Kryptologie

Vorlesung: Prof. P. Hauck
Mitschrieb: Cornelia Schulz, Jan-Peter Hohloch
LaTeX: Jan-Peter Hohloch

WS 14/15

Inhaltsverzeichnis

0.1 Einführung

Kryptologie: Wissenschaft von der sicheren räumlichen und zeitlichen Übertragung von Daten.

0.1.1 Themen

- Geheimhaltung
- Authentifizierung
- (Daten-) Integrität
- Verbindlichkeit
- Anonymität

0.1.2 Vorlesungsinhalt

- Symmetrische Verschlüsselungsverfahren (Block-, Stromchiffren)
- Asymmetrische Verschlüsselungsverfahren (Public-Key Systeme)
- theoretische Grundlagen
- Digitale Signaturen und Authentifizierung
- Zero-Knowledge Beweise
- Secret Sharing Schemes

1 Grundbegriffe

1.1 Encryption & Decryption

Klartext (unverschlüsselte Daten (plain text) über Eingabealphabet R)

$\downarrow$ (Sender (Alice) verschlüsselt mit Schlüssel k_e (encryption))

Chiffretext (verschlüsselte Daten (cipher text) über Chiffrealphabet S)

$\downarrow$ (Empfänger (Bob) entschlüsselt mit Schlüssel k_d (decryption))

Klartext

Verschlüsselung: Verschlüsselungsalgorithmus + Schlüssel zum Verschlüsseln ($k_e \in \mathcal{K}$)
$\Rightarrow$ Verschlüsselungsfunktion $E(m, k_e) = E_{k_e}(m) =: c$
Entschlüsselung: Entschlüsselungsalgorithmus + Schlüssel zum Entschlüsseln ($k_d \in \mathcal{K}'$ (k_d abhängig von k_e)
$\Rightarrow$ Entschlüsselungsfunktion $D(c, k_d) = m$

$k_e \approx k_d$: (k_d aus k_e leicht zu berechnen) Symmetrische Verschlüsselung

sonst: Asymmetrische Verschlüsselung ($\rightarrow$ k_d aus k_e nicht oder nur schwer zu berechnen)
$\Rightarrow$ k_e kann veröffentlicht werden ("public key"); k_d jedoch muss geheimgehalten werden!

1.1.1 Beispiel

$R = S = \{0, ..., 25\},\ \mathcal{K} = \{0, ..., 25\}$
Verfahren: Verschiebechiffre (shift cipher)
Verschlüsselung:

$$m = x_1...x_r \in R \text{ Klartext}$$
$$\text{Schlüssel } i \in \mathcal{K}$$
$$c = (x_1 + i) \mod 26...(x_r + i) \mod 26 = y_1...y_r \text{ Chiffretext}$$

Entschlüsselung:

$$m = (y_1 - i) \mod 26...(y_r - i) \mod 26$$

1.2 Prinzip von Kerkhoffs

Die Sicherheit eines Verfahrens darf nur von der Sicherheit des (Entschlüsselungs-) Schlüssels abhängen, nicht aber von der Geheimhaltung des zugrunde liegenden Algorithmus.

1.3 Angiffe

1.3.1 Wann ist ein System kompromittiert?

- Angreifer kann für gewisse Chiffretexte die zugehörigen Klartexte ermitteln.
- Angreifer kann k_d (zum verwendeten k_e) ermitteln

1.3.2 Arten von Angriffen

Aufsteigend nach mehr Information sortiert. Algorithmus wird danach bewertet, welchen Angriffsarten er standhält.

