.5. *Needham-Schroeder.* Das folgende Protokoll kombiniert Authentifikation und Schlüsselaustausch, um ein allgemeines Problem zu lösen: Wie können sich zwei Personen, die einen geheimen Schlüssel austauschen wollen, sicher sein, daß sie auch wirklich miteinander und nicht mit einem Angreifer kommunizieren? In folgendem Protokoll wird eine dritte Instanz eingeführt – **Key Distribution Center (KDC)** – ihr obliegt die Schlüsseladministration [**Sch96**]. In unserem Beispiel sei dies Trent. Das Verfahren basiert auf der absoluten Sicherheit der Administration.

| Symbole | Bedeutung der verwendeten Symbole |
|---|---|
| A | Name von Alice |
| B | Name von Bob |
| $E_A$ | Verschlüsselung durch einen Schlüssel (nur Trent und Alice bekannt) |
| $E_B$ | Verschlüsselung durch einen Schlüssel (nur Trent und Bob bekannt) |
| K | zufällig gewählter Sitzungsschlüssel |
| $R_A, R_B$ | zufällig gewählte Zahlen von Alice bzw. Bob |

Der Verlauf des Protokolls:

(1) Alice schickt Trent $A, B, R_A$.
(2) Trent generiert einen Sitzungsschlüssel $K$, verschlüsselt diesen und den Namen von Alice mit $E_B$, dann verschlüsselt er die Teilnachricht, Bobs Namen, $R_A$ und den Schlüssel mit $E_A$ und schickt dies an Alice.
(3) Alice extrahiert den Schlüssel $K$, versichert sich, daß $R_A$ mit jenem aus dem ersten Schritt übereinstimmt und sendet Bob die Teilnachricht.
(4) Bob extrahiert nach der Entschlüsselung $K$, generiert ein $R_B$, verschlüsselt diesen Wert mit $K$ und sendet es an Alice.

(5) Alice entschlüsselt die Nachricht und berechnet $R_B - 1$, verschlüsselt den neuen Wert mit $K$ und sendet dies an Bob.
(6) Bob entschlüsselt mit $K$ und kann $R_B - 1$ verifizieren.

Die letzten drei Schritte des Verfahrens sind ein Challenge-Response-Protokoll, was zusätzliche Sicherheit unter Alice und Bob garantiert.

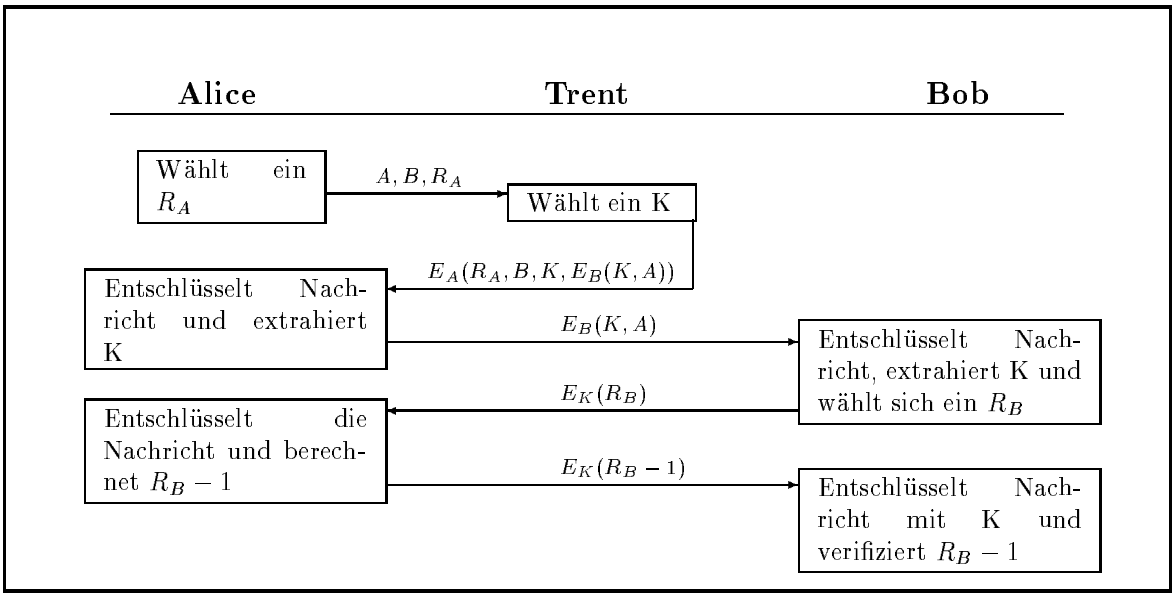



ABBILDUNG 9. Das Needham-Schroeder Protokoll

Ein möglicher Angriff findet statt, wenn ein Angreifer, Mallory, sich Zugang zur Administration verschafft und $E_B$ kennt. Er nimmt einen willkürlichen Schlüssel K', Alices Namen und verschlüsselt diese mit $E_B$. Die Nachricht schickt er dann Bob, der K' extrahiert, sich ein $R_B$ wählt und diesen, mit K' verschlüsselt, an Alice sendet. Mallory kann die Nachricht allerdings abfangen und durch ein Challenge-Response-Verfahren Bob davon überzeugen, daß dieser mit Alice kommuniziert.

3.2.6. *Blinde Signatur.* Normalerweise will der Unterzeichner eines Dokuments wissen, was er unterschreibt. Das ist auch bei der elektronischen Signatur der Fall. Allerdings gibt es auch Anwendungen, in denen gefordert wird, daß der Unterschreibende nicht wissen *darf*, was er unterschreibt (z.B. elektronische Münzen oder elektronische Wahlen).

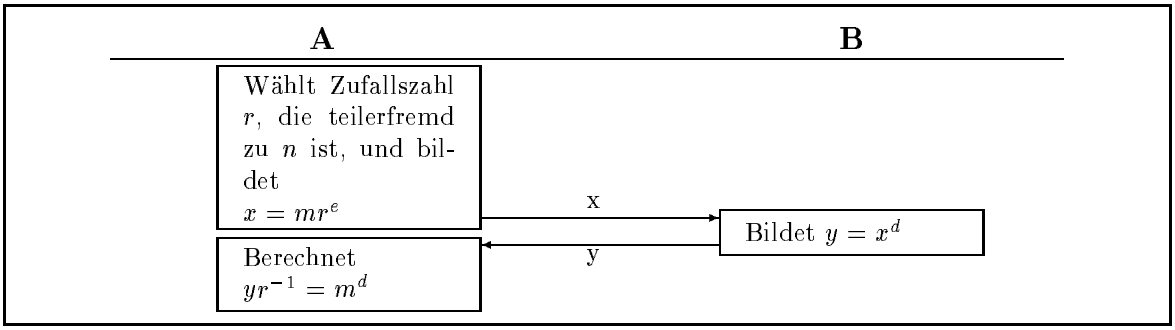



ABBILDUNG 10. Protokoll zur blinden Signatur eines Dokuments m mit dem RSA-Schemas

Verfahren zur Erzeugung blinder Signaturen:
Eine Person $A$ kann einem Unterzeichner $B$ ein Dokument so vorlegen, daß

- B die Nachricht nicht sieht
- A die gültige Unterschrift von B unter dem Dokument erhält.

Die kryptographische Realisierung erfolgt durch Anwendung des RSA-Algorithmus.
Wie aus dem Abbildung 10 hervorgeht, wird die eigentliche Nachricht $m$ mit $r^e$ multipliziert, wobei $e$ der öffentliche Schlüssel von $B$ ist. $B$ ist es unmöglich, den Inhalt zu lesen, denn er kennt $r$ nicht. Nach Erhalt der Nachricht, unterschreibt er durch Potenzierung mit $d$. Das Ergebnis wird an $A$ geschickt. $A$ multipliziert den erhaltenen Wert $y$ mit dem Inversen $r^{-1}$ von $r$.
Dadurch erhält $A$ wegen

$$y \cdot r^{-1} = x^d \cdot r^{-1} = (m \cdot r^e)^d \cdot r^{-1} = m^d \cdot r^{ed} \cdot r^{-1} = m^d \cdot r \cdot r^{-1} = m^d \bmod n$$

die unterschriebene Nachricht.

3.2.7. *Knobeln übers Telefon.* Die Problematik, die sich beim Knobeln übers Telefon ergibt, ist folgende: Derjenige, der als letzter antworten darf, kann seine Meinung noch ändern. Demnach wird eine Methode benötigt, die für beide eine verbindliche und unveränderliche Entscheidung gleich zu Beginn festlegt. Zur Lösung dieses Problems braucht man Commitment-, genauer gesagt Bit-Commitment-Techniken.
*Beispiel:* Alice und Bob wollen per „Papier-Schere-Stein"übers Telefon knobeln. Dabei muß sich einer der beiden auf ein Element der 3-elementigen Menge {Papier, Stein, Schere} festlegen.
Mathematisch kann man dieses Problem mit Hilfe der Isomorphie von Graphen lösen:

(1) Beide legen gemeinsam drei nichtisomorphe Graphen $G_{Papier}, G_{Stein}, G_{Schere}$
(2) Alice wählt einen der Graphen und wendet eine zufällige Permutation auf ihn an und schickt Bob das Ergebnis
(3) Bob, der aus der Permutation heraus nicht den ursprünglichen erkennen kann, wählt nun selber einen Graphen und teilt Alice seine Wahl mit
(4) Jetzt erst teilt Alice Bob ihren Graphen mitsamt der von ihr angewandten Permutation

Alice kann ihre Wahl nicht mehr rückgängig machen, denn der übermittelte Graph ist nur zu einem der drei Graphen isomorph. Bob hingegen kann im zweiten Schritt nicht erkennen, welchen der Graphen Alice gewählt hat, da das Problem zu erkennen, ob zwei Graphen isomorph oder nicht isomorph sind, für große Graphen praktisch unlösbar [**BSW98**].

## 4. Zusammenfassung

Die Kryptographie läßt sich in symmetrische und asymmetrische Verschlüsselungsverfahren (Public-Key-Verfahren) unterteilen, wobei letztere vorteilhafter sind und denen zudem ein größeres Anwendungsgebiet obliegt. Hierbei sind der RSA-Algorithmus und das ElGamal-Verschlüsselungsverfahren zu erwähnen. Neben dem wohl bekanntesten Authentifikationverfahren – dem Paßwortverfahren – existiert das Wechselcodeverfahren. Das Challenge-and-Response Verfahren gehört zu den bidirektionalen und kann einem Angriff durch Vorproduktion von Authentifikationsnachrichten standhalten. Bei den meisten Protokollen wird die Kenntnis des öffentlichen bzw. die Erzeugung eines gemeinsamen Schlüssels zu ihrer effizienten Nutzung vorausgesetzt. Ganz anders verhält es sich bei Shamirs No-Key-Protokoll. Unter Verwendung der diskreten Exponentialfunktion kommt er ohne jegliche Schlüsselkenntnisse aus. Erwähnenswert ist auch die Möglichkeit zur Erzeugung blinder Signaturen mit Hilfe des RSA-Schemas und das Ausnutzen der Graphenisomorphie für das Knobeln über das Telefon.

KAPITEL 6

# Zero-Knowledge Verfahren

**Christian Mathis**

## 1. Einleitung

Betrachten wir folgenden Dialog zwischen Alice und Bob:

*Alice:* „Ich kenne die private Telefonnummer unseres Bundespräsidenten. “
*Bob:* „Kann nicht sein. “
*Alice:* „Doch.“
*Bob:* „Dann beweise es mir. “
*Alice:* „Ok, ich sage sie Dir. “Sie flüstert in sein Ohr.
*Bob:* „Wirklich? Ich werde es direkt ausprobieren. “

...und bis am selben Abend erhält der Bundespräsident tausende Anrufe... Unglücklicherweise besteht der gewöhnlichste Weg, eine Person vom Besitz eines Geheimnisses zu überzeugen darin, es ihm zu verraten. Natürlich möchte man dies in den allerwenigsten Fällen. Die moderne Kryptographie kann hier Abhilfe schaffen, indem sie ein Verfahren bereitstellt, das obiges Problem behebt: Durch so genannte Zero-Knowledge Protokolle wird Alice ermöglicht, Bob vom Besitz ihres Geheimnisses zu überzeugen, *ohne* dieses zu verraten.
Kapitel 2 widmet sich den Grundlagen der ZK-Protolle, es wird ein historischer Zusammenhang hergestellt und der Begriff „Zero-Knowledge “definiert. In Kapitel 3 analysieren wir eine der wichtigsten mathematischen Implementierungen eines ZK-Protokolles. Danach beschäftigen wir uns in Kapitel 4 mit zwei Weiterentwicklungen, den *parallelen* und den *nicht-interaktiven* Verfahren. Kapitel 5 zeigt auf, wie ZK-Protokolle hintergangen werden können.

## 2. Grundlagen

**2.1. Interaktive Beweise.** Jeder ist wohl schon einmal in die Situation gekommen, einen mathematischen Satz beweisen zu müssen (oder auch zu wollen). Wie geht man dabei vor? Man bedient sich einer Menge von Axiomen (unbeweisbare Grundaussagen) und Sätzen (schon bewiesene mathematische Aussagen) und wendet solange logische Schlussfolgerungen an, etwa Inferenzregeln eines Kalküls, bis man schließlich das gewünschte Resultat erzielt. Schreibt man nun diese Schlussfolgerungen in der „richtigen Reihenfolge “auf, so kann sich jeder von der Richtigkeit des Satzes überzeugen.

Die Beweisführung verlief allerdings nicht immer in diesem Stil, denn erst vor etwa 100 Jahren wurde die Logik, einschließlich des Beweisbegriffes, so gefasst, wie wir sie heute kennen. In der Antike schrieb man Beweise oft als *Dialoge* auf, wobei eine Person die Rolle des Beweisführers übernahm, eine andere Person die Rolle des Beweis-Verifikators. Solche Beweisdialoge können als Spiel verstanden werden:

- Eine Person $P$ (Prover, Beweisführer, im folgenden immer als „Peggy “bezeichnet) möchte eine Aussage $A$ beweisen.
- Eine Person $V$ (Verifier, Beweisverifikator, im folgenden „Victor “) kontrolliert Peggys Beweis.
- Das Spiel besteht aus mehreren Runden, wobei in jeder Runde Victor ein Argument liefert, worauf Peggy ein Gegenargument finden muss. Eine Runde wird entweder von Peggy oder von Victor gewonnen.

- Kann Peggy *jede* Runde für sich entscheiden, so ist Aussage $A$ bewiesen. Genauer: Kann man zeigen, dass Peggy eine „Gewinnstrategie "hat, mit der sie jeder der Runden für sich entscheiden kann, ist die Aussage bewiesen.

Machen wir uns dies an einem Beispiel aus der Analysis klar:
Wir möchten beweisen, dass der Grenzwert der Folge $\left(\frac{1}{n^2}\right)_{n\in\mathbb{N}}$ gleich Null ist. Nach Definition ist dies äquivalent zu:

$$\forall_{\varepsilon>0}\exists_{D\in\mathbb{N}}\forall_{n\geq D} : \left|\frac{1}{n^2}\right| < \varepsilon$$

Spielverlauf:

- Victor übernimmt die Rolle der Allquantoren.
- Peggy übernimmt die Rolle des Existenzquantors.
- Pro Runde gibt Victor $\varepsilon$ vor (z.B. $\varepsilon = 0.01$).
- Peggy muss nun so mit einem $D$ antworten, dass die Bedingung $\forall_{n\geq D} : \left|\frac{1}{n^2}\right| < \varepsilon$ erfüllt ist. Sie wählt einfach $D = \lceil\sqrt{\frac{1}{\varepsilon}}\rceil$ (im Beispiel: $D = 10$).
- Durch obige Wahl von $D$ gewinnt Peggy immer, sie hat also eine „Gewinnstrategie ", die Aussage ist bewiesen.

Wie schon angeklungen, sind Zero-Knowledge Protokolle Spezialfälle interaktiver Beweise. Weiterhin haben wir schon gehört, dass in Zero-Knowledge Protokollen eine Person vom Besitz eines Geheimnisses überzeugt werden soll, ohne dieses zu erfahren. Wie lässt sich nun dieser Aspekt in einen interaktiven Beweis einbetten? Dazu betrachten wir im nächsten Abschnitt die Vorgehensweise zweier Mathematiker aus dem 16. Jahrhundert, die ebenfalls einen interaktiven Beweis darstellt.

**2.2. Tartaglia und die kubische Gleichung.** Nicolò Tartaglia (eigentlich Fontana) war ein italienischer Mathematiker im 16. Jahrhundert. Sein Vater starb als er sechs Jahre alt war. Mit vierzehn besuchte Tartaglia die Schreibschule, die er wegen fehlender Mittel nicht beendigen konnte. Ab diesem Zeitpunkt brachte er sich alles selbst bei [**Mey98**]. Im Alter von 35 Jahren fand Tartaglia als einer der ersten Mathematiker die allgemeine Lösung der kubischen Gleichung $x^3 + ax^2 + bx + c = 0$. [1] Aus Angst mit den groß en Rechenmeistern seiner Zeit in Konflikt zu geraten, hielt er jedoch seine Lösung geheim. Dennoch drang die Nachricht, Tartaglia habe eine allgemeine Lösung der kubischen Gleichung, zum Rechenmeister Antonio Maria Fior durch. Dieser wollte sich überzeugen und schickte Tartaglia 30 Aufgaben, die dieser allesamt löste und zurückschickte. Das Protokoll ist in Bild 1 dargestellt.

Dieses interaktive Beweisverfahren entspricht einem kryptographischen Protokoll: Es ist *durchführbar* und *korrekt*. Durchführbarkeit bedeutet hierbei, dass alle Schritte, die beim Ablauf der Interaktion notwendig sind, schnell ausgeführt werden können (genauer: Es muss einen polynomialzeitbeschränkten Algorithmus geben, der die nötigen Schritte ausführt). Die Korrektheit gewährleistet die Sicherheit eines Protokolls. Ein möglicher Angreifer muss bei einem korrekten Protokoll vor schwer oder gar nicht lösbaren Problemen stehen (z.B. vor Problemen aus der Komplexitätsklasse $NPV$, das sind solche Probleme, für die bisher noch kein polynomial zeitbeschränkter Algorithmus gefunden wurde, um sie zu lösen. Einfach ausgedrückt, es gibt noch keine schnellen Algorithmen um Probleme aus NPV zu lösen, genaueres lässt sich in [**Rei99**] nachlesen).
Durchführbarkeit ist in unserem Beispiel gegeben, da Tartaglia mit seiner allgemeinen Lösungsformel alle ihm vorgelegten Aufgaben schnell lösen kann und Tartaglias Lösung mittels eines einfachen Vergleiches verifizierbar ist, Korrektheit, da ein möglicher Angreifer die Lösung einer kubischen Gleichung höchsten raten könnte. Die Erfolgswahrscheinlichkeit beim Raten hängt allerdings stark von der jeweiligen Gleichung ab.
Wir betrachten dieses Beispiel allerdings noch wegen einer weiteren interessanten Eigenschaft:
Tartaglia konnte seine Behauptung, eine allgemeine Lösung der kubischen Gleichung zu kennen, beweisen, ohne diese Lösung zu verraten. Diese Eigenschaft wird in Zero-Knowledge Protokollen

[1] Es ist unbekannt, wer tatsächlich als erster diese Lösung fand

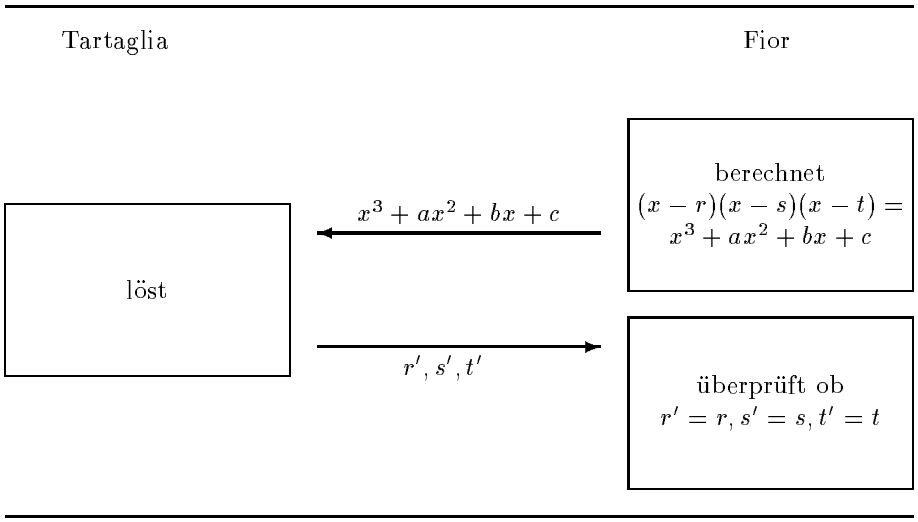



ABBILDUNG 1. Das interaktive Beweisverfahren von Tartaglia

nahezu perfekt realisiert. Im nächsten Abschnitt richten wir unser Augenmerk auf die Definition des Zero-Knowledge Begriffes, anhand dessen man beweisen kann, dass in einem ZK-Protokoll tatsächlich keine geheime Information verraten wird.

**2.3. Die ZK-Eigenschaft.** Warum Tartaglias Protokoll aus dem vorherigen Abschnitt „nur "eine „interessante Eigenschaft "besitzt nicht aber die so genannte Zero-Knowledge Eigenschaft, werden wir später besprechen. Zuerst wird ein Protokoll angegeben, das tatsächlich die ZK-Eigenschaft aufweist.
Peggy kennt den geheimen Zahlencode einer „magischen "Tür und möchte Victor davon überzeugen. Die magische Tür befindet sich in einem Gebäude das schematisch in Bild 2 dargestellt ist. Peggy und Victor vereinbaren nun folgendes Protokoll:

- Peggy
  - geht in Vorraum, schließt die Tür.
  - wählt zufällig „rechts "oder „links", verschwindet hinter gewählter Tür.
- Victor
  - betritt Vorraum.
  - wählt zufällig „rechts "oder „links".
  - fordert von Peggy auf der gewählten Seite zu erscheinen, indem er ruft.
- Peggy kann in jedem Fall Victors Wunsch erfüllen.

*Ist das Protokoll korrekt?*
Sicherlich nicht. Ein Angreifer könnte versuchen, Victors zufällige Wahl vorherzusagen. Die Wahrscheinlichkeit, dass er dabei richtig liegt, ist $\frac{1}{2}$. Vereinbaren Peggy und Victor aber zusätzlich, dass das Protokoll mindestens $t$-Mal durchlaufen werden soll, so sinkt die Vorhersage-Wahrscheinlichkeit des Angreifers auf $p(t) = \left(\frac{1}{2}\right)^t$. Victor kann sich also mit beliebig hoher Wahrscheinlichkeit überzeugen. Wichtig dabei ist, dass Victors und Peggys Entscheidungen („rechts"oder „links") wirklich zufällig erfolgen, andernfalls würde dem Partner die Möglichkeit zum Betrügen gegeben. Das Protokoll ist außerdem durchführbar, da Peggy den Zahlencode kennt.

*Was ist nun die ZK-Eigenschaft und warum wird sie von diesem Protokoll erfüllt?*
Wir nehmen an, Victor habe eine Videokamera in der Hand, mit der er alle Aktionen des Protokolles aus seiner Sicht filmt. Auf dem Band ist später zu sehen und zu hören, wie Peggy im Gebäude verschwindet, wie Victor die vordere Tür selbst öffnet und „rechts"oder „links"ruft, und das $t$-Mal. Ein Außenstehender hat höchstens die Chance diese Aktionen des Protokolles mitzuverfolgen, mehr nicht (er darf ja Peggy nicht begleiten, o.ä.). Das Band repräsentiert also die Sicht

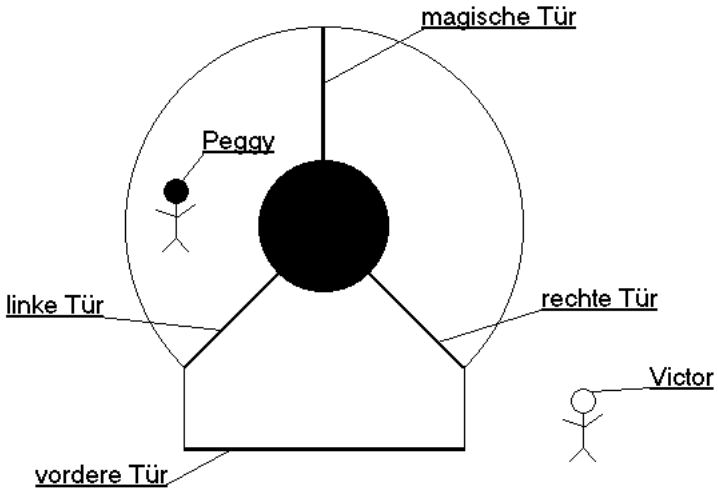



ABBILDUNG 2. Das Gebäude mit der magischen Tür

aller Personen außer der von Peggy. Ein zu Victors Band identisches Videoband lässt sich allerdings auch ohne den geheimen Zahlencode herstellen. Ein Simulator M („Malroy“) rekonstruiert mit Hilfe von einer Person C („Carol“) die Interaktion, dabei übernimmt Malroy die Rolle von Victor und Carol die Rolle von Peggy, wobei Carol den geheimen Zahlencode jedoch nicht kennt. Malroy filmt, wie Carol im Gebäude verschwindet, wie er den Vorraum betritt und „rechts“oder „links“ruft. Steht Carol zufällig auf der Seite, die Malroy fordert, kann sie seinen Wunsch erfüllen. Kann sie es nicht, muss die Szene auf dem Videoband gelöscht werden. Nach durchschnittlich $2t$ Versuchen haben die beiden $t$ „gute“Versuche im Kasten. Kann man durch das Anschauen Malroys Videobandes geheime Information erfahren? - Nein, denn es wurde keine geheime Information verwendet, um das Band herzustellen. Daraus folgt aber unmittelbar, dass man auch aus Victors Videoband keine geheime Information erfahren kann, denn beide Bänder sind gleich! Dies führt uns zur Definition des Zero-Knowledge Begriffes:

DEFINITION 2.1 (**Zero-Knowledge**). Eine Interaktion zwischen Geheimnisträgern P und V besitzt genau dann die *ZK-Eigenschaft*, wenn es einem Simulator M mit Hilfe von V ohne Kenntnis des Geheimnisses möglich ist, seine Sicht der Interaktion so zu rekonstruieren, dass sie von einem Außenstehenden nicht von der Original-Interaktion unterschieden werden kann.

Nun wird auch klar, warum Tartaglias Protokoll aus vorigem Abschnitt nicht die Zero-Knowledge Eigenschaft besitzt: Um Tartaglias Protokoll simulieren zu können, benötigen wir die geheime Information, ohne diese ist das Protokoll nicht simulierbar

## 3. Mathematische Realisierung von ZK-Protokollen

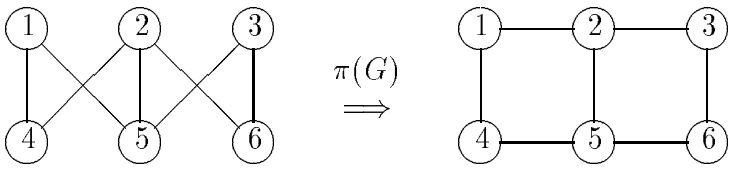
Um ein Zero-Knowledge Protokoll mathematisch zu implementieren, bedient man sich eines Problems $P$, welches idealerweise NP-Vollständig ist. Peggys Geheimnis ist die Lösung dieses Problems (Wie Peggy zu dieser schwer zu findenden Lösung kommen kann, wird am Beispiel gezeigt). Victor wird im Protokoll von Peggy überzeugt, dass sie tatsächlich die Lösung zum Problem $P$ besitzt. Da $P$ NP-Vollständig ist, kann weder Victor noch sonst eine Person hinter Peggys Geheimnis kommen, auch das Protokoll verrät nichts darüber.

### 3.1. Hamiltonische Kreise.

Um das folgende Protokoll, das auf Graphenisomorphie, hamiltonischen Kreisen und Bit-Commitment aufbaut, verstehen zu können, brauchen wir einige Definitionen.

DEFINITION 3.1 (**Isomorphie von Graphen**). Zwei Graphen $G$ und $H$ heißen *isomorph*, falls es eine bijektive Abbildung $\pi$ zwischen den Eckenmengen von $G$ auf $H$ gibt, wobei zwei Ecken in $H$ genau dann miteinander verbunden sind, wenn ihre Urbilder in $G$ miteinander verbunden sind.

Im Bild 3 sind zwei isomorphe Graphen gegeben.



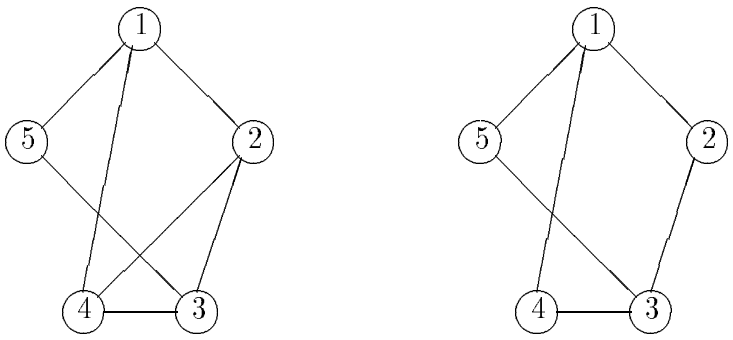

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 5 & 3 & 4 & 2 & 6 \end{pmatrix}$$

ABBILDUNG 3. Zwei isomorphe Graphen $G$ und $H$ mit Permutation $\pi$

Es ist klar, dass die Mächtigkeit der Eckenmengen/Kantenmengen beider isomorpher Graphen gleich ist. Durch eine Permutation der Ecken und Kanten von $G$, wie sie oben angegeben ist, wird in jedem Fall ein zu $G$ isomorpher Graph $H$ erzeugt. Die Komplexität dieser Operation ist, abhängig von der Anzahl der Ecken, polynomial zeitbeschränkt, d.h. sie ist in kurzer Zeit einfach durchzuführen. Möchte man andersherum prüfen, ob zwei gegebene Graphen $G$ und $H$ isomorph zueinander sind, so muss man jede Permutation auf $G$ anzuwenden und das Ergebnis mit $H$ zu vergleichen. Da es aber $n!$ verschiedene Permutationen gibt - wenn $n$ die Anzahl der Ecken im Graphen $G$ darstellt - vermutet man, dass es keinen polynomial zeitbeschränkten Algorithmus zur Lösung dieses Problems gibt, d.h. dass bei großen Graphen dieses Problems praktisch unlösbar ist. Die Graphenisomorphie ist eines der wenigen Probleme für die es bislang weder gelungen ist, die NP-Vollständigkeit zu zeigen, noch zu beweisen, dass es in P liegt.[**Rei99**]

DEFINITION 3.2 (**Hamiltonischer Kreis**). Ein hamiltonischer Kreis $k$ in einem Graphen $G$ ist ein geschlossener Weg entlang der Kanten von $G$, der jede Ecke des Graphen genau einmal passiert. Ein Graph heißt hamiltonisch, wenn er einen hamiltonischen Kreis hat.

Der linke Graph in Bild 4 hat einen hamiltonischen Kreis $k$, der rechte nicht.




$$k = (1, 4, 2, 3, 5, 1)$$

ABBILDUNG 4. Ein hamiltonischer und ein nicht-hamiltonischer Graph

Erzeugt man durch eine Permutation $\pi$ einen zu $G$ isomorphen Graphen $H$, so ist dieser auch hamiltonisch, und es gilt $k_H = \pi(k)$. Möchte man einen hamiltonischen Graphen mit $n$ Ecken erzeugen, so kann man zum Beispiel so vorgehen: Man permutiert die Zahlen $1 \ldots n$ zufällig, ordnet sie der entstandenen Reihenfolge nach als Ecken in einem Kreis an und verbindet je zwei Nachbarn durch eine Kante. Man erhält einen Graphen, der aus einem einzigen hamiltonischen Kreis besteht. Zur „Verwirrung"sollte man nun noch beliebige andere Kanten einzeichnen. Es ist wieder klar, dass dies in polynomialer Zeit geschehen kann. Möchte man umgekehrt einen hamiltonischen Kreis in einem hamiltonischen Graphen finden, so steht man vor einem NP-vollständigen Problem.
Um in Rechnern mit Graphen operieren zu können, müssen diese als Datenstruktur modelliert werden. Eine Möglichkeit bieten so genannte Adjazenzmatrizen. Hier werden die Knotennummern entlang zweier Achsen aufgetragen und notiert, ob zwei Knoten im Graphen miteinander verbunden sind (durch eine 1 dargestellt) oder nicht (durch eine 0 dargestellt). In Bild 5 sehen wir die Adjazenzmatrix des hamiltonischen Graphen aus Bild 4.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | * | 1 | 0 | **1** | **1** |
| 2 | 1 | * | **1** | **1** | 0 |
| 3 | 0 | **1** | * | 1 | **1** |
| 4 | **1** | **1** | 1 | * | 0 |
| 5 | **1** | 0 | **1** | 0 | * |

ABBILDUNG 5. Adjazenzmatrix des hamiltonischen Graphen in Bild 4

Die 1-en, die zu den Kanten des hamiltonischen Kreis gehören, sind markiert. In jeder Zeile und in jeder Spalte stehen genau zwei markierte 1-en, da jede Ecke des hamiltonischen Kreises je eine Vorgängerecke und eine Nachfolgerecke besitzt. Auf den ersten Blick scheint es einfach zu sein, in einer gegebenen Adjazenzmatrix genau zwei 1-en pro Zeile/Spalte zu markieren, sprich einen hamiltonischen Kreis zu finden. Doch stellt sich bei näherer Betrachtung heraus, dass gewisse Abhängigkeiten in der Matrix erfüllt sein müssen: Markiert man zum Beispiel die 1 an der Koordinate $(2, 1)$, so muss man auch die 1 an der Koordinate $(1, 2)$ markieren. Setzt man dieses Spiel weiter fort, erkennt man schnell, wie schwierig es ist, diese Abhängigkeiten, bei gleichzeitiger Markierung von genau zwei 1-en pro Zeile/Spalte, zu erfüllen.
Im folgenden ZK Protokoll kommen Bit-Commitments zum Einsatz. Einfach ausgedrückt stellt Bit-Commitment eine Methode zur Verschlüsselung eines einzelnen Bit (welches im folgenden mit $\diamond$ bezeichnet wird) dar. Um ein $\diamond$ entschlüsseln zu können, benötigt man den Schlüssel $z$. Funktionen, welche Bit-Commitment realisieren, müssen nicht nur sicher sein, d.h. es soll unmöglich sein, aufgrund des $\diamond$ zu entscheiden, ob eine 0 oder 1 verschlüsselt wurde, sondern es muss ebenfalls unmöglich sein, mit einem Schlüssel $z_0$ eine 0 zu entschlüsseln und mit einem Schüssel $z_1$ eine 1.
Kommen wir zur Interaktion:
Victor möchte ein geheimes Dokument an Peggy schicken, doch zuvor will er sichergehen, dass er es auch mit der „echten"Peggy zu tun hat. Dabei benutzen Peggy und Victor ein Authentifizierungssystem, welches folgendermaßen arbeitet: Peggy erzeugt einen großen hamiltonischen Graphen $G$ und hinterlegt ihn öffentlich und unter ihrem Namen bei einer „vertrauenswürdigen Instanz"[2]. Peggys Geheimnis ist nun der hamiltonische Kreis $k$ im Graphen $G$. Im folgenden Protokoll, welches Peggy identifiziert, beweist Peggy Victor, dass sie den hamiltonischen Kreis $k$ im Graphen $G$ kennt (siehe auch Bild 6):

(1) Peggy wählt eine zufällige Permutation $\pi$ und erzeugt einen zu $G$ isomorphen Graphen $H$ [3] und schickt diesen an Victor.

[2]Diese Instanz hat die Aufgabe Peggys „Echtheit"zu prüfen, wie dies geschieht, sei hier nicht weiter ausgeführt
[3]der wie schon gesehen, auch hamiltonisch ist.

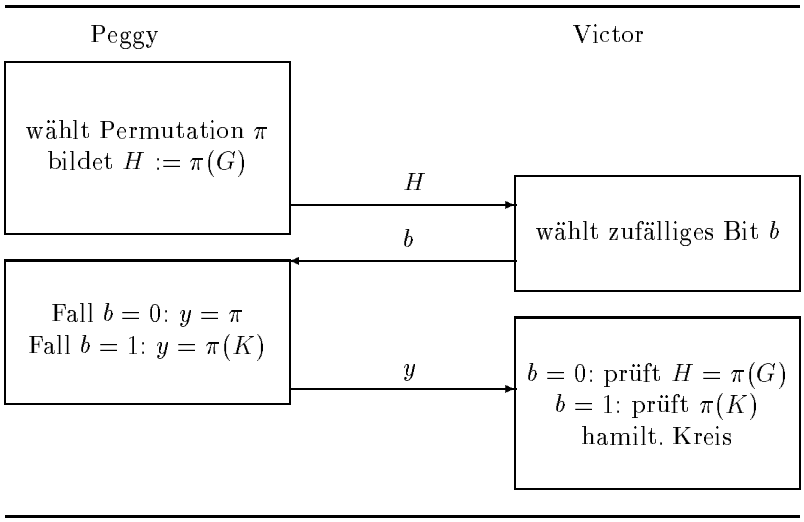



ABBILDUNG 6. ZK Protokoll zum Beweis „$G$ ist hamiltonisch “

(2) Victor antwortet mit einem zufälligen Bit b.
(3)
- Im Fall $b = 0$ soll Peggy die Isomorphie zwischen $G$ und $H$ beweisen. Sie schickt die Permutation $\pi$ an Victor, welcher die Isomorphie verifizieren kann. [4]
- Im Fall $b = 1$ soll Peggy zeigen, dass $H$ hamiltonisch ist. Sie schickt Victor den permutierten hamiltonischen Kreis $k_H = \pi(k)$. Auch in diesem Fall kann Victor verifizieren.[5]

Betrachten wir die technischen Einzelheiten, die das Protokoll praktisch durchführbar machen:
In Schritt 1 bildet Peggy die Adjazenzmatrix $M_1$ des öffentlichen Graphen $G$ und wählt eine zufällige Permutation, mit der sie die Adjazenzmatrix $M_2$ des isomorphen Graphen $H$ konstruiert. $M_2$ wird durch Bit-Commitment verschlüsselt und an Victor geschickt. Victor antwortet in 2. In 3. geschieht eine Fallunterscheidung: Im Fall $b = 0$ schickt Peggy ihm alle Schlüssel um die Adjazenzmatrix $M_2$ zu entschlüsseln und die Permutation $\pi$. Victor kann nun verifizieren. Im Fall $b = 1$ bekommt Victor nur diejenigen Schüssel geschickt, um die „hamiltonischen 1-en“, die zum hamiltonischen Kreis gehören, zu öffnen. Victor kann verifizieren, indem er überprüft, ob sich in jeder Zeile/Spalte genau zwei 1-en befinden. Nach $t$ Wiederholungen ist Victor überzeugt, dass Peggy tatsächlich einen hamiltonischen Kreis im Graphen $G$ kennt, und da sie die einzige ist, die diese Information besitzen kann[6], ist Peggys Identität bewiesen, Victor kann sein geheimes Dokument senden.
Bild 7 zeigt eine Beispielrunde.
*Warum ist das Protokoll durchfürbar?*
Alle elementaren Operationen, etwa das Herstellen des isomorphen Graphen, Verschlüsseln der Adjazenzmatrix, usw. sind in polynomial beschränkter Zeit durchführbar, deshalb ist das gesamte Protokoll durchführbar.
*Warum ist das Protokoll korrekt?*
Eine Angreiferin Carol kann Peggys öffentlichen Graphen einsehen, doch sie hat keine Chance Peggys hamiltonischen Kreis, also Peggys Geheimnis, zu finden. Sie könnte trotzdem versuchen Victor zu hintergehen, indem sie ihre Nachrichten in Schritt 1 präpariert. Dabei kann sie entweder Victor vortäuschen, einen hamiltonischen Kreis zu kennen oder eine Isomorphie zwischen den Graphen $G$ und $H$. Sie kann nicht beide Problemlösungen gleichzeitig vortäuschen, denn um im

[4]dies entspricht in 2.3 dem Fall, dass Peggy auf der richtigen Seite steht und ihr Geheimnis nicht anwendet
[5]hier muss Peggy ihr Geheimnis anwenden
[6]sofern Peggy sie nicht weitergegeben hat