- Ciphertext Only Attack
- Known Plaintext Attack
- Chosen Plaintext Attack
- Chosen Ciphertext Attack

Brute-force Attack

Beispiel:
Schlüssellänge: 128 Bit
$2^{128} \approx 2.5 \cdot 10^{38}$ Schlüssel
Angenommen: 10^{12} Schlüssel pro Sekunde testbar, 50% der Schlüssel müssen im Schnitt getestet werden
Erfordert: $4 \cdot 10^{18}$ Jahre
(bei 56 Bit nur 10 Stunden)

2 Klassische symmetrische Verschlüsselungsverfahren

Unterscheidung symm. Verschverf.:

- Stromchiffren (stream ciphers):
  Jedes Symbol einzeln verschlüsselt, in der Regel abhängig von der Position
- Blockchiffren (block ciphers):
  Klartext in Blöcke fester Länge n zerlegen, je Block einzeln verschlüsseln (z.B. n=64, 128, 256 Bit)

Klassische Verfahren:

- Substitutionschiffren
- Transpositionschiffren

2.1 Monoalphabetische Substitutionschiffren

Annahme: $R = S$, $|R| = n$
Schlüssel: Permutation σ auf R, Entschlüsselung σ^{-1}
Klartext: $m = x_1 \dots x_r \overset{encrypt}{\rightarrow} c = \sigma(x_1) \dots \sigma(x_r)$
Entschlüsselung: $c = y_1 \dots y_r \overset{decrypt}{\rightarrow} c = \sigma^{-1}(x_1) \dots \sigma^{-1}(x_r)$

Annahme: $|R| = 26$. Dann Anz. Schlüssel 26!

Anmerkung

Verschiebechiffre ist monoalphabetische Substitutionschiffre:
$i \in \{0, \dots, 25\} : \sigma(x) = (x + i) \mod 26$
$\Rightarrow$ 26 Schlüssel

2.1.1 Sicherheit

Verschlüsselte natürlichsprachliche Texte lassen sich selbst bei Ciphertext-Only-Attack auf Grund von Häufigkeitsanalysen leicht dechiffrieren.

2.2 Polyalphabetische Verschlüsselungen

ein Klartextzeichen $\rightarrow$ versch. Chiffretextzeichen
R Klartextalphabet, $S_0, \dots, S_{d-1}$ Chiffretextalphabet ($S_0 = \dots = S_{d-1}$ ist erlaubt)
$f_j : R \rightarrow S_j$ bijektive Abb., für $j = 0, \dots, d-1$
$h : \mathbb{N} \rightarrow \{0, \dots, d-1\}$ (häufig $h(x) = x \mod d$)
Klartext: $m = x_1 \dots x_r$, $x_i \in R$
Chiffretext: $c = f_{h(1)}(x_1) \dots f_{h(r)}(x_r)$, h erzeugt den Schlüsselstrom $(f_{h(1)}, f_{h(2)}, \dots)$
Polyalphabetische Chiffre ist die Folge von monoalphabetischen Substitutionschiffren mit wechselnden Schlüsseln (Stromchiffre).
Falls $h(x) = x \mod d$, so wiederholt sich die Schlüsselfolge periodisch mit Periode d.

2.3 Viginère-Chiffre

Sei im Folgenden $R = S_0 = \dots = S_r$.
$R = \{0, \dots, n-1\}$ (z.B. n=26)
$h(x) = x \mod d$
periodische Verschlüsselung:
$f_j(r) = (r + k_j) \mod n,\ k_j \in R$ fest
Schlüssel: $k_0, \dots, k_{d-1}$, Schlüsselwort: $k_0 \dots k_{d-1}$
$\rightarrow$ es gibt n^d verschiedene Schlüsselwörter.

2.3.1 Konkretes Beispiel

$R = \{0, \dots, 25\},\ A = 0, \dots, Z = 25$
Schlüsselwort: $DONNERSTAG = 3, 14, 13, 13, 4, 17, 18, 19, 0, 6$
Klartext: $VORLESUNGNICHTVERGESSEN$

3	14	13	13	4	17	18	19	0	6	← Schlüsselwort
21	14	17	11	4	18	20	13	6	13	← Klartext
24	2	4	24	8	9	12	6	6	19	← Chiffretext
8	2	7	19	21	4	17	15	0	18	← Klartext
11	16	20	6	25	21	9	8	0	24	← Chiffretext
18	4	13								← Klartext
21	18	0								← Chiffretext

Chiffretext: $YCEYIJMGGTLQUGZVJIAYUSA$
$\Rightarrow$ Gleiche Klartextzeichen können unterschiedlichen Chiffretextzeichen zugeordnet sein. Buchstabenhäufigkeiten werden geglättet:
Bsp.: Wie oft taucht $8 \stackrel{\wedge}{=} I$ in einem bel. Chiffretext auf?