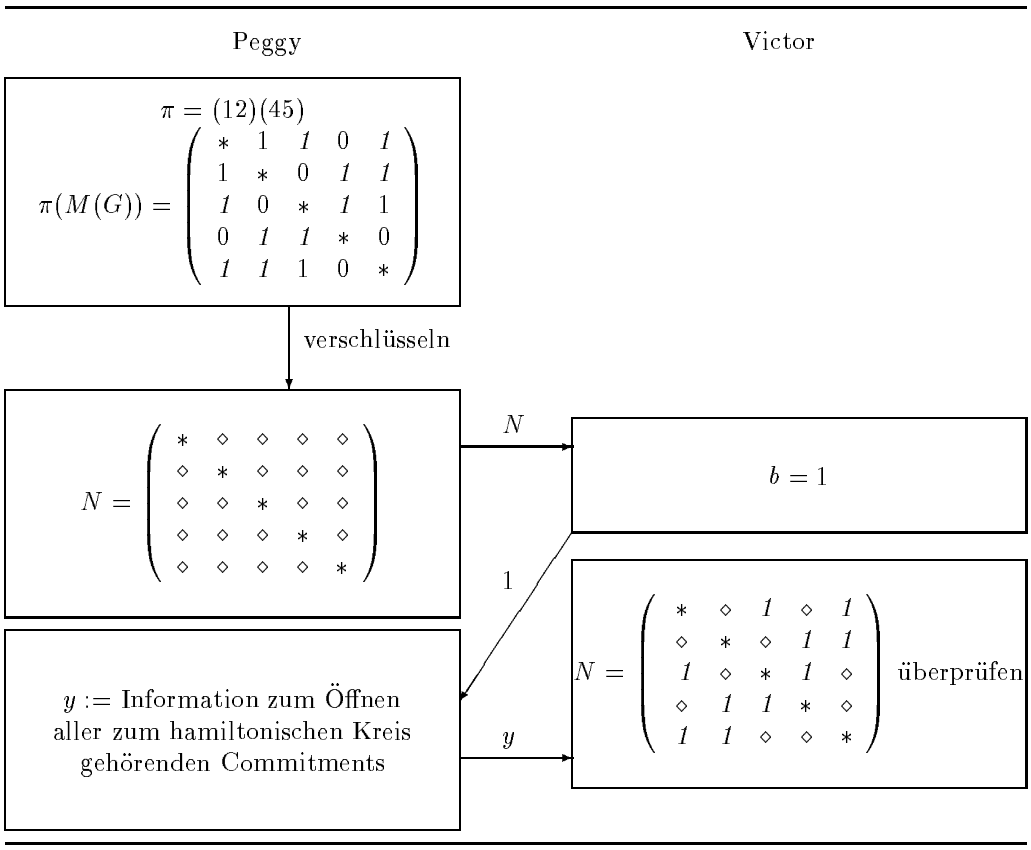



ABBILDUNG 7. Beispielrunde

isomorphen Graphen $H$ einen hamiltonischen Kreis zu finden, benötigt sie den hamiltonischen Kreis in $G$. Carol geht folgendermaßen vor:

- Carol wählt Bit $b'$ und sagt somit Victors Wunsch in 2. voraus.
- – Fall $b' = 0$: Carol wählt eine beliebige Permutation $\pi$ und erzeugt damit den isomorphen Graphen $H$. Trifft ihre Vorhersage zu, so kann sie Victor von der Isomorphie zwischen $G$ und $H$ überzeugen.
  – Fall $b' = 1$: Carol erzeugt einen neuen Graphen $K$, der aus einem einzigen hamiltonischen Kreis auf den Ecken von $G$ besteht. Dieser Graph ist natürlich nicht isomorph zu $G$, das muss er auch nicht sein, denn falls Carols Vorhersage zutrifft, muss sie lediglich die Commitments öffnen, die zum hamiltonischen Kreis gehören. Victor bekommt den Graphen $K$ also nicht als Ganzes zu sehen, kann somit auch nicht entscheiden, ob $K$ isomorph zu $G$ ist
  – Carol schickt den neu entstandenen Graphen an Victor
  – der Rest des Protokolls verläuft identisch zum Originalprotokoll

Nur falls Carols Vorhersage zutrifft, dies geschieht in der Hälfte aller Fälle, kann sie Victor betrügen, andernfalls nicht. Die Betrugswahrscheinlichkeit ist also $\frac{1}{2}$. Das Protokoll ist deswegen nach ausreichender Wiederholung korrekt.

*Warum hat das Protokoll die Zero-Knowledge Eigenschaft?*
Nun müssen wir wieder prüfen, ob die Interaktion simulierbar ist. Hierbei tritt wieder unser Simulator Malroy auf. Im wesentlichen tätigt er dieselben Aktionen, die Carol eben ausgeführt hat, um Victor zu betrügen. Er sagt mittels eines Bit $b'$ Victors Wunsch voraus und präpariert seine Lösungen. Im Falle der richtigen Vorhersage verläuft das Protokoll normal, andernfalls wird es

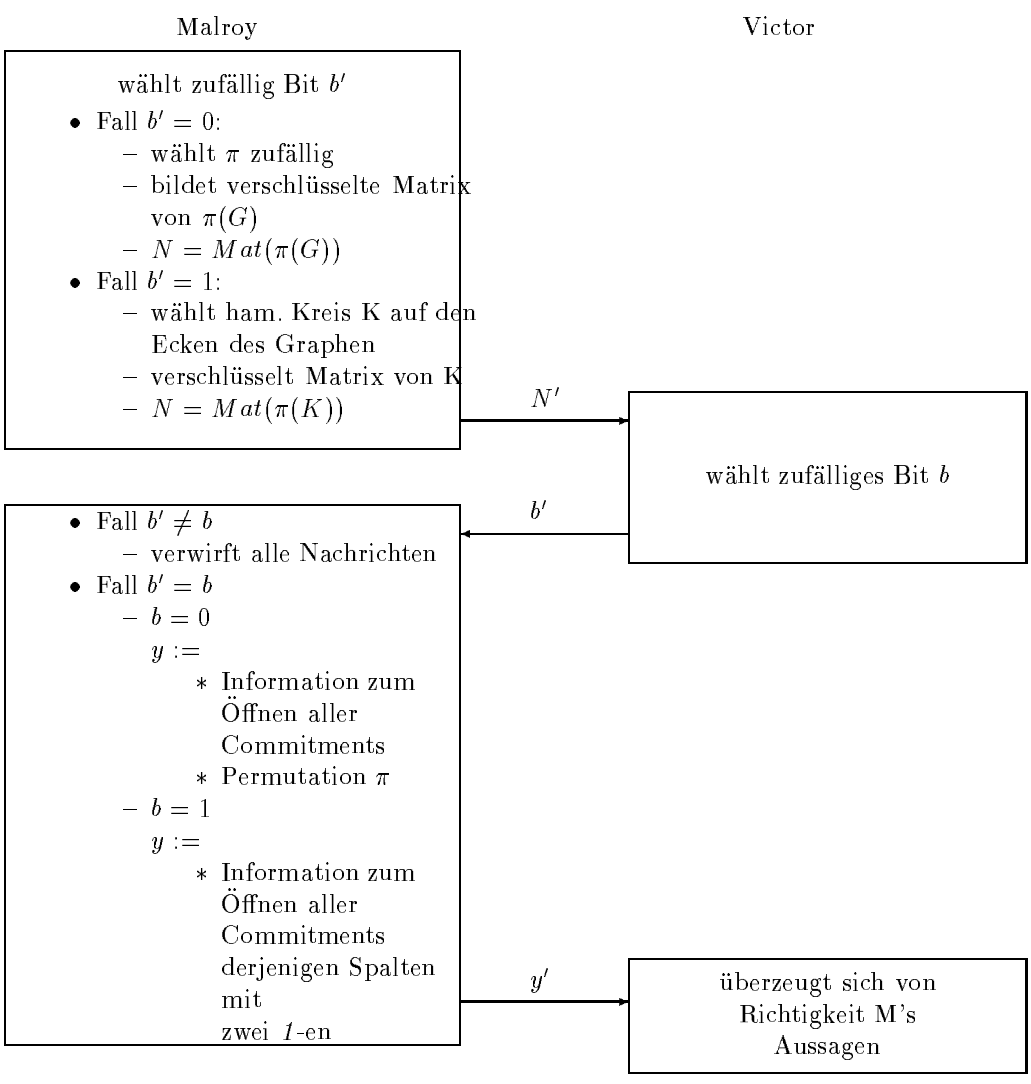



ABBILDUNG 8. Die ZK-Eigenschaft

abgebrochen und alle Nachrichten werden gelöscht[7]. Bild 8 verdeutlicht den Ablauf. Wie auch in der Originalinteraktion repräsentiert das Tupel $(N', b', y')$ die Nachrichten, welche über das Kommunikationsmedium gehen. Diese Tupel sind identisch zu den Nachrichten $(N, b, y)$ der Originalinteraktion, sind aber ohne geheimes Wissen erzeugt worden. Deshalb hat das Protokoll die ZK Eigenschaft[8].
Die NP-Vollständigkeit des Entscheidbarkeitsproblemes „Graph $G$ ist hamiltonisch oder nicht“führt zu einem besonderen Ergebnis: „Alle Probleme in NP besitzen ein Zero-Knowledge Protokoll“. Oder, wenn man ein Protokoll als Beweis für die Existenz eines hamiltonischen Kreises betrachtet: „Alle Probleme in NP besitzen einen Zero-Knowledge *Beweis*“. Dieses wichtige Ergebnis ist durch die polynomiale Reduzierbarkeit der NP-Vollständigen Probleme erklärbar: Zu jedem Paar von

[7]dies entspricht dem Löschen der „schlechten“Szenen auf dem Videoband in 2.3
[8]und eine mögliche Angreiferin Carol kann aus $(N, b, y)$ nichts lernen

Problemen $(P_1, P_2)$ , wobei $P_1$ in NPV und $P_2$ in NP liegt, gibt es einen polynomial zeitbeschränkten Algorithmus, der $P_1$ auf $P_2$ reduziert. Wenn somit das Problem „Ist Graph $G$ hamiltonisch oder nicht"einen ZK-Beweis hat, dann auch jedes andere Problem in NP.
Zum Ende des Kapitels möchte ich noch die „allgemeine Version"eines ZK Protokolles skizzieren: (siehe Bild
9)

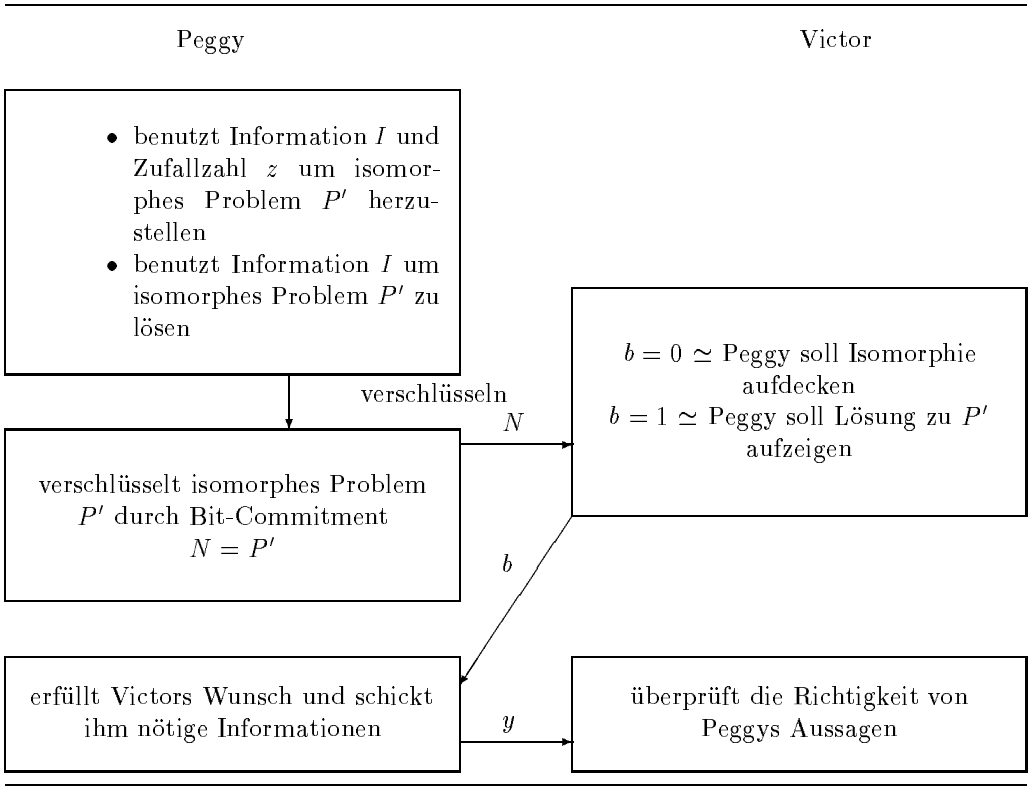



ABBILDUNG 9. Allgemeines ZK Protokoll

## 4. Weiterentwicklungen

Im Folgenden betrachten wir zwei Weiterentwicklungen von Zero-Knowledge Protokollen: *Parallele ZK Protokolle*, die aus der Motivation heraus entstanden sind, die Anzahl der über das Transportmedium gesendeten Nachrichten zu minimieren und *nicht-interaktive ZK Protokolle*, welche es Peggy ermöglichen, sich nicht nur Victor gegenüber identifizieren zu können, sondern jeder beliebigen Gruppe anderer Personen.

**4.1. Parallele ZK-Protokolle.** Möchten beispielsweise zwei Personen, die eine in Kaiserslautern, die andere in Sydney, über das Internet ein Zero-Knowledge Protokoll durchführen, so kann dieser Vorgang wegen der $t$-fachen Wiederholung des Protokolls einige Minuten dauern. Um die Übertragungszeit zu verringern, bedient man sich eines parallelen Protokolls, welches sich aus einem ZK Protokoll ableitet. Bild 10 zeigt die allgemeine Version.
Peggy stellt also nicht nur ein einzelnes isomorphes Problem her, sondern $n$. Victor antwortet mit einem $n$-Bit-Vector. Ist das $i$-te Bit eine 0, so muss Peggy die Isomorphie des $i$-ten Problems zum Ausgangsproblem beweisen, andernfalls eine Lösung aufzeigen. Durchführbarkeit und Korrektheit beruhen auf der Durchführbarkeit und der Korrektheit des zugrundeliegenden Original-Protokolles. Doch wie sieht es mit der ZK-Eigenschaft aus? Wählt Victor beim Generieren der Bit-Strings eine Einweg-Funktion, die als Eingabe Peggys Nachricht $N$ erhält, so ist das Protokoll *nicht länger*

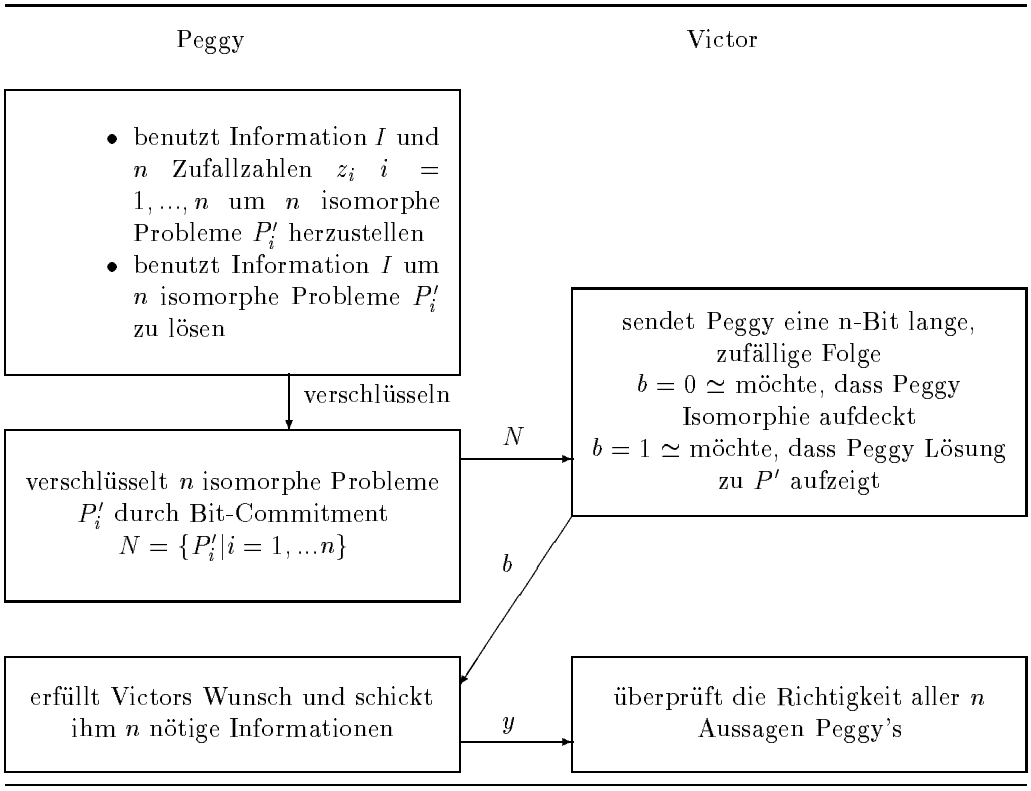



ABBILDUNG 10. Allgemeines $n$-paralleles ZK-Protokoll

*simulierbar* [**Sch96**], es hat also nicht die Zero-Knowledge Eigenschaft. Die Sicherheit, welche bei den parallelen Protokollen jetzt nicht mehr zwingend ist, macht man nun an einem neuen Begriff fest: Die Witness-Hiding Eigenschaft. Ein Protokoll ist genau dann Witness-Hiding, wenn Victor, auch nach mehrfacher Durchführung, Peggys Geheimnis nicht berechnen kann. Diese Eigenschaft lässt sich für ein gegebenes Protokoll beweisen. Interessanterweise besitzt nicht jedes parallelisierte ZK Protokoll die Witness-Hiding Eigenschaft. Genaueres lässt sich in [**BSW98**] nachlesen.

**4.2. Nicht-Interaktive ZK-Protokolle.** Wie wir am Anfang gesehen haben, leitete sich ein Zero-Knowledge Protokoll aus interaktiven Beweisen ab. Nun soll die Interaktivität unterbunden werden. Funktioniert dies überhaupt und welcher Sinn steckt dahinter?
Peggy kann Victor vom Besitz eines Geheimnisses überzeugen, jedoch nicht eine Beobachterin Carol (Peggy und Victor könnten sich ja vorher abgesprochen haben). Möchte Peggy jedoch jede beliebige Person von ihrem Besitz überzeugen, so muss sie ein nicht-interaktives Verfahren anwenden. Das heißt, sie muss all ihre Schritte aufschreiben, so dass jeder, der ihre Schritte nachvollzieht von Peggys „Echtheit"überzeugt wird. Im einzelnen läuft das „Protokoll"folgendermaßen ab:
Peggy's geheime Information $I$ ist Lösung eines NP-Problems $P$.
Sie:

(1) benutzt Information $I$ und $n$ Zufallzahlen $z_i$, $i = 1, \ldots, n$ um $n$ isomorphe Probleme $P_i'$ herzustellen und löst sie mittels $I$.
(2) verschlüsselt alle $P_i'$.
(3) generiert eine $n$-bit Zufallsfolge $F$ mittels Einweg-Funktion $f$:
$F$ entspricht den ersten $n$ Bit von $f(P_i')$, d.h. als Eingabe in die Einweg-Funktion benutzt sie die verschlüsselten Probleme $P_i'$ (die ja auch nur einen Bit-String darstellen).
(4) Sie betrachtet für $P_i'$ das $i$-te Bit der Folge $F$.
   (a) $b_i = 0$: Peggy beweist, dass $P$ und $P_i'$ isomorph sind.

(b) $b_i = 1$: Peggy zeigt, dass sie Lösung zu $P_i'$ kennt.

(5) Peggy veröffentlicht alle $P_i'$, die Hash-Funktion $f$ und ihre Lösungen.

(6) *Jeder* kann nachvollziehen, dass Peggy Information $I$ besitzt.

Durchführbarkeit und Korrektheit beruhen wieder auf dem Original-Protokoll. Es ist leicht einzusehen, dass die nicht-interaktive Version mit der selben Begründung wie im 4.1 vorher nicht die ZK Eigenschaft aufweist, jedoch analog WH ist. Wir überlegen uns noch kurz, wie eine Person Carol, die Peggys geheime Information $I$ nicht kennt, betrügen könnte:
Carol

- „rät"$n$-mal, ob sie in 4. (a) oder (b) ausführen muss.
- präpariert ihre Lösungen[9].
- führt Schritt 1,2 und 3 aus.
- schaut, ob sie richtig geraten hat, falls nicht, muss sie das Ergebnis der Einweg-Funktion in Schritt 3 manipulieren. Dies tut sie, indem sie die Eingabereihenfolge der Probleme $P_i'$ in die Einweg-Funktion $f$ ändert oder in 1. andere Isomorphien erzeugt.

Für eine kleine Bitstring-Länge $n$, lassen sich schnell einige Konstellationen berechnen, die den Betrug ermöglichen, es gibt ja nur $2^3 = 8$ verschiedene Möglichkeiten. Damit Peggys Beweis anerkannt wird, muss daher $n$ groß sein (mindestens 64, besser 128).

## 5. Wie man ZK-Protokolle hintergeht

**5.1. The chess grandmasters problem.** Die Idee, wie man ein ZK-Protokoll hintergeht, baut auf dem „chess grandmasters problem"auf. Es ist möglich ein Schachspiel gegen einen bekannten Großmeister zu gewinnen, sogar ohne selbst die Regeln von Schach zu kennen:

- Man lädt gleichzeitig zwei Großmeister (z.B. Garry Kasparov und Viswanathan Anand) zu sich ein und platziert sie an Schachtischen in zwei verschiedenen Räumen.
- Gegen Kasparov spielt man weiß und schwarz gegen Anand.
- Anand beginnt. Man merkt sich den Zug, wechselt den Raum und spielt ihn gegen Kasparov.
- Kasparovs Antwort merkt man sich ebenfalls, wechselt wieder den Raum und spielt gegen Anand.

Das Schachspiel findet also in Wirklichkeit zwischen Anand und Kasparov statt, selbst nimmt man nur eine „Vermittlerrolle"ein. Endet das Spiel nicht mit einem Unentschieden, so gewinnt man in jedem Fall gegen einen der beiden.

**5.2. The Mafia Fraud.** Das oben beschriebene Verfahren kann man einsetzen, um ein Zero-Knowledge Protokoll zu hintergehen. Wie schon mehrfach erwähnt, werden ZK Protokolle gerne bei Identifikationssystemen eingesetzt. Im letzten Beispiel sehen wir, wie sich Carol mit Hilfe von Bob als Alice ausgeben kann, und somit Dave betrügen:
Alice ist eine vertrauenswürdige Person. Sie befindet sich gerade bei einem Mittagessen in Bobs Restaurant. Bob und seine Komplizin Carol sind Mitglieder der Mafia, die beiden können sich über eine abhörsichere Leitung unterhalten. Carol befindet sich zur Zeit in Daves Juwelierladen. Carol möchte sich nun als Alice ausgeben, um Diamanten auf Alices Rechnung zu kaufen. Bob und Carol gehen dabei so vor:

- Dave fordert Carol auf sich durch ein ZK Protokoll zu identifizieren. Carol verständigt Bob.
- Bob fordert darauf Alice auf, sich mittels eines ZK Protokolles zu identifizieren.
- Alle Nachrichten, die Alice an Bob gibt, werden an Carol weitergeleitet. Carol benutzt diese um sich zu identifizieren. Alle Nachrichten, die Dave an Carol gibt, werden an Bob weitergeleitet, welcher sie Alice zustellt.

Bob und Carol spielen in diesem Verfahren die Vermittlerrolle, das eigentliche Protokoll läuft zwischen Alice und Dave ab.

[9] wie z.B. auch in Kapitel 2

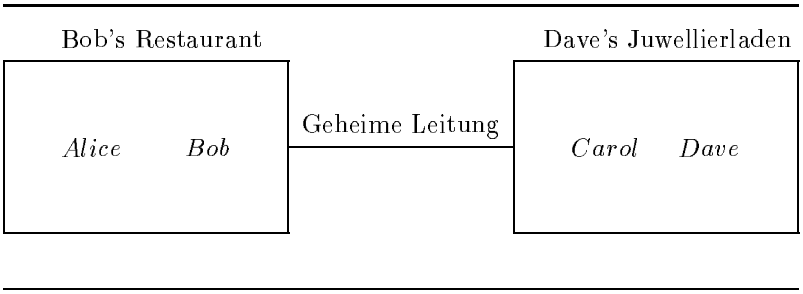



ABBILDUNG 11. The Mafia Fraud

## 6. Abschließende Betrachtung

Nach dieser kurzen Einführung in die Welt des Zero-Knowledge, fragt sich der Leser möglicherweise, ob und wie solche Protokolle eine praktische Anwendung finden. In Kapitel 3 sahen wir eine mögliche mathematische Realisierung und, gegen Ende des Kapitels, dass alle Probleme in NP einem ZK-Protokoll als Grundlage dienen können. Folglich gibt es natürlich viele verschiedene mathematische Realisierungen dieser Protokoll-Gattung. So zum Beispiel ein Verfahren, welches auf dem NPV Problem beruht, Quadratwurzeln in der primen Restklassengruppe $\mathbb{Z}_n^*$ zu berechnen. Dieses Verfahren wurde in Pay-TV Systemen benutzt und ist als „Videocrypt "bekannt: Ein solches System besteht aus zwei Komponenten. Einem Decoder, der keinerlei geheime Informationen enthält, und der frei verkauft wird, und einer Chipkarte, auf der alle nötigen Informationen gespeichert sind, welche zum entschlüsseln des Bildes nötig sind. Die Chipkarte wird in den Decoder eingeschoben. Das Problem ist nun, dass der Decoder „echte "Chipkarten von falschen „Piratenkarten ", ohne Verwendung geheimer Information, die er nach Voraussetzung nicht hat, unterscheiden muss. Zur Realisierung hat man sich hier für das oben beschriebene Verfahren entschieden.
Der an parallelen und nicht-interaktiven Protokollen interessierte Leser findet genaue Definitionen und einige praktische Beispiele in [**Weg96**]. Die „Mafia fraud "-Methode, welche in Kapitel 5 gezeigt wurde, eignet sich nicht nur, um ZK Protokolle zu hintergehen. In der Praxis lassen sich so alle Protokolle, die keinen entsprechenden Schutzmechanismus haben, hintergehen.

KAPITEL 7

# Multiparty Computing

**Christoph Thelen**

## 1. Einführung

Beim Multiparty Computing geht es darum, mit den Eingaben mehrerer Teilnehmer ein gemeinsames Ergebnis zu berechnen. Dabei sollen alle Eingaben der Teilnehmer geheim bleiben, das Ergebnis muss aber natürlich trotzdem "stimmen". Dafür benötigt man die Hilfe der Kryptologie, denn mit ihrer Hilfe kann man Verfahren und Protokolle entwerfen, die alle geforderten Eigenschaften besitzen.
Einige Beispiele dieser Verfahren werden in diesem Text gezeigt.

## 2. Secret Sharing Schemes

**2.1. Ziel.** Secret Sharing Schemes dienen zur Zerlegung eines Geheimnis derart in Teilgeheimnisse, so dass das Geheimnis nur aus bestimmten, vorher festgelegten Gruppen von Teilgeheimnissen wieder rekonstruiert werden kann.
Dabei darf kein Teilgeheimnis aus der Kenntnis beliebig vieler anderer Teilgeheimnisse berechnet werden können. Das Geheimnis darf nur aus vollständigen Gruppen rekonstruiert werden können.
**Beispiele:**

- **Schatzkarte:** Das "Geheimnis", nämlich der Lageplan des Schatzes wird in vier Einzelgeheimnisse aufgeteilt, die nur zusammen ein sinnvolles Ganzes ergeben.
- **Threshold-Verfahren (Schwellenverfahren):** Eine geheime Zahl soll durch andere Zahlen(-Paare) gespeichert werden, so dass sie nur mit Hilfe einer bestimmten Anzahl dieser Zahlen rekonstruiert werden kann. Diese Anzahl nennt man die "Schwelle".

**2.2. Threshold-Verfahren.** Ein Geheimnis $S$ wird in $n$ Teilgeheimnisse $S_1, ..., S_n$ aufgeteilt, so dass für $t \in \mathbb{N}$ (die "Schwelle") gilt:

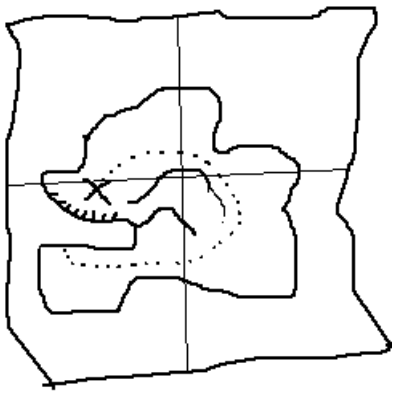


ABBILDUNG 1. Schatzkarte

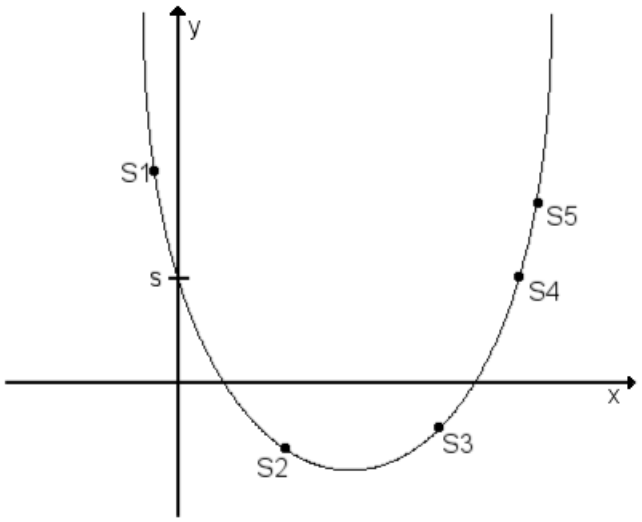



ABBILDUNG 2. Threshold-Verfahren

(1) Aus je $t$ oder mehr Teilgeheimnissen kann das Geheimnis rekonstruiert werden.
(2) Aus weniger als $t$ Teilgeheimnissen kann man das Geheimnis nicht berechnen, sondern höchstens mit einer sehr kleinen Wahrscheinlichkeit raten.

**Beispiel** für den Fall $t = 3$: Reelle Funktion 2. Grades.
Das Geheimnis $S$ ist dabei eine Zahl $s$ auf der y-Achse des Koordinatensystems.

Erzeugung der Teilgeheimnisse:
Wähle Polynom $f(x) = ax^2 + bx + s$, das durch $(0, s)$ läuft ($a, b$ beliebig)
Teilgeheimnisse = beliebige Punkte auf dem Graphen von $f$

Rekonstruktion von $S$:
Mit drei Teilgeheimnissen, also Punkten auf $f$ lässt sich $f(x)$ rekonstruieren und damit $f(0) = s$ berechnen.
Verallgemeinerung: Für eine beliebige Schwelle $t \in \mathbb{N}$ gilt:
Wähle Polynom $f(x) = a_{t-1}x^{t-1} + ... + a_1x + s$ vom Grad $t - 1$
Mit der Lagrange-Interpolation von $t$ Teilgeheimnissen (also Punkten auf $f$) ergibt sich wieder $f(x)$ und damit $s$

*Lagrange-Interpolation:* Das endliche Polynom $f(x)$ vom Grad $t-1$ durch die Punkte $(a_i, b_i), 1 \leq i \leq t$ ergibt sich durch $f(x) = \sum_{i=1}^{t} \frac{(x-a_1)\cdot\ldots\cdot(x-a_{i-1})(x-a_{i+1})\cdot\ldots\cdot(x-a_t)}{(a_i-a_1)\cdot\ldots\cdot(a_i-a_{i-1})(a_i-a_{i+1})\cdot\ldots\cdot(a_i-a_t)} \cdot b_i$

In der Praxis verwendet man endliche Körper. Der Vorteil dabei ist, dass keine Rundungsfehler auftreten

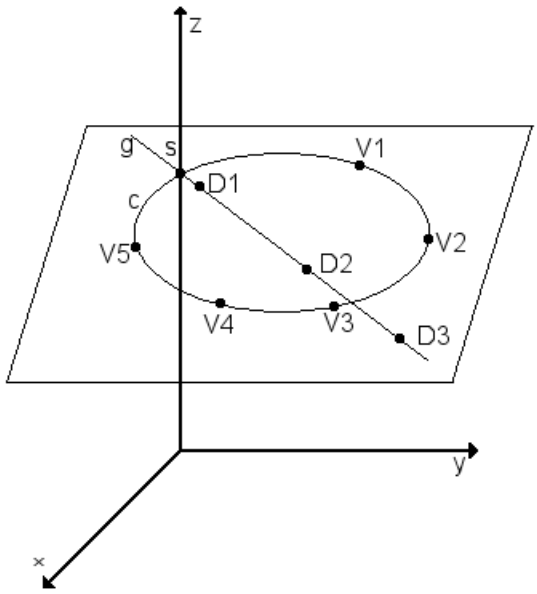



ABBILDUNG 3. Secret Sharing Scheme

**Ein komplexeres Verfahren:**
Ein Tresor soll sich öffnen, wenn

- 2 Direktoren
- 3 Vizedirektoren oder
- 1 Direktor und 2 Vizedirektoren

einverstanden sind.

Das Geheimnis $S$, also der Tresorcode sei $s \in \mathbb{R}$ auf der z-Achse im dreidimensionalen Raum, d.h. $S = (0, 0, s)$
Wähle Ebene $E$ durch $S$, die nicht die z-Achse enthält
Dann wähle:

- Eine Gerade $g$, die auf $E$ liegt und durch $S$ läuft.
  Die "Direktorenpunkte" $D_1, D_2, ...$ sind dann beliebige Punkte auf $g$
- Einen Kreis $c$ in $E$ durch $S$, darauf die "Vizedirektorenpunkte" $V_1, V_2, ....$

Dann kann $S$ aus den o.g. Kombinationen rekonstruiert werden, indem entweder die Gerade $g$, der Kreis $c$ oder die Ebene $E$ aus den gegebenen Punkten berechnet wird. $S$ ist dann der Schnitpunkt der jeweiligen geometrischen Figur mit der z-Achse.

## 3. Einfaches Multiparty Computing einer Funktion

Für eine gegebene Funktion $f$ soll das Ergebnis der Eingaben mehrerer Teilnehmer, die die Eingaben der anderen Teilnehmer nicht kennen, bzw. kennenlernen dürfen, berechnet werden.

**3.1. Beispiele: Durchschnitsgehalt:** Die Angestellten $A$, $B$ und $C$ verdienen $a, b, c$ DM/Monat. Gesucht ist nun der Durchschnitt von $a, b, c$, also $f(a, b, c) = \frac{a+b+c}{3}$

*Vorgehensweise:*

(1) $A$ wählt eine Zufallszahl $r$ und gibt $a + r$ an $B$ weiter.
(2) Der addiert sein eigenes Gehalt und schickt $a + b + r$ an $C$.
(3) $C$ addiert $c$ und gibt $a + b + c + r$ an $A$ zurück.
(4) Da $A$ (und nur er) die Zufallszahl $r$ kennt, kann er $a+b+c$ und damit $f(a, b, c)$ berechnen. Dieses Ergebnis teilt er $B$ und $C$ mit.

Dabei treten allerdings Probleme auf:

- Alle Teilnehmer müssen vertrauenswürdig sein, da ein falsches Ergebnis berechnet wird, falls einer eine falsche Eingabe macht.
- Bei einer Verschwörung von zwei Teilnehmern kann das Gehalt des dritten unbefugt errechnet werden.

**Skat über das Telefon:** Gezeigt wird hier ein Protokoll für das sichere Austeilen und Spielen von Karten über grosse Entfernungen ohne Betrugsmöglichkeit.
Die Spieler $A$, $B$ und $C$ einigen sich auf eine (große) Primzahl $p$ und wählen entsprechend $a, a', b, b', c, c'$ mit $aa' \equiv bb' \equiv cc' \equiv 1(mod\ p-1)$
(Nach dem Kleinem Satz von Fermat gilt:$(g^a)^{a'}\ mod\ p = g^{aa'\ mod\ p-1}$, also $(g^a)^{a'}\ mod\ p = g^1$. Diese Eigenschaft wird unten wichtig.)

**Mischen:**
Das Kartenspiel $K = \{(1, x_1), ..., (32, x_{32})\}$ soll gemischt werden

Dazu wählt $A$ eine Permutation $\alpha$ der Menge $\{1, ..., 32\}$ und schickt
$K' = \{(\alpha(1), x_1^a\ mod\ p), ..., (\alpha(32), x_{32}^a\ mod\ p)\}$ an $B$, wobei er die Karten nach der ersten Komponente sortiert.

$B$ mischt nun mit Permutation $\beta$:
$K'' = \{(\beta(\alpha(1)), (x_1^a)^b\ mod\ p), ..., (\beta(\alpha(32)), (x_{32}^a)^b\ mod\ p)\}$

$C$ erhält das wiederum sortierte $K''$ und mischt mit $\gamma$:
$K''' = \{(\gamma(\beta(\alpha(1))), ((x_1^a)^b)^c\ mod\ p), ..., (\gamma(\beta(\alpha(32))), ((x_{32}^a)^b)^c\ mod\ p)\}$
und sortiert ebenfalls nach der ersten Komponente

**Austeilen:**
Eine Karte für $C$ wir nach folgendem Verfahren ermittelt:

(1) $A$ nimmt die oberste Karte $(1, x_1^{abc}\ mod\ p)$ ($i$ mit $\gamma(\beta(\alpha(i))) = 1$) vom Stapel und berechnet $(x_i^{abc})^{a'} \equiv x_i^{bc}(mod\ p)$ (gilt wegen der oben beschriebenen Eigenschaft)
(2) Nun entfernt $B$ seine Verschlüsselung:
$(x_i^{bc})^{b'} \equiv x_i^c(mod\ p)$
(3) $C$ erhält $(1, x_i^c\ mod\ p)$ und kann durch
$(x_i^c)^{c'} \equiv x_i(mod\ p)$
den Wert der Karte berechnen.

Analog für Karten für $B$, sie durchlaufen den Weg $C - A - B$, bzw. Karten für $A$ gehen über $B - C - A$.

**Spielen:**
Die Karten können nun einfach durch Bekanntgabe des Wertes ausgespielt werden. Nach Spielende geben die Spieler ihre Zahlen $a, a', b, b', c, c'$ und Permutationen $\alpha, \beta, \gamma$ bekannt, damit kontrolliert

werden kann, ob niemand betrogen hat.

Dieses Protokoll ist sicher, denn keiner der Spieler kann nach dem Austeilen mehr als seine eigenen Karten kennen. Die Karten können nach dem Mischen nur durch Entfernen aller Verschlüsselungen sichtbar gemacht werden. Wenn also keiner der Spieler den Schlüssel eines anderen hat, kann er dessen Karten nicht einsehen. Das Veröffentlichen der Schlüssel und Permutationen am Schluss dient dazu festzustellen, ob niemand eine Karte ausgespielt hat, die er gar nicht erhalten hatte.

## 4. Secure Circuit Evaluation

**4.1. Ziel/Anforderungen.** Auswerten eines Wertes einer Partei $A$ mit dem Algorithmus einer anderen Partei $B$, wobei $A$ keine Kenntnis des Algorithmus und $B$ keine Kenntnis des Wertes erlangen darf.

Sei $f$ die Funktion zum Algorithmus von $B$, $x$ der zu berechnende Wert von $A$.
Gesucht ist also $f(x)$. Voraussetzung dafür: $f$ ist als logischer Schaltkreis gegeben mit den Operationen $\wedge, \vee$ und $\neg$

**4.2. Durchführung.**

(1) $A$ wandelt $x$ in Bitschreibweise um: $x = b_m, ..., b_2, b_1$
(2) $A$ verschlüsselt die Bits mit einem Bit-Commitment Verfahren (s.[**BSW98**]) und sendet sie an $B$.
(3) $B$ berechnet (z.T. mit Hilfe von $A$) die Funktion und sendet das Ergebnis an $A$

**4.3. Verschlüsselung der Bits.** (Bezeichnung: Operation $E$)
$A$ wählt zwei Primzahlen $p, q$ mit $p \ mod \ 4 = 3$ und $q \ mod \ 4 = 3$.
(dann hat -1 das Jacobisymbol +1 und ist quadratischer Nichtrest modulo $n = pq$)

Für jedes Bit $b_i$ wählt $A$ dann ein zufälliges $k$ und rechnet $k^2$ modulo $n$. Für $b_i = 0$ ist $E$ gleich dieser Zahl, also quadratischer Rest modulo $n$. Für $b_i = 1$ wird mit (-1) multipliziert, also ist $E$ ein quadratischer Nichtrest modulo $n$
$\Longrightarrow E(b_i) := (-1)^{b_i} k^2 \ mod \ n$

*Entschlüsselung:* Operation $D$:

$$D(c) = \begin{cases} 0 & \text{falls } c \text{ quadratischer Rest modulo } n \\ 1 & \text{sonst} \end{cases}$$

Damit gilt $D(E(b)) = b$
Diese Rechnung ist einfach für $A$, weil er $p, q$ kennt, aber schwer für $B$, weil er nur $n$ kennt und deshalb nicht entscheiden kann, ob $c$ ein quadratischer Rest ist oder nicht. Dieses Problem hängt eng mit der Faktorisierung von $n$ zusammen

**4.4. Berechnung durch den logischen Schaltkreis.** Notwendig ist dafür die Berechnung von $\wedge$ und $\neg$, da $\vee$ durch $a \vee b = \neg(\neg a \wedge \neg b)$ simuliert werden kann.

4.4.1. *Negation.* Dazu wählt $B$ ein zufälliges $r \in \mathbb{N}$. Dann gilt $E(\neg b) = (-1) r^2 E(b) \ mod \ n$

*Beweis:* Fallunterscheidung

$b = 0$: $E(b) = k^2 \ mod \ n$, also quadratischer Rest modulo $n$. Einsetzen:
$E(\neg b) = (-1)r^2E(b) \ mod \ n = (-1)r^2k^2 \ mod \ n = (-1)(rk)^2 \ mod \ n$
also quadratischer Nichtrest modulo $n$

$b = 1$: $E(b)$ ist quadratischer Nichtrest modulo $n$
$E(\neg b) = (-1)r^2E(b) \ mod \ n = (-1)r^2(-)k^2 \ mod \ n = (rk)^2 \ mod \ n$
also quadratischer Rest modulo $n$

Die "eigentliche" Negationsoperation ist die Multiplikation mit (-1) (Das Vertauschen von quadratischer Rest/Nichtrest). Die Zahl $r$ verhindert, dass $A$ herausfindet, welche Bits negiert werden. Er könnte sonst Informationen über $f$ sammeln.