3	14	13	13	4	17	18	19	0	6	← Schlüsselwort
5	20	21	21	4	17	16	15	8	2	← theoretischer Klartext
8	8	8	8	8	8	8	8	8	8	← I

$\Rightarrow$ I hat eine Häufigkeit von $\sim 4.4\%$ im Chiffretext gegenüber 7.7% im Klartext (vgl. Zufallstext: $\frac{100}{26} = 3.8\%$)

2.4 Kryptoanalyse periodischer polyalph. Versch.

Annahme: Ciphertext-Only-Attack

1) Bestimmung von d
2) Entschlüsselung der monoalphabetische verschlüsselten Teiltexte (bei Viginère-Chiffre nur Verschiebechiffre $\Rightarrow$ nur ein Buchstabe muss entschlüsselt werden!)

2) lässt sich durch Häufigkeitsanalysen leicht erreichen
1) mit Kasiski-Test und / oder Friedmann-Test

2.4.1 Kasiski-Test

(Friedrich Kasiski, 1805-1881)

Idee

Wenn sich im Klartext zwei Zeichenfolgen im Abstand $k \cdot d$ ($k \in \mathbb{N}$) wiederholen, so liefern diese gleiche Zeichenfolgen im Chiffretext.
Wir suchen also Wiederholungen von Teiltexten im Chiffretext. Diese könne auch zufällig entstehen, aber das ist selten der Fall (v.a. bei längeren Zeichenfolgen). Die Bestimmung des ggT der Abstände liefert nur wenige Kandidaten für d.

2.4.2 Friedmann-Test

(William Frederic Friedmann, 1891-1969)

Koinzidenzindex

Zeichenfolge m über $R,\ |R| = n,\ R = \{r_1, \dots, r_n\}$
Koinzidenzindex von m $\kappa(m)$ = Wahrscheinlichkeit, dass an zwei zufällig gewählten Positionen in m das gleiche Zeichen steht.
l = Länge von m, l_i = Anzahl, wie oftr_i in m vorkommt ($i \in \{1, \dots, n\}$)
$l = \sum\limits_{i=1}^{n} l_i$
Gesamtzahl der ungeordneten Paare in m ist: $\frac{l(l-1)}{2}$
Gesamtzahl der ungeordneten Paare in m mit dem gleichen Symbol aus R: $\sum\limits_{i=1}^{n} \frac{l_i(l_i-1)}{2}$
Also: $\kappa(m) = \frac{\sum_{i=1}^{n} \frac{l_i(l_i-1)}{2}}{\frac{l(l-1)}{2}} = \frac{\sum_{i=1}^{n} l_i(l_i-1)}{l(l-1)}$
Ist l groß, $p_i = \frac{l_i}{l}$, so ist $\kappa(m) \approx \sum\limits_{i=1}^{n} p_i^2$

Anwendung

Im Folgenden $n = 26$.
In langen deutschen Texten ist der Koinzidenzindex $\kappa(m) = \kappa_d \approx 0.0762$
Außerdem: englisch: 0.067, russisch: 0.0561 (mehr Alphabetszeichen!)