4.4.2. *Konjunktion:* $b \wedge b'$. Dazu benötigt $B$ die Hilfe von $A$. Es ist ein offenes Problem, ob es Systeme gibt, wo dies nicht notwendig ist.
Auch hier gilt: $A$ soll möglichst wenig über den Schaltkreis lernen. Er soll weder wissen, welche Bits verknüpft werden, noch welche Werte die verknüpften Bits haben. Es müssen also Identität und Wert verschleiert werden.
$B$ wählt dafür die Zufallszahlen $r$ und $r'$ und die zufälligen Bits $c, c'$ und berechnet
$E(d) = (-1)^c r^2 E(b) \ mod \ n$ und $E(d') = (-1)^{c'} r'^2 E(b') \ mod \ n$. Dann ist der Wert von $d$ gleich dem Wert von $b$ für $c = 0$ und gleich dem Wert von $\neg b$ für $c = 1$. Analog für $d'$.

$A$ kann damit $d = D(E(d))$ und $d' = D(E(d'))$ berechnen und sendet in fester Reihenfolge an $B$:

| Gesendeter Wert | $E(b \wedge b')$ falls |
|---|---|
| $E(d \wedge d')$ | $c = 0, \ c' = 0$ |
| $E(d \wedge \neg d')$ | $c = 0, \ c' = 1$ |
| $E(\neg d \wedge d')$ | $c = 1, \ c' = 0$ |
| $E(\neg d \wedge \neg d')$ | $c = 1, \ c' = 1$ |

Das ist nötig, weil $A$ die Werte von $b$ und $b'$ nicht kennt und somit $b \wedge b'$ nicht direkt berechnen kann. $B$ dagegen kennt $c, c'$ und wählt den empfangenen Wert entsprechend aus.

Somit ist die Berechnung von $f(E(b_m), ..., E(b_1))$ möglich, die Ausgabe ist $E(a_l), ..., E(a_1)$.
$A$ berechnet dann $a_l, ..., a_1$ und damit $f(x)$.

Es werden alle Anforderungen erfüllt: Die Geheimhaltung des Wertes von $A$ ist gewährleistet, weil $B$ nur mit den verschlüsselten Bits rechnet und keine Möglichkeit hat, die Verschlüsselung zu brechen. Umgekehrt erfährt $A$ nichts über den Algorithmus, weil $B$ alle Rechnungen und von $A$ berechneten Werte mit Zufallszahlen verschleiert.

## 5. Anonymität

Es werden drei Arten der Anonymität unterschieden:

- Anonymität des Senders
- Anonymität des Empfängers (möglich z.B. durch Broadcasting)
- Anonymität der Kommunikationsbeziehung (Verbergen von Sender und Empfänger voreinander und vor anderen)

**5.1. Dining Cryptographers Protocol.** Dieses Protokoll dient der Anonymität des Senders

Die drei Kryptographen $A, B$ und $C$ sitzen beim Essen im Restaurant und treffen folgende Verabredung mit dem Inhaber:
Entweder zahlt einer von ihnen oder ihr Arbeitgeber.
Der "Bezahler" bleibt dabei anonym, aber sie wollen wissen, ob jemand von ihnen oder ihr Arbeitgeber zahlt.
Dazu verabreden sie mit ihren jeweiligen Nachbarn ein geheimes Bit. Also $A$ und $B$ vereinbaren Bit $b_{AB}$ usw..

Jeder "Nichtzahler" addiert dann die ihm bekannten Bits modulo 2.
z.B. $A$ schreibt $b_{AB} + b_{AC}\ mod\ 2$ auf (wenn er nicht zahlt).

Der "Bezahler" unter ihnen (wenn es denn einen gibt) addiert noch 1 zu seinem Ergebnis.
Zahlt $C$, notiert er also $b_{AC} + b_{BC} + 1\ mod\ 2$

Die Ergebnisse werden modulo 2 addiert. Steht im Endergebnis eine 0, so zahlt der Arbeitgeber, denn :
$(b_{AB} + b_{AC}) + (b_{AB} + b_{BC}) + (b_{AC} + b_{BC})\ mod\ 2 =$
$(b_{AB} + b_{AB}) + (b_{AC} + b_{AC}) + (b_{BC} + b_{BC})\ mod\ 2 = 0$
Die geheimen Bits treten je genau zweimal auf und addieren sich so immer zu $0\ mod\ 2$.

Bei einer 1 zahlt einer der Kryptographen, denn es gilt:
$(b_{AB} + b_{AC}) + (b_{AB} + b_{BC}) + (b_{AC} + b_{BC} + 1)\ mod\ 2 = 1$ (In diesem Beispiel ist für $A$ und $B$ nicht nachvollziehbar, wer zahlt)

**Problem:** Zahlen alle drei, tritt ebenfalls 1 auf. Der Inhaber kann in diesem Fall dreimal kassieren, ohne dass es von den Kryptographen bemerkt wird. Zahlen zwei der Kryptographen, wird es bemerkt, da beide eine 1 erwarten.

**5.2. MIXe.** MIXe werden zum Verschleiern der Kommunikationsbeziehung eingesetzt.
Sie gewährleisten Anonymität der Verbindung zwischen Sender und Empfänger, nicht nur gegenüber Abhören, sondern auch gegenüber dem Kommunikationsvermittler.

Ein MIX besitzt zwei Grundfunktionen:

(1) *Auspacken und Weiterschicken:* Der MIX erhält eine verschlüsselte Nachricht mit einer Zieladresse, entschlüsselt beides und leitet sie weiter.
(2) *Mischen:* Entschlüsselte Nachrichten werden nicht in Reihenfolge des Eingangs weitergeschickt, sondern vermischt.

Ein Beispiel mit zwei MIXen, $MIX_1$ und $MIX_2$ und einem asymetrischen Verschlüsselungsverfahren (z.B. RSA):
$MIX_i$ hat einen öffentlichen $(E_i)$ und einen privaten Schlüssel $(D_i)$.

Drei Sender $(A_1, A_2, A_3)$ wollen je eine Nachrichte $(m_1, m_2, m_3)$ über die MIXe an die Empfänger $B_1, B_2$ und $B_3$ verschicken.

Zuerst verschlüsselt $A_1$ $m_1$ und die Adresse von $B_1$ mit $E_2$, fügt die Adresse von $MIX_2$ an und verschlüsselt mit $E_1$.

Dies wird an $MIX_1$ geschickt. $A_2, A_3$ handeln analog.

$MIX_1$ entschlüsselt die Nachrichten mit $D_1$, vertauscht die Reihenfolge und schickt die Nachrichten $E_2(B_i, m_i)$ an $MIX_2$.

Der entschlüsselt mit $D_2$ und und schickt die Nachrichten an die entsprechenden Empfänger.

$$\begin{array}{lll} A_1 \xrightarrow{E_1(MIX_2, E_2(B_1, m_1))} & & \xrightarrow{E_2(B_1, m_1)} \\ A_2 \xrightarrow{E_1(MIX_2, E_2(B_2, m_2))} & MIX_1 & \xrightarrow{E_2(B_2, m_2)} \\ A_3 \xrightarrow{E_1(MIX_2, E_2(B_3, m_3))} & & \xrightarrow{E_2(B_3, m_3)} \end{array} \quad \begin{array}{l} MIX_2 \\ \\ \\ \end{array} \quad \begin{array}{l} \xrightarrow{m_1} B_1 \\ \xrightarrow{m_2} B_2 \\ \xrightarrow{m_3} B_3 \end{array}$$

Es ist auch eine Kommunikation über mehr als 2 MIXe möglich. Der Sender legt dann vorher den Vermittlungsweg fest.
Wird nur ein einzelner MIX verwendet, so kann dieser die Kommunikationsbeziehung ermitteln, zwei oder mehr unabhängige MIXe können das nicht

**5.3. Elektronisches Geld.** Physikalische Münzen sind anonym, man kann nicht erkennen, wer sie vorher ausgegeben hat. Bei Schecks und Kreditkarten dagegen ist das nicht der Fall, sie hinterlassen verfolgbare Spuren.
Das Ziel ist also die Entwicklung von anonymen elektronischen Geld.

5.3.1. *Anforderungen.*

- *Anonymität:* Einlösende Bank weiss nicht, wem das Geld gehört hat.
- *Sicherheit:* Kunde, Händler und Bank können von der Gültigkeit der Elektronischen Münze überzeugt sein.
  Ausserdem nötig ist ein Schutz vor doppeltem Ausgeben, denn das Kopieren von elektronischem Geld ist sehr einfach.

5.3.2. *Realisierung.*

- Die Bank besitzt geheime RSA-Schlüssel für jeden Münzwert.
- Münzen müssen beim Nachprüfen mit dem entsprechendem öffentlichen Schlüssel der Bank in ein Redundanzschema passen, wie zwei gleiche Hälften. (z.B. 1234512345 passt in dieses Schema).

**Erzeugung einer Münze:**
Der Kunde wählt eine Zahl, z.B. $v = 1234567$ und bildet $w = 12345671234567 \Longrightarrow w$ passt ins Redundanzschema.
$w$ muss dabei kleiner als das Modul des RSA-Schlüssels sein, sonst wird das Redundanzschema beim Verschlüsseln zerstört.
Dann wird $w$ blind von der Bank mit entsprechendem RSA-Schlüssel signiert (sofort nachprüfbar mit öffentlichem Schlüssel).
(Blinde Signatur: s. [**BSW98**])

**Ausgeben der Münze:**
Der Kunde überträgt dann die Münze an den Händler. Der prüft den Wert der Münze mit dem öffentlichen Schlüssel der Bank und akzeptiert, wenn das Redundanzschema passt.

**Einlösen der Münze:**
Der Händler sendet die Münze an die Bank, welche die Echtheit prüft wie der Händler vorher und ihm dann entsprechend Geld überweist.
Die Anonymität des Kunden ist wegen der blinden Signatur gewährleistet.

**Probleme:**

- Ein Kunde kann seine Münze mehrmals bei verschiedenen Händlern ausgeben, ohne dass die Händler es bemerken.
- Auch der Händler kann versuchen, die Münze doppelt einzulösen (oder erst einzulösen und dann selber damit zu bezahlen).
- Allgemeines Problem: Der Schuldige des doppelten Ausgebens kann nicht ermittelt werden.

**Ein sicheres Protokoll**

Nachfolgend wird ein komplexeres Protokoll zum Beheben der oben genannten Probleme beschrieben.

(1) Der Kunde erzeugt $m$ anonyme Geldstücke eines bestimmten Wertes mit je:
- einer zufälligen Eindeutigkeitszeichenkette $X$
- $n$ Paaren von Identitätsbit-Wörtern $I_1, ..., I_n$:
  Dazu erzeugt der Kunde einen String mit seinem Namen, Adresse usw..
  Dieser wird mit einem geheimen Teilungsprotokoll $n$-mal in zwei Teile aufgeteilt.
  Die Teile werde mit einem Bit-Commitment Verfahren (s. [**BSW98**]) verschlüsselt.
  Der Kunde (und nur er) kann diese Verschlüsselung rückgängig machen, wobei die Korrektheit sofort verifiziert werden kann, d.h. jeder kann prüfen, ob der Kunde einen Teilstring richtig entschlüsselt hat.

Die Münzen sehen dann aus wie folgt:

| | |
|---|---|
| Betrag | $k$ |
| Eindeutigkeitsstring | $X$ |
| Identitäts-Strings | $I_1 = (I_{1_L}, I_{1_R})$ |
| | ... |
| | $I_n = (I_{n_L}, I_{n_R})$ |

(2) Der Kunde "verblendet" alle Geldstücke mittels eines blinden Signatur-Protokolls.
(3) Die Bank fordert den Kunden auf, $m-1$ Geldstücke sichtbar zu machen und prüft jeweils, ob Betrag $X$ und die ebenfalls entschlüsselten Identitäts-Strings korrekt sind.
(4) Die Bank signiert das übrige Geldstück blind und sendet es zurück.
(5) Der Kunde macht es sichtbar und gibt es aus.
(6) Der Händler überprüft die Banksignatur.
(7) Der Händler wählt einen zufälligen $n$-Bit Auswahlstring $b_n, ..., b_1$
(8) Kunde entschlüsselt je eine Hälfte der Identitäts-Strings, z.B.
$I_{i_L}$ wenn $b_i = 0$ und $I_{i_R}$ wenn $b_i = 1$
(9) Händler bringt Geldstück zur Bank.
(10) Die Bank überprüft ihre Signatur und ob $X$ schon in ihrer Datenbank gespeichert ist.
Wenn nicht, bekommt der Händler sein Geld, die Bank zeichnet $X$ und alle entschlüsselten Identitäts-Strings auf.
(11) Ist $X$ schon in der Datenbank wird das Geldstück nicht akzeptiert.
Wenn die Identitäts-Strings mit den gespeicherten Strings übereinstimmen, hat der Händler das Geld kopiert. Die Chance, dass der Kunde betrogen hat, ist in diesem Fall $(\frac{1}{2})^n$.
Stimmen die Identitäts-Strings nicht überein, hat der Kunde die Münze kopiert.
Da der Auswahlstring verschieden ist, gibt es nun ein Paar, bei dem beide Hälften entschlüsselt sind. Die Identität des Kunden wird damit aufgedeckt.

Das Protokoll ist sicher.

- Der Kunde kann zwar betrügen, wird aber überführt. Das doppeltem Ausgeben einer Münze bemerkt die Bank wegen des gleichem $X$ und kann in 11 den Kunden überführen
- Der Kunde kann versuchen, ein falsches Geldstück signieren zu lassen. Die Chance ist $\frac{1}{m}$. Bei entsprechend abschreckenden Strafen dürfte niemand dieses Risiko eingehen.
- Der Händler kann nicht betrügen, da ein doppelter Auswahlstring bemerkt würde.

- Sogar ein Zusammenschluss von Kunde und Händler kann nicht betrügen, wegen $X$ und der Signatur der Bank.
- Die Bank hat bei korrektem Ablauf keine Möglichkeit, die Identität des Kunden zu entschlüsseln, weil sie die Münze blind signiert hat.
- Auch ein Zusammenschluss Bank-Händler kann dies nicht.
- Ein Angreifer von außen kann betrügen:
  Wenn er den Transfer zwischen Kunde und Händler abhört und das Geldstück vor dem Händler bei der Bank einlöst, wird der Händler als Schuldiger erkannt.
  Deshalb müssen Kunde und Händler das Geldstück schützen wie physikalisches Geld.

**Das perfekte Verbrechen**

(1) Ein Verbrecher entführt ein Kind.
(2) Der Verbrecher erzeugt beliebig viele Geldstücke für beliebige Beträge.
(3) Der Verbrecher verblendet die Geldstücke und sendet sie an die Polizei mit der Anweisung:
  (a) Eine Bank soll sie blind signieren.
  (b) Die Ergebnisse sollen in einer Zeitung veröffentlicht werden.
(4) Verbrecher kauft eine Zeitung, macht die Geldstücke sichtbar und gibt sie aus. Die Polizei hat keine Möglichkeit zur Verfolgung des Geldes (s.o.).
(5) Der Verbrecher befreit das Kind.

**5.4. Elektronische Wahlen.** Der Vorteil gegenüber der "klassischen" Wahl: Das Ergebnis liegt sofort vor. Dabei müssen aber die selben Voraussetzung gegeben sein:

5.4.1. *Anforderungen.*

- Jeder Bürger hat genau eine Stimme.
- Abgegebene Stimmen können nicht mit dem Wähler in Verbindung gebracht werden.
- Keine Stimme kann von irgendwem vervielfältigt werden.
- Keine Stimme kann von irgendwem unentdeckt verändert werden.
- Wahlhelfer (ideal: jeder Bürger) können die Korrektheit der Wahl überprüfen.

5.4.2. *Zwei Realisierungen.*

(1) Der Bürger $B$ wählt eine Zahl $x$ mit bestimmtem Redundanzschema
  Die Wahlbehörde $W$ signiert $x$ blind und sendet den "Wahlzettel" $z$ verblendet an $B$ zurück.

| $B$ | | $W$ |
|---|---|---|
| wählt $x$ und zufälliges $r$ | | |
| bildet $y = xr^{e_W}$ | $\xrightarrow{y}$ | |
| | | berechnet $t = y^{d_W}$ |
| | $\xleftarrow{t}$ | |
| berechnet $z = \frac{t}{r} = x^{d_W}$ | | |

Dann wählt $B$ $n = pq$, wobei $p$ und $q$ Primzahlen sind. $z = e_B$ ist sein öffentlicher Schlüssel. $B$ berechnet den privaten Schlüssel $d_B$ zu $e_B$ und veröffentlicht $(z, n)$.
Um die Anonymität zu gewährleisten, sendet $B$ $(z, n)$ über einen MIX an $W$. Dieses Paar wird an eine öffentliche Tafel geschrieben. $B$ kann dann prüfen, ob sein Paar richtig angekommen ist.

Zur Wahl: $B$ hat eine nummerierte Liste der Parteien, verschlüsselt die Nummer $k$ seiner Wahl mit $d_B$ und sendet $(e_B, k^{d_B})$ über einen MIX an $W$.

Auszählung: $W$ und jeder Bürger kann die Stimmen entschlüsseln und zählen, da alle $(e_B, k^{d_B})$ bekannt sind. Sie berechnen damit $(k^{d_B})^{e_B} \ mod \ n = k$.

$W$ weiss, dass jeder Bürger nur eine Stimmte abgegeben hat, denn zu jedem Wahlzettel

akzeptiert sie nur eine Stimme und jeder gültige Schlüssel ist auch ein Wahlzettel (überprüfbar mittels des Redundanzschemas). Keine Stimme kann von Anderen verändert werden, da $d_B$ geheim ist.

**Problem:** $W$ kann die Wahl fälschen, indem sie einfach mehr Stimmen als tatsächlich abgegebene auflistet und zählt.

(2) Ein besseres Protokoll:
   (a) $W$ veröffentlicht eine Liste der Wahlberechtigten.
   (b) Bis zu einem Stichtag teilt jeder Bürger $W$ mit, ob er an der Wahl teilnimmt.
   (c) $W$ veröffentlicht Liste mit allen teilnehmenden Wählern.
   (d) Jeder Wähler $B$ erhält über ein geheimes Protokoll eine Identifikationsnummer $I$, die nicht bei $W$ gespeichert ist.
   (e) $B$ generiert ein asynchrones Schlüsselpaar $k, d$. Seine Wahl sei $v$. Er sendet anonym (z.B. über einen MIX) an $W$: $I, E_k(I, v)$
   (f) $W$ veröffentlicht eine Liste mit den eingegangenen Stimmen $E_k(I, v)$
   (g) $B$ sendet $I, d$ an $W$
   (h) $W$ entschlüsselt die Stimmen. Nach der Wahl veröffentlicht $W$ das Ergebnis und für jede Partei die Liste aller Stimmen $E_k(I, v)$ für diese Partei.
   (i) Stellt $B$ fest, dass seine Stimme nicht korrekt gezählt wurde, sendet er an $W$: $I$, $E_k(I, v)$, $d$.

In Schritt 1-3 wird die absolute Nummer der Wähler festgestellt. Selbst wenn nicht alle wählen, kann $W$ die Wahl nur noch sehr begrenzt abändern, weil eine Höchstmenge an Stimmen bekannt ist.
Sollten zwei Wähler ein gleiches $I$ zugewiesen bekommen wählt $W$ eine Stimme aus, generiert eine neue Nummer $I'$ und veröffentlicht $I', E_k(I, v)$. Der Besitzer dieser Stimme merkt dies und sendet nun seine Wahl erneut mit dem neuen $I'$.
In Schritt 6 kann $B$ prüfen, ob seine Stimme richtig angekommen ist. Wenn nicht, kann er dies in Schritt 9 beweisen.

**Probleme:**

- Ein korruptes $W$ kann die Stimmen der angemeldeten Wähler herausfinden, die nicht wirklich gewählt haben, und deren Stimmen beliebig vergeben.
- Das Protokoll zur Vergabe der Identifikationsnummern ist sehr komplex.
- $W$ kann sich weigern, eine Stimme zu zählen: $B$ behauptet, $W$ hätte seine Stimme absichtlich ignoriert, $W$ sagt, dass $B$ nie gewählt hat.

## 6. Zusammenfassung

Abschliessend lässt sich sagen, dass die aufgezählten Verfahren in der nächsten Zeit an Bedeutung zunehmen werden. Besonders die im letzten Kapitel beschriebenen Protokolle dürften eine wichtige Rolle spielen, wenn es zur Verbreitung von elektronischem Geld kommt. Die mathematischen Grundlagen dafür sind gegeben. Sollte jedoch jemand einen schnellen Algorithmus zur Faktorisierung großer Zahlen finden, müssen neue Verfahren der Verschlüsselung gefunden werden.

KAPITEL 8

# Oblivious Transfer

**Wei Zheng**

## 1. Einführung

*Oblivious Transfer* (*Übertragung ohne Gedächtnis* auf Deutsch) ist der Name eines auf den ersten Blick seltsam anmutenden Protokolls, das eine große Bedeutung hat. Es ist eine Art 'kontrolliertes Zufallsexperiment' und ist zuerst im Zusammenhang mit der Unterzeichnung von Verträgen beschrieben worden.