Was nützt der Koinzidenzindex zur Bestimmung von d? Zerlege den Chiffretext c in d monoalphabetisch verschlüsselte Teiltexte; Wahrscheinlichkeit für 2 Positionen innerhalb monoalphabetischer Teiltexte mit gleichen Symbolen $\approx \kappa_d$, in verschiedenen Teiltexten ≈ 0.0385 (wegen Glättung)
$\kappa(c)$ kann man berechnen, Länge l von c
$\kappa(c)$ kann man abschätzen, d monoalphabetisch verschlüsselte Teiltexte, Länge $\frac{l}{d}$
In jedem dieser Texte gibt es $\frac{1}{2} \cdot \frac{l}{d} \left(\frac{l}{d} - 1\right)$ ungeordnete Paare in jedem der Texte. Also insgesamt $d \cdot \frac{1}{2} \cdot \frac{l}{d} \left(\frac{l}{d} - 1\right) = \frac{l(l-d)}{2d}$ Paare in den Teiltexten.
Es gibt $\frac{l(l-1)}{2} - \frac{l(l-d)}{2d}$ Paare mit Position in verschiedenen Teiltexten.
$\Rightarrow \kappa(c) \approx \frac{\frac{l(l-d)}{2d}\kappa_d + \frac{l^2(d-1)}{2d} \cdot \kappa_z}{\frac{l(l-1)}{2}}$
$\Leftrightarrow d \approx \frac{(\kappa_d - \kappa_z)l}{(l-1)\kappa(c) - \kappa_z \cdot l + \kappa_d}$ Bsp.: $\kappa(c) \approx 0.048 \Rightarrow d \approx 3.93$

2.5 Nichtperiodische polyalphabetische Verschlüsselung

Typisch: Viginère-Chiffre, Länge des Schlüsselwortes $\geq$ Länge Klartext
$\rightarrow$ "Lauftextverschlüsselung"
bei $R = \{0, 1\} \rightarrow$ "Vernam-Chiffre"
Falls Schlüsseltext und Klartext sinnvolle Texte sind, so gibt es statistisch begründete Angriffsmöglichkeiten. Häufige Buchstaben treffen oft wieder auf häufige Buchstaben (Schlüssel und Text)

Besserer Ansatz

Addition einer Zufallsfolge
$R = \{0, 1\}$, Zufällige 0,1-Folge
Bitweise Addition mod 2, d.h. XOR, $m_i \oplus k_i = c_i$
Wann ist eine 0,1-Folge der Länge l eine Zufallsfolge?
$\rightarrow$ Sinnlose Frage, entscheidend ist die Erzeugung
Binär symmetrisch Quelle: Erzeugt 0,1 mit Wahrscheinlichkeit $\frac{1}{2}$ unabhängig von vorherigen Bits.
Klartext bitweise auf Zufallsfolge addieren
$m_i \oplus k_i = c_i \Rightarrow c_i \oplus k_i = m_i$
$\rightarrow$ One-Time-Pad

2.5.1 Perfekte Sicherheit

One-Time-Pad ist perfekt sicher (informationstheoretischer Begriff)
Geg.: Chiffrierverfahren mit Verschlüsselungsfunktion E

$$\begin{aligned} M &= \text{Menge aller möglichen Klartexte; M sei endlich} \\ \mathcal{K} &= \text{Menge aller möglichen Schlüssel; } \mathcal{K} \text{ sei endlich} \\ C &= \text{Menge aller möglichen Chiffretexte, d.h. } C = \{c : \exists m \in M, k \in \mathcal{K} : c = E(m,k)\} \end{aligned}$$

Auf M existiert Wahrscheinlichkeitsverteilung $P_M = P$, Schlüssel werden unabhängig von M gleichverteilt gewählt, d.h. $P_{\mathcal{K}}(x) = \frac{1}{|\mathcal{K}|}\ \forall x \in \mathcal{K}$
Liefert Wahrscheinlichkeitsverteilung auf C: $c \in C$:

$$P(c) = \sum_{\substack{x \in M \\ k \in \mathcal{K} \\ E(m,k)=c}} \frac{1}{|\mathcal{K}|} \cdot P(x)$$

Bedingte Wahrscheinlichkeit, dass $m \in M$ verschlüsselt wurde, falls c der Chiffretext ist:

$$P(m|c) = \frac{\sum\limits_{\substack{k \in \mathcal{K} \\ E(m,k)=c}} \frac{1}{|\mathcal{K}|} \cdot P(m)}{P(c)}$$

Definition

Ein Chiffrierverfahren heißt perfekt sicher, wenn für jeden Klartext m und jeden Chiffretext c gilt:
$\underbrace{P(m|c)}_{\text{A posteriori}} = \underbrace{P(m)}_{\text{A priori}}$
Mit Bayes'scher Formel:
Perfekte Sicherheit $\Leftrightarrow P(c) = P(c|m)$ für $P(c) > 0$
Also: Wahrscheinlichkeit von c ist unabhängig von m
Anmerkung: One-Time-Pad ist perfekt sicher bei Einschränkung auf Klartexte einer festen Länge.

Satz (Shannon)

Sei $n \in \mathbb{N}$, $E_n : \overbrace{\{0,1\}^n}^{M} \times \overbrace{\{0,1\}^n}^{K} \to \overbrace{\{0,1\}^n}^{C}$ sei das Verschlüsselungsverfahren des One-Time-Pad. Wird bei der Verschlüsselung von Klartexten immer aufs Neue ein Schlüssel gleichverteilt zufällig gewählt, ist One-Time-Pad perfekt sicher.

Beweis:

Vorbemerkung: $\forall m \in M, c \in C \exists! k \in \mathcal{K} : m \oplus k = c$

$$\begin{aligned}
P(m|c) &= \frac{\sum\limits_{\substack{k \in \mathcal{K} \\ E(m,k)=c}} \frac{1}{|\mathcal{K}|} \cdot P(m)}{\sum\limits_{\substack{x \in M \\ k \in \mathcal{K} \\ E(x,k)=c}} \frac{1}{|\mathcal{K}|} P(x)} \\
&\stackrel{Vorb.}{=} \frac{\frac{1}{|\mathcal{K}|} \cdot P(m)}{\sum\limits_{x \in M} \frac{1}{|\mathcal{K}|} \cdot P(x)} \\
&= \frac{P(m)}{\sum\limits_{x \in M} P(x)}
\end{aligned}$$

Wichtig: Schlüssel darf nur einmal verwendet werden!
Klar: bei Known-Plaintext-Attack:

$$m, m \oplus k = c \rightarrow k = m \oplus c$$

Auch bei Ciphertext-Only-Attack:
$c_1 = m_1 \oplus k,\ c_2 = m_2 \oplus k$ bekannt.
$c_1 \oplus c_2 = m_1 \oplus m_2 \leftarrow$ statistische Angriffsmöglichkeiten

2.6 Stromchiffren

Es gibt zwei Typen von Stromchiffren:

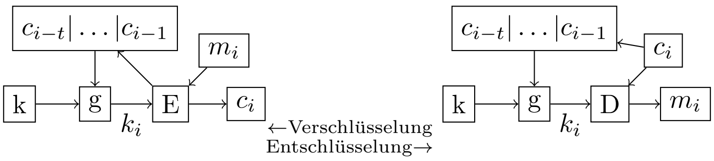
2.6.1 Synchrone Stromchiffren

Schlüsselstrom wird unabhängig von Klar- und Chiffretext erzeugt. Ausgangsschlüssel k. Funktion zur Erzeugung des Schlüsselstroms $g(i, k) = k_i,\ i = 1, 2, \ldots$
Verschlüsselung von $m = m_1 m_2 \ldots,\ m_i \in R : E(m_i, k_i) = c_i,\ c = c_1 c_2 \ldots$
Entschlüsselung: $D(c_i, k_i) = m_i$
Wichtig: Bob und Alice müssen synchron sein. Geht beispielsweise ein Chiffretextbit verloren, schlägt die Entschlüsselung fehl (ab diesem Punkt); die Änderung eines Chiffretextbit führt nicht zu einer Fehlerfortpflanzung.
Falls $E(m_i, k_i) = m_i \oplus k_i = c_i,\ R = \mathbb{Z}_2 = \{0, 1\}$ spricht man von einer "binären additiven Stromchiffre".

2.6.2 Selbstsynchronisierende Stromchiffren

Schlüsselstrom wird erzeugt als Funktion von Ausgangsschlüssel und einer festen Anzahl vorheriger Chiffretextbits.
Ausgangsschlüssel k.