## 2. Anwendungsbeispiele

In diesem Abschnitt werden die Anforderungen von Oblivious Transfer anhand einiger Beispiel verdeutlicht.

**2.1. Faktorisierung.** Bob ist ein Kryptograph, er muss ein 500-Bit Nummer entschlüsseln. Er weiß nur, dass diese Nummer ein Produkt von fünf 100-Bit Nummern ist. Alice kennt nur einen Faktor dieser Nummern, und sie wird diesen Faktor verkaufen zu einem Preis von nur 100 DM. Das heißt pro DM ein Bit. Das ist sehr interessant für Bob. Aber Bob hat nur 50 DM dabei. Alice wird ihren Preis nicht verändern. Dann schlägt sie vor, dass sie ihm 50 Bits Faktor verkaufen kann, es kostet nur 50 DM. Aber Bob antwortet: "Wie kann ich wissen, dass deine Nummer wirklich ein Faktor ist? Wenn Sie mir eine Nummer geben, dann kann ich überprüfen, ob diese Nummer ein Faktor ist, und somit die Bedingung akzeptieren."Bob weiß, dass Alice nicht zuverlässig ist. Deswegen soll Alice nicht wissen, welchen Faktor er erhalten wird. Um dieses Problem zu lösen, muss Bob Alice einen ganzen Faktor geben, aber Bob kann nur 50% Bits erhalten, und Alice kann nicht wissen, welchen Faktor Bob erhalten hat.
Wie kann man diese Anforderung erfüllen?

**2.2. Unterzeichnen von Verträgen.** Alice und Bob haben einen Vertrag $v$ ausgehandelt, und dieser soll auf elektronischem Weg unterschrieben werden. Damit kann man Zeit und Geld sparen! Der Vertrag wird für Alice bindend, wenn sie ihre Unterschrift $D_A(v)$ an den Vertrag angefügt hat. Das Entsprechende gilt für Bob, wobei $(D_A, E_A)$ und $(D_B, E_B)$ die Schlüsselpaare von Alice und Bob für das Signatursystem sind.
Dabei tritt folgendes Problem auf: Was passiert, wenn Alice unterschreibt und $(v, D_A(v))$ an Bob schickt, dieser sich aber dann weigert, seinerseits zu unterzeichnen und statt dessen darauf pocht, dass Alice ihre Verpflichtungen ihm gegenüber einhält?
Betrachte etwa die Situation, in welcher Alice für eine Ware bezahlt hat, die sie dann aber nicht erhält. Die traditionelle Lösung dieses Problems ist ein TTP (ein Notar).
Aber wie kann man ohne Notar dieses Problem lösen?

## 3. Theorie von Oblivious Transfer

Für Oblivious Transfer gibt es zwei Protokolle: $OT$ und $OT_2^1$. $OT_2^1$ ist eine Variante von Oblivious Transfer, und heißt *1-aus-2-Oblivious Transfer*.

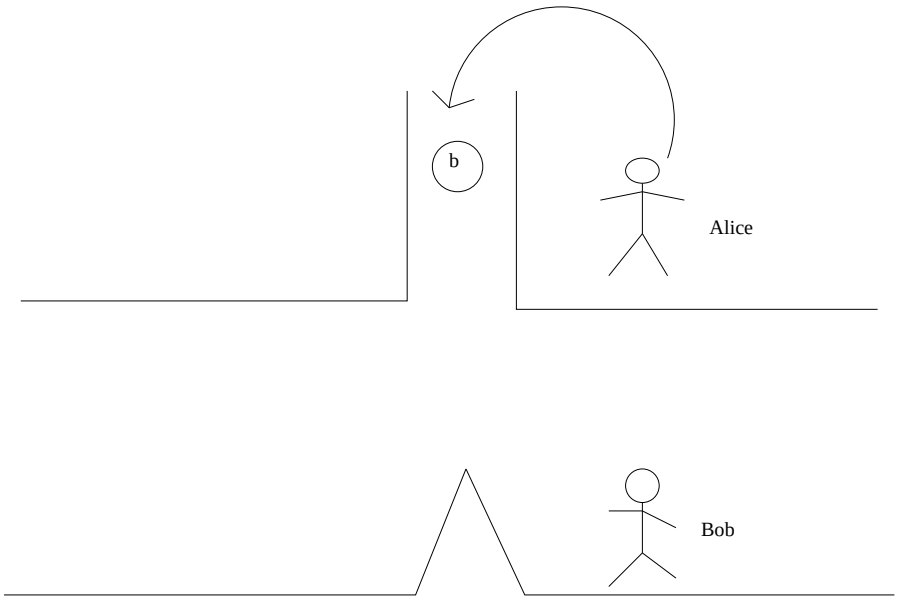



ABBILDUNG 1. Eine mechanische Realisierung von $OT$

**3.1. *Oblivious Transfer* Axioms.** Ein $OT$ ist ein Protokoll, das drei Elemente hat.

(1) Sender S, der Nachricht N senden will.
(2) Empfänger R, der Nachricht N erhalten will.
(3) Nachricht N.

Das Protokoll *OT(S,R,N)* soll folgende Axiome erfüllen:

(1) Für Empfänger R,
Nach OT Ausführung kann der Empfänger R diese Nachricht N mit Wahrscheinlichkeit von genau $\frac{1}{2}$ erhalten. Wenn R die Nachricht N nicht erhält, dann kann er auch keine 'zusätzliche Information' über N erhalten. (*'No Clue'*)
(2) Wenn S versucht, die Wahrscheinlichkeit zu ändern, mit der R die Nachricht N erhält, dann kann R dies herausfinden.

**3.2. Ein mechanisches Modell für *OT*.** In dem mechanischen Modell von Oblivious Transfer ($OT$) wirft Alice einen Ball, auf den sie die Nachricht $b$ geschrieben hat, in einen Schacht, den Alice nicht einsehen kann. Der Schacht endet ein Stockwerk tiefer; diese Etage ist von einer spitz zulaufenden Mauer in zwei Hälften geteilt, von denen nur eine zugänglich ist; in dieser befindet sich Bob. Kommt der Ball nun aus dem Schacht, so springt er jeweils mit Wahrscheinlichkeit $\frac{1}{2}$ in die zugängliche bzw. in die unzugängliche Hälfte der Etage. Bob erhält also in genau der Hälfte aller Fälle die Nachricht $b$, und Alice kann nicht sehen, in welche Hälfte der Ball gefallen ist.
Dieses Modell erfüllt die beiden folgenden Bedingungen:

(1) Bob erhält die Nachricht b genau mit Wahrscheinlichkeit $\frac{1}{2}$.
(2) Alice kann nicht wissen, ob Bob die Nachricht b erhält oder nicht.

**3.3. $OT_2^1$ Axiome.** Ein $OT_2^1$ ist ein Protokoll, der vier Elementen hat:

(1) Sender S, der Nachrichten ($N_1$, $N_2$) senden will.
(2) Empfänger R, der Nachricht erhalten will.
(3) Nachrichten $N_1$ und $N_2$.

Im Allgemein kann der Protokoll $OT_2^1(S, R, N_1, N_2)$ dieser Axiom erfüllen:

(1) Sender S hat zwei Nachrichten $N_1$ und $N_2$.
(2) Nach Protokoll $OT_2^1$ kann der Empfänger R eine Nachricht $N_\sigma$ erhalten, $\sigma \in \{0, 1\}$, aber der Empfänger R erfährt nichts über $\sigma$.

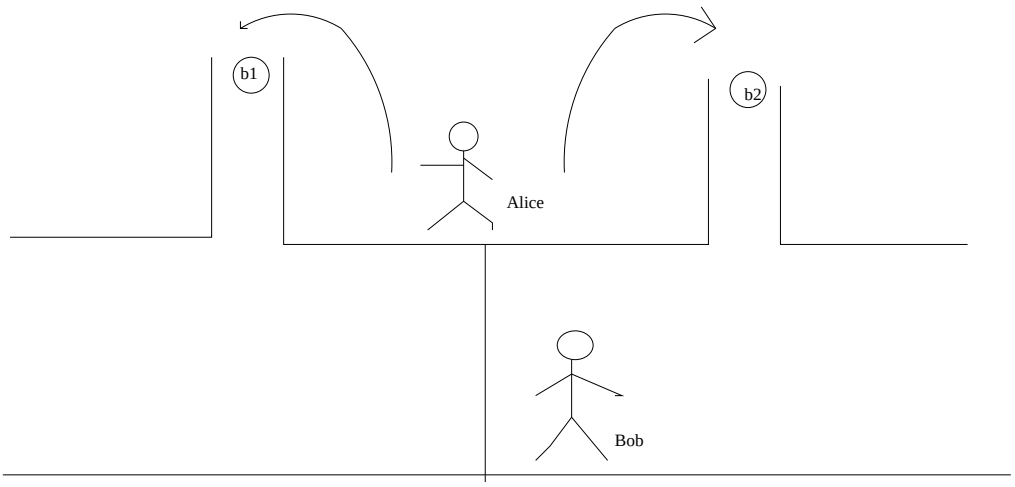



ABBILDUNG 2. Eine mechanische Realisierung von $OT_2^1$

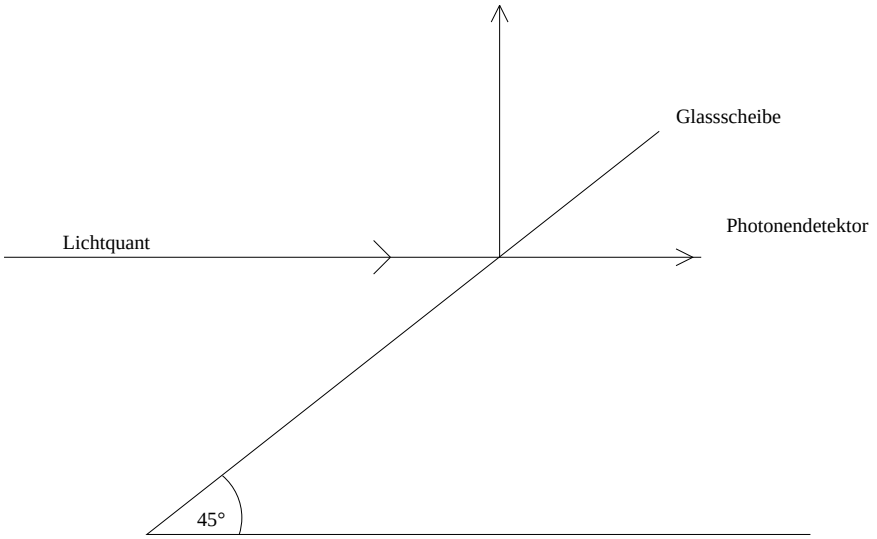



ABBILDUNG 3. Eine mechanische Realisierung von $OT_2^1$

(3) Der Empfänger R erhält Nachricht $N_{1-\sigma}$ nicht.
(4) Der Sender S weiß nicht, welche Nachricht empfangen wurde.

**3.4. Ein mechanisches Modell für $OT_2^1$.** Es gibt zwei Schächte, die Alice nicht einsehen kann. In der unteren Etage gibt es eine Mauer, d.h. das untere Stockwerk hat zwei Hälften. Bob steht in einer der Hälften. Alice schreibt die Nachricht $b_0$ auf einen Ball, und die Nachricht $b_1$ auf einen anderen Ball. Danach wirft Alice die Bälle in jeweils einen Schacht. In dieser Situation kann Bob genau eine der zwei Nachrichten erhalten, und Alice weiß nicht, welcher Nachricht Bob erhalten hat.
Man kann sich auch eine sehr effektive quantenphysikalische Implementierung verstellen, wie in Bild 3 dargestellt ist. Hierbei trifft ein einzelnes Lichtquant im Winkel von $45^o$ auf eine Glasscheibe. Es durchdringt diese genau mit Wahrscheinlichkeit $\frac{1}{2}$ reflektiert. Die als Wellenlänge codierte Information erreicht also genau in der Hälfte aller Fälle ihren Adressaten, den Photonendetektor.

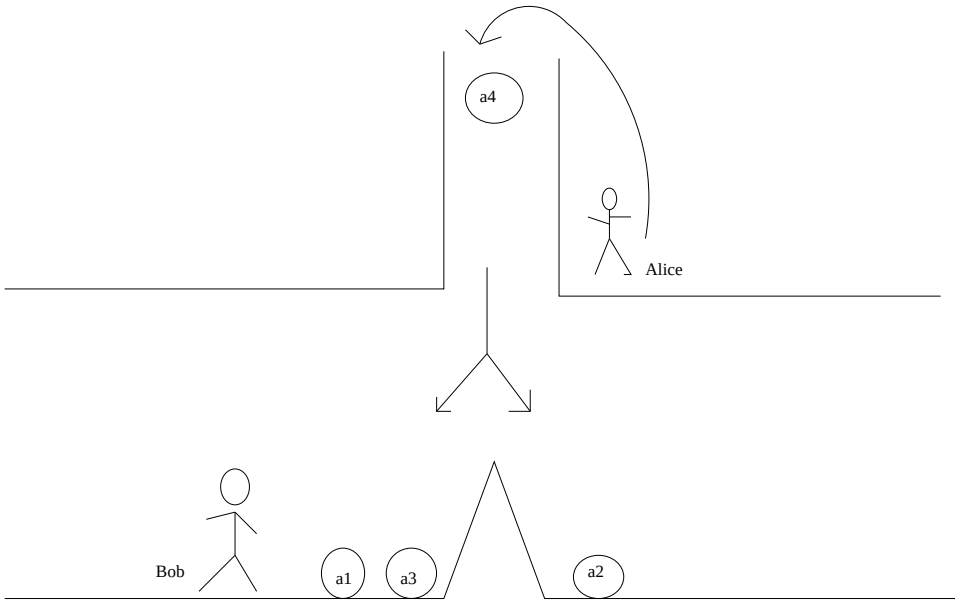



ABBILDUNG 4. Durchführung des OT-Protokolls für die Bits $a_1$ bis $a_n$

**3.5. *OT* und $OT_2^1$ sind äquivalent.** Es ist klar, dass man aus einem $OT_2^1$-Protokoll ein $OT$-Protokoll manchen kann. Alice braucht nur zufällig zu entscheiden, in welchen der beiden Schächte sie den Ball mit der Nachricht werfen soll.
Aber wie kann man mit ein $OT$-Protokoll ein $OT_2^1$-Protokoll konstruieren? Das ist nicht trivial. So sieht eine Lösung aus:

(1) Alice schreibt $n$ sinnlose Nachrichten $a_1, \cdots, a_n$ auf von 1 bis $n$ nummerierte Bälle und führt für jede dieser Nachrichten ein $OT$-Protokoll mit Bob durch. Wenn die Zahl $n$ sehr groß ist, dann soll in jede Hälfte des unteren Stockwerks mindestens $\frac{n}{3}$ und höchstens $\frac{2n}{3} - 1$ Bälle fallen.
(2) Dann erzeugt Bob zwei Listen. Auf die Liste 1 schreibt er die Nummern der $\frac{n}{3}$ Nachrichten, die er bekommen hat, und auf die Liste 0 schreibt er die $\frac{n}{3}$ anderen Nummern. Dann gibt er die zwei Listen an Alice.
(3) Alice verschlüsselt $b_0$ mit Liste 0 und $b_1$ mit Liste 1. Dann gibt sie die Ergebnisse $s_0$ und $s_1$ an Bob. Bob hat genau eine Nachricht erhalten.

**3.6. Implementierung von OT.** Alice wählt zwei große Primzahlen $p, q$ und benutzt ein zu $m = pq$ gehörendes RSA-System, um ein Geheimnis $s$ zu verschlüsseln. Dann sendet sie $m$ an Bob. Dieser berechnet einen quadratischen Rest $z = x^2 \bmod m$ und gibt $z$ zurück. Alice benutzt nun ihre Kenntnis der Faktorisierung von $m$, um eine Quadratwurzel $y$ von $z$ zu berechnen, und schickt diese an Bob. Da es zu jedem quadratischen Rest modulo $m$ genau vier Quadratwurzeln, nämlich $a$, $-a$ sowie $b$, $-b$ gibt, ist die Wahrscheinlichkeit, dass Bob eine Wurzel der Form $y = \pm x$ erhält, genau $\frac{1}{2}$. In diese Fall kann er nichts Neues berechnen. Mit Wahrscheinlichkeit $1/2$ erhält Bob aber eine Quadratwurzel, die nicht gleich $x$ oder $-x$ ist. In diesem Fall gilt

$$(x + y)(x - y) = x^2 - y^2 = 0(\bmod m). \tag{1}$$

Indem Bob mit dem Euklidischen Algorithmus $\mathrm{ggT}(x + y, m) \in p, q$ berechnet, kann er $m$ faktorisieren und das RSA-System berechnen. Bob kann also genau mit Wahrscheinlichkeit $\frac{1}{2}$ das Geheimnis erhalten.

## 4. Beispielen zu lösen mit Oblivious Transfer

Zunächst werden wir zwei Anwendungsbeispiele mit Oblivious Transfer lösen.

### 4.1. Faktorisierung.

4.1.1. *Schritte.* Mit Oblivious Transfer können wir das Faktorisierung Problem so lösen:

(1) Alice erstellt zwei öffentliche Schlüssel und zwei private Schlüssel ($O_A^1$ /$G_A^1$ und $O_A^2/G_A^2$). Sie gibt zwei öffentliche Schlüssel ($O_A^1$ und $O_A^2$) an Bob.
(2) Bob erstellt einen öffentlichen Schlüssel und einen privaten Schlüssel ($O_B/G_B$). Dann verschlüsselt er seinen privaten Schlüssel ($O_B$) mit einem öffentlichem Schlüssel von Alice ($O_A^x$). Danach gibt er dieses Ergebnis ($Z = O_A^x(O_B)$, $x \in \{0,1\}$ ) an Alice.
(3) Alice entschlüsselt dieses Ergebnis zweimal mit ihren zwei privaten Schlüsseln. Sie bekommt zwei Zahlen $\{O_B, O_B^{falsch}\}$, eine ist Bob's öffentlicher Schlüssel ($O_B$), die andere ist eine 'sinnlose' Zahl. Alice weiß nicht, welche die richtige Zahl ist.
(4) Alice teilt einen Faktor (Nachricht $b$) in zwei 'Halbfaktoren' (Zahlen $b_0$ und $b_1$) und verschlüsselt diese mit $\{O_B, O_B^{falsch}\}$. Dann gibt sie die zwei verschlüsselten Zahlen $O_B(b_x)$ und $O_B^{falsch}(b_{1-x})$, $x \in \{0,1\}$ an Bob.
(5) Bob entschlüsselt diese zwei verschlüsselten Zahlen mit seinem privaten Schlüssel. Er erhält zwei Zahlen, eine ist der richtige Halbfaktor, die andere ist 'falsch'.
(6) Alice gibt ihre privaten Schlüssel an Bob, so dass Bob diesen Halbfaktor überprüfen kann. Er kann beweisen, dass Alice nicht weiß, welchen Halbfaktor er erhalten hat.

Jetzt hat Bob genau einen Faktor, und Alice weiß nicht, welchen Faktor Bob hat.
Diese Protokoll garantiert, dass Alice zwei Nachrichten sendet, aber Bob kann nur genau eine Nachricht erhalten. Natürlich kann Alice auch sinnlose Nachrichten senden. Um 'sinnvolle' Nachrichten zu garantieren, braucht man noch einen Zero-Knowledge Beweis (siehe [**BSW98**]). Zero-Knowledge Beweise werde in dieser Arbeit nicht behandelt.

4.1.2. *Was passiert, wenn Alice lügt?* Angenommen Alice möchte lügen. Sie kann in Schritt 4 dieselbe Nachricht mit zwei Schlüsseln verschlüsseln. In dieser Situation wird nach dem letzten Schritt Bob zwei private Schlüssel von Alice erhalten. Deswegen kann er die Schritte 1 bis 3 wiederholen. Wenn Alice lügt, dann kann Bob seine Berechnungsergebnisse mit den zwei Zahlen vergleichen, die er im Schritt 4 erhält. Somit kann Bob diese Lüge entdecken.

### 4.2. Vertragsunterzeichnung mit $OT$.

4.2.1. *Schritten.*

(1) *Vorbereitung:*
Alice und Bob einigen sich auf ein Verfahren, den Vertrag $V$ insgesamt $n$-mal auf verschiedene Arten zu halbieren. Also die beiden Hälften der ersten Halbierung sind $V(1)$ und $V(n+1)$, die zweiten sind $V(2)$ und $V(n+2)$ usw.. Danach unterschreiben Alice und Bob die beiden Vertragshälften.

$$a_i := D_A(V(i)) und b_i := D_B(V(i))(i = 1, 2, \cdots, 2n)$$

Der Vertrag wird für Alice bindend, wenn Bob die Unterschriften $a_i$ und $a_{n+i}$ von zueinander passender Hälften des Vertrages $V$ präsentieren kann. Analog für Bob.
(2) *1-aus-2-Oblivious Transfer:*
Alice gibt mit $OT_2^1$ jeweils genau eine der beiden Hälften von

$$(a_1, a_{n+1}), \cdots, (a_n, a_{2n})$$

an Bob. Sie weiß nicht, ob Bob sich bei dem Paar $(a_i, a_{n+i})$ für $a_i$ oder für $a_{n+i}$ entschieden hat. Bob seinerseits kann überprüfen, ob er jeweils unterschriebene Hälften des Vertrags erhalten hat, indem er die Unterschrift $a_j$ mit Hilfe des öffentlichen Schlüssels $E_A$ verifiziert. Auf die gleiche Art gibt Bob seine Hälfte seiner Unterschriften an Alice.
(3) *Bitweise Übertragung aller Unterschriften:*
Alice und Bob senden sich nun Bit für Bit abwechselnd ihre Unterschriften zu. z.B., Alice beginnt mit dem ersten Bit von $a_1$, dann antwortet Bob mit dem ersten Bit von $b_1$, usw. Am Ende dieses Schrittes kennen beide Parteien alle Teilunterschriften ihres Partners, können also insbesondere ein Paar von Unterschriften präsentieren. Damit ist der Vertrag gültig.

4.2.2. *Warum ist diese Lösung sicher?* Angenommen dass Bob Alice betrügen wollte. Also Bob möchte ein Paar$(a_i,a_{n+i})$ erhalten, aber ohne seinerseits ein Paar$(b_j,b_{n+j})$ an Alice weiterzugeben. Dazu fallen ihm folgende Möglichkeiten ein:

(1) Er könnte in Schritt 2 die $b_j$ fälschen, die er Alice anbietet, das heißt, er bietet ein $b'_j$ an, das keine Signatur von $v(j)$ ist. Da Alice diese Bedingung überprüft, kann Bob jeweils höchstens eine der beiden Teilunterschriften $(b_j, b_{n+j})$ verändern mit der Hoffnung, dass Alice in Schritt 2 die unveränderte Unterschrift auswählt. Um andererseits auch im Schritt 3 betrügen zu können, muss Bob in jedem Paar mindestens eine Hälfte fälschen, denn Alice vergleicht die erhaltenen Bits (falls möglich) mit den in Schritt 2 erhaltenen Teilgeheimnissen. Die Wahrscheinlichkeit, dass Alice in allen Fällen die korrekte Hälfte von $(b_j, b_{n+j})$, $j \in \{1, 2, \cdots, n\}$, wählt, ist verschwindend klein, nämlich $(\frac{1}{2})^n$. Diese Wahrscheinlichkeit wird durch den Sicherheitsparameter n gesteuert. Alice erkennt diesen Betrugsversuch von Bob also praktisch immer.

(2) Bob könnte in Schritt 3 falsche Bits übertragen. Dazu muss er raten, welche Hälfte von $(b_j, b_{n+j})$ Alice in Schritt 2 gewählt hat. War dies zum Beispiel $b_j$, so muss er die Bits von $b_j$ korrekt übertragen und kann die Bits von $b_{n+j}$ verändern. Die Wahrscheinlichkeit, dass Bob für alle n Paare richtig rät, ist mit $(\frac{1}{2})_n$ verschwindend klein. Alice wird diese Betrugsversuche entdecken.

(3) Da Bob in Schritt 3 das letzte Bit von $a_1$ erhält, bevor er das letzte Bit von $b_1$ senden muss, könnte er in Schritt 2 den Wert $a_{n+1}$ auswählen, um dann in Schritt 3 keine weiteren Nachrichten mehr an Alice zu senden, nachdem er das letzte Bit von $a_1$ erhalten hat. In diesem Fall besitzt Bob $(a_1, a_{n+1})$, und Alice ist an den Vertrag gebunden. Aber Alice kann hier leicht ein Paar berechnen: Mit höher Wahrscheinlichkeit hat Alice in Schritt 2 eine Hälfte $b_{n+j}$ für ein $j \in \{1, 2, \cdots, n\}$ gewählt. In diesem Fall kennt Alice das Paar $(b_j, b_{n+j})$ bis auf das letzte Bit von $b_j$, das sie einfach durch Ausprobieren der beiden Möglichkeiten 0 und 1 bestimmen kann.

Eine ähnliche Argumentation kann man verwenden, wenn Bob die Übertragung schon früher einstellt. Alice benötigt höchsten doppelt soviel Zeit wie Bob, um ein Paar zu vervollständigen.

## 5. Zusammenfassung und Ausblick

*Oblivious Transfer* ist ziemlich sinnvoll für "faire "Übertragungen ohne Notar. Aber alle bekannten Implementierungen sind zu langsam, als dass ein praktischer Einsatz heute schon möglich wäre.

KAPITEL 9

# Kryptanalyse des Merkle-Hellman-Rucksacksystems

**Klaus Dräger**

## 1. Einleitung

Das Merkle-Hellman-Rucksacksystem gehörte zusammen mit RSA zu den ersten Kryptosystemen mit einem öffentlichen Schlüssel. Mittlerweile sind alle bekannten Rucksacksysteme gebrochen; die Analyse des Verfahrens von Merkle und Hellman bildete dabei den Ausgangspunkt und wird hier beschrieben.

## 2. Einführung

Nachdem 1976 Diffie und Hellman das Konzept der asymmetrischen Verschlüsselung eingeführt hatten, wurden 1978 die ersten Public-Key-Kryptosysteme veröffentlicht: RSA von Rabin, Shamir und Adleman sowie das System von Merkle und Hellman, das auf der Schwierigkeit des Subset-Sum-Problems basiert. Nachdem in den folgenden Jahren mehrere Schwächen des Systems bekannt wurden, brachen Shamir (1981) und Brickell (1984) das einfach bzw. mehrfach iterierte Merkle-Hellman-Kryptosystem. Alle anderen bekannten Kryptosysteme, die auf dem Rucksackproblem aufbauen, sind mittlerweile ebenfalls kryptanalysiert worden, zuletzt das 1984 entwickelte Chor-Rivest-Rucksackproblem, das 1998 von Vaudenay gebrochen wurde.
In Kapitel 2 werden zunächst die Grundlagen, auf denen das System von Merkle und Hellman basiert, eingeführt. Das einfach iterierte System wird im dritten Kapitel beschrieben, gefolgt vom Ansatz der Kryptanalyse. In den Kapiteln 5 und 6 werden einige benötigte Verfahren eingeführt, nämlich der LLL-Algorithmus zur Bestimmung reduzierter Gitterbasen und die Verwendung desselben für die Ermittlung guter simultaner diophantischer Approximationen. Das siebte Kapitel schildert darauf aufbauend den Ablauf der Kryptanalyse des einfachen Systems und untersucht ihre Zeitkomplexität, gefolgt von der Behandlung des permutierten und mehrfach iterierten Systems.
Die folgende Ausarbeitung basiert haupsächlich auf [**Kux99**], [**Bri83**] und [**Bri84**].

## 3. Grundlagen

Das Merkle-Hellman-Kryptosystem basiert auf dem Subset-Sum-Problem, einem Spezialfall des Rucksackproblems:

**Definition:** Rucksackproblem

Gegeben:

Gewichte $w_1, \ldots, w_n \in \mathbb{N}$
Werte $s_1, \ldots, s_n \in \mathbb{N}$
Gewichtsschranke $W \in \mathbb{N}$
Zielwert $S \in \mathbb{N}$

Gesucht:

Teilmenge $I \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in I} w_i \leq W$ und $\sum_{i \in I} s_i \geq S$
bzw. Binärvektor $x \in \{0,1\}^n$ mit $\sum_{i=1}^{n} x_i w_i \leq W$ und $\sum_{i=1}^{n} x_i s_i \geq S$

Wählt man die Parameter so, dass $W = S$ und $w_i = s_i$ für alle $i$ gilt, so erhält man das Subset-Sum-Problem:

**Definition:** Subset-Sum-Problem

Gegeben:
Gewichte $s_1, \ldots, s_n \in \mathbb{N}$
Zielwert $S \in \mathbb{N}$
Gesucht:
Teilmenge $I \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in I} s_i = S$
bzw. Binärvektor $x \in \{0,1\}^n$ mit $\sum_{i=1}^{n} x_i s_i = S$

Genau wie das Rucksackproblem ist das Subset-Sum-Problem NP-schwer; es gibt aber einfache Instanzen, d.h. Folgen von Gewichten, mit denen das Subset-Sum-Problem für jeden Zielwert in Polynomialzeit lösbar ist. Ein Beispiel ist $s_i = 2^i$; in diesem Fall sind die $x_i$ gerade die Binärdarstellung von $S$ (vorausgesetzt, $S$ ist klein genug). Eine Verallgemeinerung dieses Beispiels stellen die *stark wachsenden Folgen* dar:

Eine Folge $\boldsymbol{s} = (s_1, \ldots, s_n)$ heißt **stark wachsend**, wenn $s_i > \sum_{j=1}^{i-1} s_j$ für $i = 2, \ldots, n$.

Ein Subset-Sum-Problem mit einer stark wachsenden Folge $s_1, \ldots, s_n$ von Gewichten lässt sich durch folgenden Greedy-Algorithmus in Polynomialzeit lösen:

Für $i = n$ bis 1:
Falls $s_i \leq S$:
$x_i := 1$
$S := S - s_i$
Sonst:
$x_i := 0$
Ausgabe: $x_1, \ldots, x_n$, falls $S = 0$, sonst „Keine Lösung“

## 4. Das einfache Merkle-Hellman-Rucksacksystem

Das Merkle-Hellman-Kryptosystem verwendet als Geheimschlüssel eine stark wachsende Folge $s_1, \ldots, s_n$, einen Modulus $M$ mit $M > \sum_{i=0}^{n} s_i$ sowie einen zu $M$ teilerfremden Multiplikator $W$ bzw. dessen Inverses $U$ modulo $M$. Aus diesen ermittelt der Benutzer den öffentlichen Schlüssel in Form der Folge $a_1, \ldots, a_n$ mit $a_i = W s_i \bmod M$.
Der Klartextraum besteht aus allen Binärvektoren der Länge $n$; der zum Klartext $x = (x_1, \ldots, x_n)$ gehörige Geheimtext ist dann $S^* = \sum_{i=1}^{n} x_i a_i$. Um den Text zu entschlüsseln, berechnet der Empfänger zunächst $S = US^* \bmod M$ und wendet dann den Greedy-Algorithmus auf $S$ und die $s_i$ an. Der Binärvektor, den der Empfänger dadurch erhält, ist die ursprüngliche Nachricht:
Es gilt $a_i = W s_i \bmod M \;\Rightarrow\; s_i = U a_i \bmod M$ und damit für $S^* = \sum_{i=0}^{n} x_i a_i$:

$$\begin{aligned} S &= US^* \bmod M \\ &= U \sum_{i=0}^{n} x_i a_i \bmod M \\ &= \sum_{i=0}^{n} x_i U a_i \bmod M \\ &= \sum_{i=0}^{n} x_i s_i. \end{aligned}$$

Die Lösung, die der Algorithmus liefert, ist also $x_1, \ldots, x_n$.

**Beispiel:** $n = 6$; $s_i = 3^i$; $M = 1100$, $W = 101 \;\Rightarrow\; U = 501$

$s_1 = 3$; $a_1 = 303$
$s_2 = 9$; $a_2 = 909$
$s_3 = 27$; $a_3 = 527$
$s_4 = 81$; $a_4 = 481$
$s_5 = 243$; $a_5 = 343$
$s_6 = 729$; $a_6 = 1029$

Sei $x = 101011$.

Verschlüsselung: $S^* = 303 + 527 + 343 + 1029 = 2202$.

Entschlüsselung: $S = 2202 \cdot 501 \bmod 1100 = 1002$.

Greedy-Algorithmus:

$1002 \geq 729 \Rightarrow x_6 = 1;\ S = 273$
$273 \geq 243 \Rightarrow x_5 = 1;\ S = 30$
$30 < 81 \Rightarrow x_4 = 0$
$30 \geq 27 \Rightarrow x_3 = 1;\ S = 3$
$3 < 9 \Rightarrow x_2 = 0$
$3 \geq 3 \Rightarrow x_1 = 1;\ S = 0$

Ausgabe also $101011 = x$.

Die Hoffnung auf Sicherheit des Kryptosystems basiert auf der Tatsache, dass die $a_i$ im Gegensatz zu den $s_i$ nicht stark wachsend sind, also keine einfache Instanz von Subset-Sum bilden; sie sollten für eine Entschlüsselung in annehmbarer Zeit also unbrauchbar sein. Dies hat sich als Trugschluss herausgestellt.

## 5. Ansatz der Kryptanalyse

Wenn man sich das Verfahren genauer betrachtet, bemerkt man, dass zur Entschlüsselung der Nachricht der Geheimschlüssel nicht unbedingt nötig ist; es genügt, eine beliebige stark wachsende Folge $s_1^*, \ldots, s_n^*$ der Form $s_i^* = U^* a_i \bmod M^*$ zu finden, die dann ebenfalls zur Dekodierung verwendet werden kann. Eine solche Folge kann in Polynomialzeit aus den $a_i$ ermittelt werden.
Da für alle $i \in \{1, \ldots, n\}$ $s_i = U a_i \bmod M$ gilt, gibt es $k_1, \ldots, k_n \in \mathbb{N}$ mit

$$s_i = U a_i - M k_i. \tag{2}$$

Für diese $k_i$ gilt:

**Lemma:** Es gibt ein $\varepsilon > 0$, so dass für alle $U^*, M^*$ mit $\left|\frac{U^*}{M^*} - \frac{U}{M}\right| < \varepsilon$ gilt: $s_i^* = U^* a_i - M^* k_i$ ist stark wachsend. Es gibt also insbesondere unendlich viele solche **Falltürpaare** $(U^*, M^*)$.

Da außerdem wegen (2) gilt $0 < \left|\frac{U}{M} - \frac{k_i}{a_i}\right| = \frac{s_i}{M a_i} < \frac{1}{a_i}$, liegt $\frac{U}{M}$ für alle $i$ in $I_i = \left[\frac{k_i}{a_i}, \frac{k_i+1}{a_i}\right]$; wir können uns also insbesondere bei der Suche nach Falltürpaaren auf $I_1$ beschränken.
Im Folgenden werden die Verfahren vorgestellt, mit deren Hilfe (Kandidaten für) $k_1$ bestimmt werden können, sowie die Verwendung von $k_1$ in der Kryptanalyse.

## 6. Gitter im $\mathbb{R}^m$; der LLL-Algorithmus

**Definition:** Gitter.

Seien $v_1, \ldots, v_n \in \mathbb{R}^m$ linear unabhängig. Dann ist $L = \{\sum_{i=1}^n z_i b_i \mid z_i \in \mathbb{Z}\}$ ein **Gitter** der Dimension $n$ im $\mathbb{R}^m$; die $v_i$ heißen **Gitterbasis** von $L$.

Für viele Anwendungen ist es interessant, eine *reduzierte Gitterbasis* zu einem gegebenen Gitter $L$ zu finden. Diese zeichnet sich durch relativ kurze und „fast orthogonale“ Vektoren aus; was darunter genau zu verstehen ist, wird durch bestimmte Eigenschaften der Ergebnisse des Gram-Schmidtschen Orthogonalverfahren bei Anwendung auf eine solche Basis definiert:

**Definition:** Gram-Schmidtsches Orthogonalisierungsverfahren.

Eingabe: linear unabhängige Vektoren $b_1, \ldots, b_n$
Definiere $\mu_{ij},\ 1 \leq j < i \leq n$, sowie $b_1^*, \ldots, b_n^*$ rekursiv durch
$b_1^* := b_1$;
$b_i^* := b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^*, \quad \mu_{ij} := \frac{(b_i, b_j^*)}{(b_j^*, b_j^*)}$

Die Vektoren $b_1^*, \ldots, b_n^*$ bilden dann eine orthogonale Basis des von $b_1, \ldots, b_n$ aufgespannten Vektorraums; wenn $b_1, \ldots, b_n$ eine Gitterbasis von $L$ ist, liegen sie aber in der Regel nicht selbst in $L$.
Die Basis $b_1, \ldots, b_n$ heißt nun **reduziert**, wenn die $\mu_{ij}$ und $b_i^*$ folgende Bedingungen erfüllen:

- $|\mu_{ij}| \leq \frac{1}{2}$
- $\|b_i^* + \mu_{i,i-1} b_{i-1}^*\|^2 \geq \frac{3}{4}\|b_{i-1}^*\|^2$

Für die Vektoren einer solchen Basis gilt dann:

- $\|b_1\| \leq 2^{\frac{n-1}{2}}\|x\|$ für alle $x \in L \setminus \{0\}$
- $\|b_j\| \leq 2^{\frac{n-1}{2}} max\{\|x_1\|, \ldots, \|x_t\|\}$ für alle linear unabhängigen $\{x_1, \ldots, x_t\} \in L$,

d. h. die $b_i$ sind nicht erheblich länger als der kürzeste Vektor des Gitters. In der Regel sind diese Abschätzungen sogar relativ pessimistisch; oft gehören die Vektoren einer reduzierten Basis zu den kürzesten Elementen eines Gitters.
Der von Lenstra, Lenstra und Lovász veröffentlichte LLL-Algorithmus berechnet zu einer vorgegebenen Gitterbasis $v_1, \ldots, v_n$ in Polynomialzeit eine reduzierte Basis.

**Definition:** LLL-Algorithmus

(1) Initialisierung:
$b_i := v_i$ für $i = 1, \ldots, n$
Berechnung der $\mu_{ij}$, $b_i^*$ wie im Algorithmus von Gram-Schmidt
$B_i := \|b_i^*\|^2$
$k := 2$.

(2) Hauptschleife:
2.1 RED($k$,$k-1$) (um Bedingung 1 für $k$, $k-1$ zu gewährleisten)
2.2 Falls $B_i + \mu_{i,i-1} B_{i-1} \geq \frac{3}{4} B_{i-1}$ (d. h., falls Bedingung 2 für $k$ nicht gilt):
Vertausche $b_i$ und $b_{i-1}$ und passe die entsprechenden Werte an
Falls $k > 2$, setze $k := k - 1$ (Bedingung 1 gilt für $k$, $k-1$ unter Umständen nicht mehr)
Sonst:
Für $l = k-2, \ldots, 1$: RED($k$,$l$)
$k := k + 1$
2.3 Falls $k \leq n$: zurück zu 2.1.
Sonst: $b_1, \ldots, b_n$ ausgeben

RED($k$,$l$):
Falls $|\mu_{kl}| > \frac{1}{2}$:
$r := \lfloor \mu_{kl} + \frac{1}{2} \rfloor$ (die $\mu_{kl}$ nächstgelegene ganze Zahl)
$b_k := b_k - r b_l$; $\mu_{kj}$ für $j = 1, \ldots, l$ entsprechend anpassen.

Die $b_i$, die letztendlich ausgegeben werden, bilden eine reduzierte Gitterbasis für $L$.

**Beispiel:**

$$b_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},\ b_2 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix},\ b_3 = \begin{pmatrix} 1 \\ 4 \\ 9 \end{pmatrix}$$

Änderungen an den $b_i$:

(1) $b_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},\ b_2 = \begin{pmatrix} \mathbf{-1} \\ \mathbf{0} \\ \mathbf{1} \end{pmatrix},\ b_3 = \begin{pmatrix} 1 \\ 4 \\ 9 \end{pmatrix}$

(2) $b_1 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix},\ b_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},\ b_3 = \begin{pmatrix} 1 \\ 4 \\ 9 \end{pmatrix}$

(3) $b_1 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix},\ b_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},\ b_3 = \begin{pmatrix} \mathbf{-4} \\ \mathbf{-1} \\ \mathbf{4} \end{pmatrix}$

(4) $b_1 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix},\ b_2 = \begin{pmatrix} -4 \\ -1 \\ 4 \end{pmatrix},\ b_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$

(5) $b_1 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$, $b_2 = \begin{pmatrix} \mathbf{0} \\ \mathbf{-1} \\ \mathbf{0} \end{pmatrix}$, $b_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$

(6) $b_1 = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}$, $b_2 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$, $b_3 = \begin{pmatrix} \mathbf{1} \\ \mathbf{0} \\ \mathbf{1} \end{pmatrix}$ ist eine reduzierte Basis.

**Bemerkung zur Laufzeit:** Im schlimmsten Fall benötigt der Algorithmus $O(n^4 \log_2(z))$ arithmetische Operationen (und bei Verwendung der klassischen arithmetischen Verfahren $O(n^6 \log_2(z)^3)$ Bitoperationen) mit $z = \log_2(\max\{\|b_i\|^2\})$; in der Praxis genügen meist sogar $O(n \log_2(z)^3)$ Bitoperationen.

## 7. Simultane Diophantische Approximationen

**Definition:** $\Delta$-gute Näherungen

Seien $v = (\frac{v_2}{v_1}, \ldots, \frac{v_n}{v_1})$, $u = (\frac{u_2}{u_1}, \ldots, \frac{u_n}{u_1})$; $v_i, u_i \in \mathbb{N}$. $u$ ist eine **$\Delta$-gute simultane diophantische Approximation** ($\Delta$-SDA) an $v$, wenn $0 < u_1 < v_1$ und $|v_i u_1 - v_1 u_i| \leq \Delta$ für $i = 1, \ldots, n$. Ist $\Delta < v_1^{1-\frac{1}{n}}$, so heißt die Näherung $u$ **ungewöhnlich gut**. Solche Näherungen sind selten: Für die Wahrscheinlichkeit $p_k$, dass zu einem gegebenen Vektor $v = (v_1, \ldots, v_n)$ mindestens $k$ ungewöhnlich gute simultane diophantische Approximationen (UGSDA) existieren (unter der Bedingung, dass mindestens eine existiert), gilt $p_k \leq \frac{c_n}{k^2}$ mit von $n$ abhängigen Konstanten $c_n$.

Die $k_i$ sind nun Lösung des Ungleichungssystems:

- $|a_i x_1 - a_1 x_i| \leq 2^{-n+l} a_1$, $2 \leq i \leq l$
- $x_1, \ldots, x_n \in \mathbb{Z}$, $0 \leq x_1 < a_1$

$\Rightarrow$ für alle $l$ mit $(l-1)(n-l) > \log_2(a_1)$ ist $(\frac{k_2}{k_1}, \ldots, \frac{k_l}{k_1})$ eine UGSDA an $(\frac{a_2}{a_1}, \ldots, \frac{a_l}{a_1})$.

Der LLL-Algorithmus liefert eine Möglichkeit, $\Delta$-SDA und damit UGSDA zu einem gebenenen Vektor $a = (\frac{a_2}{a_1}, \ldots, \frac{a_n}{a_1})$ in Polynomialzeit zu ermitteln:

Sei $b_1 = \begin{pmatrix} a_2 \\ a_3 \\ \vdots \\ a_n \\ \varepsilon \end{pmatrix}$ sowie $b_2 = \begin{pmatrix} -a_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ldots, b_n = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -a_1 \\ 0 \end{pmatrix}$, wobei $\varepsilon < \frac{\Delta}{a_1}$.

$b = (b_1, \ldots, b_n)$ ist Basis eines Gitters $L_\varepsilon$, und für die Elemente $v = \sum_{i=1}^{n} h_i b_i$, $h_i \in \mathbb{Z}$, gilt:

- $\|v\| \leq \Delta \Rightarrow (\frac{h_2}{h_1}, \ldots, \frac{h_l}{h_1})$ $\Delta$-SDA an $(\frac{a_2}{a_1}, \ldots, \frac{a_l}{a_1})$
- $(\frac{h_2}{h_1}, \ldots, \frac{h_l}{h_1})$ $\Delta$-SDA an $(\frac{a_2}{a_1}, \ldots, \frac{a_l}{a_1}) \Rightarrow \|v\| \leq \Delta\sqrt{l}$

Man kann also gute Näherungen an $(\frac{a_2}{a_1}, \ldots, \frac{a_n}{a_1})$ und damit mit hoher Wahrscheinlichkeit $k_1, \ldots, k_l$ finden, indem man in den $L_\varepsilon$ mit mehreren Werten $\varepsilon < a_1^{-\frac{1}{n}}$ nach kurzen Vektoren sucht; genau das leistet aber der LLL-Algorithmus.

## 8. Kryptanalyse des einfachen Merkle-Hellman-Rucksacksystems

Wir gehen im Folgenden davon aus, dass wir (Kandidaten für) $k_1, \ldots, k_l$ vorliegen haben.
Um nun ein Falltürpaar $(U^*, M^*)$ zu konstruieren, definiert man sich zunächst die modularen Funktionen $g_i : [0,1) \to [0,1), V \mapsto a_i V \bmod 1 = a_i V - \lfloor a_i V \rfloor$. Aus der Definition folgen sofort folgende Eigenschaften:

- $g_i$ hat im Intervall [0,1) $a_i$ Nullstellen, und zwar bei $V = \frac{k}{a_i}$ für $k = 0, \ldots, a_i - 1$
- Zwischen den Nullstellen steigt $g_i$ linear mit Steigung $a_i$
- $g_i(\frac{U}{M}) = \frac{s_i}{M}$, denn $\frac{s_i}{M} \in [0,1)$ und $\frac{s_i}{M} = \frac{a_i U}{M} - k_i$.

Ein Wert $V = \frac{U^*}{M^*}$ ist nun ein **Falltürpaar** genau dann, wenn er folgende Bedingungen erfüllt:

(i) $\sum_{i=1}^{n} g_n(V) < 1$ — dies entspricht der Bedingung, dass $\sum_{i=1}^{n} < M$.
(ii) $g_1(V), \ldots, g_n(V)$ ist stark wachsend — dies entspricht der Bedingung, dass die $s_i^*$ stark wachsend sein sollen, denn mit $s_i^* = U^* a_i \bmod M^*$ gilt

$$g_i\left(\frac{U^*}{M^*}\right) = a_i \frac{U^*}{M^*} \bmod 1 = \frac{a_i U^* \bmod M^*}{M^*} = \frac{s - i^*}{M^*}$$

Um Werte mit diesen Eigenschaften zu finden, zerlegen wir $I_1$ zunächst in Teilintervalle:
Seien $V_1, \ldots, V_s$ alle (der Größe nach geordneten) Nullstellen der $g_i$ in $I_1$, und seien die Teilintervalle $I_1^1, \ldots, I_1^{s-1}$ definiert durch $I_1^j = [V_j, V_{j+1})$. Dann sind die $g_i$ auf allen Teilintervallen linear und haben die Form $g_i|_{I_1^j}(V) = a_i V - \tau_i^j$ mit $\tau_i^j \in \mathbb{N}$.
In $I_1^j$ reduzieren sich die Bedingungen (i) und (ii) zu:

$$\begin{aligned}
(i) \quad \sum_{i=1}^{n} s_i^* < M^* \quad &\Leftrightarrow \quad \sum_{i=1}^{n} \frac{s_i^*}{M^*} < 1 \\
&\Leftrightarrow \quad \sum_{i=1}^{n} g_i(V) < 1 \\
&\Leftrightarrow \quad \sum_{i=1}^{n} (a_i V - \tau_i^j) < 1 \\
&\Leftrightarrow \quad \sum_{i=1}^{n} a_i V < 1 + \sum_{i=1}^{n} \tau_i^j \\
&\Leftrightarrow \quad V < \frac{1 + \sum_{i=1}^{n} \tau_i^j}{\sum_{i=1}^{n} a_i}
\end{aligned}$$

$$\begin{aligned}
(ii) \quad s_i^* > \sum_{k=1}^{i-1} s_k^* \quad &\Leftrightarrow \quad \frac{s_i^*}{M^*} > \sum_{k=1}^{i-1} \frac{s_k^*}{M^*} \\
&\Leftrightarrow \quad g_i(V) > \sum_{k=1}^{i-1} g_k(V) \\
&\Leftrightarrow \quad a_i V - \tau_i^j > \sum_{k=1}^{i-1} a_k V - \tau_k^j \\
&\Leftrightarrow \quad V(a_i - \sum_{k=1}^{i-1} a_k) > \tau_i^j - \sum_{k=1}^{i-1} \tau_k^j \\
&\Leftrightarrow \quad V > \frac{\tau_i^j - \sum_{k=1}^{i-1} \tau_k^j}{a_i - \sum_{k=1}^{i-1} a_k}, \text{ falls } a_i > \sum_{k=1}^{i-1} a_k \\
\text{bzw.} \quad V < \frac{\tau_i^j - \sum_{k=1}^{i-1} \tau_k^j}{a_i - \sum_{k=1}^{i-1} a_k}, &\text{ falls } a_i < \sum_{k=1}^{i-1} a_k
\end{aligned}$$

Wenn in einem der $I_1^j$ ein V gefunden werden kann, das diese Bedingungen erfüllt, haben wir damit ein Falltürpaar. Wir gehen bei der Suche folgendermaßen vor:
Für $i = 1, \ldots, s$:

Setze $a = V_i, a = V_{i+1}$ (Die Grenzen des Teilintervalls, das noch Falltürpaare enthalten kann)
Teste Bedingung (i) für $V = a$ (Wenn sie überhaupt für ein $V \in I_1^i$ gilt, dann für $a$).
Falls sie nicht erfüllt ist: nächstes $i$
Setze $b = \min(b, \frac{1+\sum_{i=1}^{n} \tau_i^j}{\sum_{i=1}^{n} a_i})$.
Für $j = 1, \ldots, n$:

Falls $a_j > \sum_{k=1}^{j-1} a_k$:
Teste Bedingung (ii) für $j$ und $V = b$.
Falls sie nicht erfüllt ist: nächstes $i$
Setze $a = \max(a, \frac{\tau_i^j - \sum_{k=1}^{i-1} \tau_k^j}{a_i - \sum_{k=1}^{i-1} a_k})$
Sonst:
Teste Bedingung (ii) für $j$und $V = a$.
Falls sie nicht erfüllt ist: nächstes $i$
Setze $b = \min(b, \frac{\tau_i^j - \sum_{k=1}^{i-1} \tau_k^j}{a_i - \sum_{k=1}^{i-1} a_k})$

Gib ein rationales Element aus $[a, b)$ aus. (Das ist dann $V = \frac{U^*}{M^*}$ mit $(U^*, M^*)$ Falltürpaar.)

**Bemerkung zur Laufzeit:** Alle Schritte des Algorithmus haben polynomiale Laufzeit. Abgesehen von Hilfsrechnungen wie der Erstellung der verschiedenen Matrizen und Vektoren sind dies im Wesentlichen:

- Die Berechnung einer reduzierten Gitterbasis mit dem LLL-Algorithmus. Die Laufzeit dieses Algorithmus wurde schon in Kapitel 5 behandelt.
- Die Bestimmung guter Näherungen an $(a_1, \ldots, a_l)$ und damit von $k_1$ verwendet eine kleine Anzahl von Iterationen (in der Praxis bis zu 20) des LLL-Algorithmus und hat damit auch polynomiale Laufzeit.
- Für die Bestimmung der $I_1^j$ und die Überprüfung der Bedingungen (i) und (ii) sind insgesamt mit $O(n^2)$ arithmetischen Operationen möglich.

## 9. Variationen des Systems

Es gibt mehrere Möglichkeiten, die Kryptanalyse des einfachen Merkle-Hellman-Systems zu erschweren; es hat sich allerdings herausgestellt, dass auch diese komplizierteren Systeme unsicher sind. Die beiden wichtigsten Erweiterungen sind die Permutation des öffentlichen Schlüssels und die Iteration der modularen Multiplikation.

**Permutation des öffentlichen Schlüssels:**
Im permutierten System ist der öffentliche Schlüssel anstatt $a_1, \ldots, a_n$ $a_{\pi(1)}, \ldots, a_{\pi(n)}$ für eine vorher bestimmte Permutation $\pi$ von $\{1, \ldots, n\}$. Bei der Entschlüsselung muss dann außer den $s_i$ noch $\pi^{-1}$ verwendet werden.
Die Analyse dieses Systems verläuft größtenteils analog zu der des Basissystems, mit folgenden Änderungen:

- Um eines der ersten $l$ $k_i$ zu ermitteln (wir benötigen nicht unbedingt $k_1$), müssen wir maximal $\binom{n}{l}$ Permutationen durchprobieren.
- Bei der Unterteilung unseres Intervalls nehmen wir zusätzlich zu den Nullstellen auch die Schnittpunkte der $g_i$ auf; da $g_1(V), \ldots, g_n(V)$ stark wachsend sein soll, wenn $V = \frac{U}{M}$ ein Falltürpaar darstellt, sind sie insbesondere auch wachsend, d. h. man kann an der Anordnung der $g_{\pi(i)}$ die korrekte Permutation ablesen, falls ein gegebenes Teilintervall eine Lösung enthalten soll. Für die Bedingungen (i) und (ii) muss man dann nur noch jeweils $\pi^{-1}$ anwenden.

**Mehrfach iteriertes System:**
Für das iterierte System werden bei der Ermittlung der $a_i$ die $s_i$ mehrfach modular multipliziert: Man definiert Vektoren $a^1, \ldots, a^y$ durch

$$a_i^1 = W_1 s_i \bmod M_1$$
$$a_i^{j+1} = W_j a_i^j \bmod M_j \text{ für } j = 1, \ldots, y-1.$$

Der öffentliche Schlüssel ist dann $(a_1, \ldots, a_n)$ mit $a_i = a_i^y$.

Bei der Kryptanalyse des iterierten Systems gelingt es im Gegensatz zum einfachen System nicht, den Geheimschlüssel zu ermitteln; der Klartext kann aber trotzdem gewonnen werden. Man wendet wieder den LLL-Algorithmus an, um das urspüngliche Subset-Sum-Problem $S^* = \sum_{i=1}^n x_i a_i$ in $t = \sum_{i=1}^n x_i d_i$ umzuwandeln. Dabei gilt, dass die Folge der $d_i$ für $i > y + \varepsilon$ mit einem „kleinen“

$\varepsilon$ stark wachsend ist; die zugehörigen $x_i$ können also mit dem Greedy-Algorithmus bestimmt werden. Die Werte $x_{y+1}, \ldots, x_{y+\varepsilon}$ müssen erraten werden, was jedoch kein Problem ist, da $\varepsilon$ klein ist; $x_1, \ldots, x_y$ ergeben sich als Lösung eines linearen Gleichungssystems.
Da man bei der Analyse von der Annahme ausgeht, dass $x$ höchstens $\frac{n}{2}$ Einsen enthält, müssen für den Fall, dass dies nicht gilt, sowohl $S^*$ als auch $\sum_{i=1}^{n} a_i - S^*$ betrachtet werden.
Zur Berechnung von $x_{y+\varepsilon+1}, \ldots, x_n$ ermittelt man zunächst mit Hilfe des LLL-Algorithmus $y$ kurze Vektoren $v_j = \sum_{i=1}^{n} h_i^j b_i$ im Gitter $L$, das von den Vektoren $b_1, \ldots, b_n$ erzeugt wird:

$$b_1 = \begin{pmatrix} a_2 \\ a_3 \\ \vdots \\ a_n \\ \frac{1}{n} \end{pmatrix}; \; b_2 = \begin{pmatrix} -a_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ldots, b_n = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -a_1 \\ 0 \end{pmatrix}.$$

Für die erhaltenen Vektoren gelten die Ungleichungen

$$\left| \frac{h_1^j}{a_1} - \frac{h_i^j}{a_i} \right| < \frac{1}{nM_y}, \; i = 1, \ldots, n. \tag{3}$$

Das Rucksackproblem

$$S = \sum_{i=1}^{n} x_i a_i, \; \sum_{i=1}^{n} x_i \leq \frac{n}{2}, \; x_i \in \{0, 1\}$$

lässt sich dann umformen in Rucksackprobleme

$$t_j = \sum_{i=1}^{n} x_i h_i^j, \text{ wobei } t_j = \left\lfloor \frac{h_1^j S}{a_1} \right\rceil$$

($\lfloor z \rceil$ ist im Folgenden die $z$ nächstgelegene ganze Zahl), denn wegen (3) gilt

$$\begin{aligned} \frac{h_1^j a_i}{a_1} &= h_i^j + \delta \text{ mit } |\delta| < \frac{1}{n} \\ \Rightarrow t_j &= \left\lfloor \sum_{i=1}^{n} \frac{x_i h_1^j a_i}{a_1} \right\rceil \\ &= \left\lfloor \sum_{i=1}^{n} x_i h_i^j + \underbrace{\sum_{i=1}^{n} x_i \delta_i}_{< \frac{1}{2}} \right\rceil \\ &= \sum_{i=1}^{n} x_i h_i^j. \end{aligned}$$

Es stellt sich heraus, dass die Folge $d_i = \det \begin{pmatrix} a_1 & \cdots & a_y & a_i \\ h_1^1 & \cdots & h_1^y & h_i^1 \\ \vdots & \cdots & \vdots & \vdots \\ h_1^y & h_y^y & \cdots & h_i^y \end{pmatrix}$ für $i > y + \varepsilon$ stark wachsend ist, wobei $\varepsilon$ klein ist. Setzt man nun $T = \det \begin{pmatrix} a_1 & \cdots & a_y & S \\ h_1^1 & \cdots & h_y^1 & t_1 \\ \vdots & \cdots & \vdots & \vdots \\ h_1^y & h_y^y & \cdots & t_y \end{pmatrix}$, so gilt damit $T = \sum_{i=1}^{n} x_i d_i$.
Da die $d_i$ für $i > y + \varepsilon$ stark wachsend sind, lassen sich diese $x_i$ mit dem Greedy-Algorithmus bestimmen; außerdem gilt $d_i = 0$ für $i \leq y$.
Man bestimmt nun

$$t^* = T - \sum_{i=y+\varepsilon+1}^{n} x_i d_i = \sum_{i=y+1}^{y+\varepsilon} x_i d_i$$

und löst dieses (kleine!) Rucksackproblem durch Ausprobieren aller Möglichkeiten. Es sind jetzt also alle $x_i$ für $i > y+1$ bekannt. Mit

$$S^* = S - \sum_{i=y+2}^{n} x_i a_i \text{ und } t_j^* = t_j - \sum_{i=y+2}^{n} x_i h_i^j \text{ für } j = 1, \ldots, y$$

reduziert sich die Bestimmung der restlichen $x_i$ jetzt auf die Lösung des (fast immer lösbaren) Gleichungssystems

$$\begin{pmatrix} a_1 & \cdots & a_y & a_{y+1} \\ h_1^1 & \cdots & h_y^1 & h_{y+1}^1 \\ \vdots & \cdots & \vdots & \vdots \\ h_1^y & h_y^y & \cdots & h_{y+1}^y \end{pmatrix} x = \begin{pmatrix} S^* \\ t_1^* \\ \vdots \\ t_y^* \end{pmatrix}$$

Danach ist der Klartext vollständig bestimmt.

## 10. Zusammenfassung und Ausblick

Das Merkle-Hellman-Rucksacksystem ist aufgrund der obigen Ergebnisse unsicher, da es in der Regel möglich ist, mit dem angegebenen Verfahren ein Falltürpaar zu bestimmen. Auch Erweiterungen wie Permutation des öffentlichen Schlüssels und Iteration der modularen Multiplikation bringen nur eine unwesentliche Erschwerung der Entschlüsselung durch Unbefugte.
Es sind nach Merkle-Hellman noch weiter auf auf dem Rucksackproblem basierende Kryptosysteme eingeführt worden, die aber letztlich alle kryptanalysiert wurden, zuletzt 1998 das 1984 eingeführte Chor-Rivest-Rucksacksystem, das auf Polynomen über GF($p$) basierte (siehe [**SH95**], [**Vau98**]).

# Literaturverzeichnis


[Bau97] F. L. Bauer, *Entzifferte geheimnisse: Methoden und maximen der kryptologie*, 2nd ed., Springer, 1997.

[Bri83] Ernest F. Brickell, *Solving Low Density Knapsacks*, Advances in Cryptology — CRYPTO '83 (David Chaum, ed.), Plenum Press, 1983.

[Bri84] ———, *Breaking Iterated Knapsacks*, Advances in Cryptology — CRYPTO '84 (G. R. Blakley and David Chaum, eds.), LNCS, vol. 196, Springer, 1984.

[BSW98] Albrecht Beutelspacher, Jörg Schenk, and Klaus-Dieter Wolfenstetter, *Moderne Verfahren der Kryptographie*, 3rd ed., Mathematik, Vieweg, 1998.

[Kux99] Georg Kux, *Kryptanalyse des Merkle-Hellman-Rucksacksystems*, Master's thesis, Universität Kaiserslautern, 1999.

[Mey98] Petra Meyer, *Von Tartaglia, Cardano, kubischen Gleichungen und einem Gedicht*, Vortrag anlässlich des Tags der Mathematik 1998, Universität Kaiserslautern, 1998.

[Rei99] K. Rüdiger Reischuk, *Komplexitätstheorie Band I: Grundlagen*, Leitfäden der Informatik, Teubner, 1999.

[Sch96] Bruce Schneier, *Angewandte Kryptographie*, Addison-Wesley, 1996.

[SH95] C. P. Schnorr and H. H. Hörner, *Attacking the Chor-Rivest Cryptosystem by improved Lattice Reduction*, Advances in Cryptology — EUROCRYPT '95 (J. J. Quisquater and L. C. Guillou, eds.), LNCS, vol. 921, Springer, 1995.

[Vau98] Serge Vaudenay, *Cryptanalysis of the Chor-Rivest Cryptosystem*, Advances in Cryptology — CRYPTO '98 (H. Krawczyk, ed.), LNCS, vol. 1462, Springer, 1998.

[Weg96] Ingo Wegener (ed.), *Highlights aus der Informatik*, Springer, 1996.

[Wob98] Reinhard Wobst, *Abenteuer Kryptologie*, Addison-Wesley, 1998.