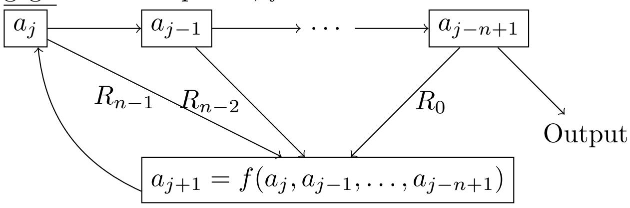
$z_0 = (c_{-t}, \dots, c_{-1})$ Anfangszustand (muss nicht geheim gehalten werden), t fest
$z_i = (c_{i-t}, \dots, c_{i-1}),\ i \geq 1$
Funktion zur Erzeugung des Schlüsselstroms: $k_i = g(z_i, k)$
$c_i = E(m_i, k_i),\ m_i = D(c_i, k_i)$



Selbstsynchronisation:
Geht Chiffretextbit bei Übertragung verloren, so geht das entsprechende Klartextzeichen verloren <u>und</u> die nächsten t Chiffretextzeichen werden (ggf.) falsch entschlüsselt. Ab dann korrekt.

2.6.3 Schieberegister

(Rückgekoppelte) Schieberegister der Länge n
<u>geg.</u>: Endl. Körper K, $f : K^n \to K$



Anfangswerte: $a_0, \dots a_{n-1}$ in Register $R_0, \dots, R_{n-1}$ ($\to$ Anfangszustand)

$$\begin{array}{rcl} 1.\ \text{Zeittakt} & : & \text{Output } a_0 \\ & \vdots & \\ (n+1).\ \text{Zeittakt} & : & \text{Output } a_n = f(a_{n-1}, \dots, a_0) \end{array}$$

Zustand eines SR: Belegung $R_{n-1}, \dots, R_0$
SR befindet sich zu jedem Zeitpunkt in einem von $|K|^n$ verschiedene Zuständen. D.h. es gibt einen Zeitpunkt t_1 zu t_2, zu denen sich das SR im gleiche Zustand befindet, d.h. die Output-Folge wird ab einem Zeitpunkt t_0 periodisch.
D.h. $a_i = a_{i+d}\ \forall i \geq t_0$, wobei d kleinstmögliche Periodenlänge.

2.6.4 Lineare Schieberegister

$f(x_{n-1}, \dots, x_0) = \sum\limits_{i=0}^{n-1} c_i x_i,\ c_i \in K$ fest
Speziell bei $K = \mathbb{Z}_2$: binäres LSR
bei $c_0 \neq 0$: nicht-singuläres LSR (falls $c_0 = 0$ ist LSR durch kürzeres ersetzbar)

Beispiel

a) $f(x_3, x_2, x_1, x_0) = x_2 + x_0$, $(a_3, a_2, a_1, a_0) = (1, 1, 0, 1)$

R_3	R_2	R_1	R_0	*Output*
1	1	0	1	
$a_4 = a_2 + a_0 = 0$	1	1	0	1
$a_5 = a_3 + a_1 = 1$	0	1	1	0
$a_6 = a_4 + a_2 = 1$	1	0	1	1

Anfangszustand $\Rightarrow$ Periode 3

b) $f(x_1, x_0) = x_1 + x_0$, Anfangszustand $(0, 1)$

R_1	R_0	*Output*
0	1	
1	0	1
1	1	0
0	1	1

Anfangszustand $\Rightarrow$ Periode 3

2.6.5 Eigenschaften (binärer) LSR

a) Befindet sich ein LSR im Zustand $(0, \ldots, 0)$, so bleibt es in diesem Zustand. D.h. die maximale Periode eines binären LSR ist $2^n - 1$.
gesucht: LSR mit möglichst langer Periode

b) Nicht-singuläres LSR ($c_0 \neq 0$) erzeugt immer periodische Folgen (ohne Vorperiode).

c) (Binäre) LSR lassen sich als lineare Abbildungen $\mathbb{Z} \to \mathbb{Z}_2^n$ auffassen:

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ & & & \vdots & \\ 0 & 0 & 0 & \ldots & 1 \\ c_0 & c_1 & c_2 & \ldots & c_{n-1} \end{pmatrix}}_{=:C} \begin{pmatrix} x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_{n-1} \\ \underbrace{c_0 x_0 + \cdots + c_{n-1} x_{n-1}}_{x_n} \end{pmatrix}$$

charakteristisches Polynom von C ist

$$\begin{aligned} p(t) &= t^n - c_{n-1} t^{n-1} - \cdots - c_0 \\ &\overset{\mathbb{Z}_2}{=} t^n + c_{n-1} t^{n-1} + \cdots - c_0 \end{aligned}$$

d) Ist $p(t)$ irreduzibel, so hat jede von dem entsprechenden LSR erzeugte Folge (bis auf die Nullfolge) die gleiche Periode d. Ist $n \geq 2$, so ist d die kleinste natürliche Zahl mit der Eigenschaft: $p(t) | t^d - 1$
Anmerkung: Es gilt stets: $p(t) | t^{2^n - 1} - 1$
(Beweis siehe Beutelspacher, Neumann, Schwarzpaul; Satz 6.2)

e) Es gibt $\frac{\phi(2^n - 1)}{n}$ irreduzible Polynome über $\mathbb{Z}_2$, für die $d = 2^n - 1$ gilt. Wobei $\phi(m) = |\{i \in \mathbb{N} : i \leq m, ggT(i, m) = 1\}|$
(Beweis: Lidl, Niederreiter: Introduction to Finite Fields and Applications)

2.6.6 Lineare Komplexität

Ist $(a_0, a_1, \dots)$ (binäre) periodische Folge mit Periode d (oder endliche binäre Folge der Länge d), so gibt es immer mind. ein LSR, das diese Folge erzeugt. LSR der Länge d, $c_0 = 1, c_1 = c_2 = \dots = c_{d-1} = 0$, Anfangszustand: $(a_{d-1}, \dots, a_0)$
Man kann zeigen:
Ist $(a_0, a_1, \dots)$ binäre periodische Folge, so gibt es ein eindeutig bestimmten LSR von minimaler Länge n, das diese Folge erzeugt.
n heißt lineare Komplexität der Folge $(a_0, a_1, \dots)$. Wenn ein LSR der Länge n eine Folge der Periode $2^n - 1$ erzeugt, so hat diese Folge eine lineare Komplexität von n.

2.6.7 Satz

Sei $(a_0, a_1, \dots)$ eine binäre periodische Folge, die von einem binären LSR der Länge n erzeugt wird. Sei $m \in \mathbb{N}_0$. Dann sind folgende Aussagen gleichwertig:

(1) $(a_0, a_1, \dots)$ hat lineare Komplexität n

(2) die n aufeinanderfolgenden Zustandsvektoren $(a_m, a_{m+1}, \dots, a_{m+n-1})$,$(a_{m+1}, \dots, a_{m+n})$, $\dots$, $(a_{m+n-1}, \dots, a_{m+2n-1})$

Beweis

$(1) \Rightarrow (2):$

$f(x_{n-1}, \dots, x_0) = \sum\limits_{i=0}^{n-1} c_i x_i$ sei Funktion des LSR der Länge n, das $(a_0, a_1, \dots)$ erzeugt.

$$C = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & & & \vdots & \\ 0 & 0 & 0 & \dots & 1 \\ c_0 & c_1 & c_2 & \dots & c_{n-1} \end{pmatrix}, \ v_i := (a_i, a_{i+1}, \dots, a_{i+n-1})^t$$

$det(C) = \pm c_0 \neq 0$, $Cv_0 = v_1, C^m v_0 = v_m$
Daher:

$$\begin{aligned} & v_m, v_{m+1}, \dots, v_{m+n-1} && \text{sind linear unabhängig} \\ \Leftrightarrow & C^{-m} v_m, C^{-m} v_{m+1}, \dots, C^{-m} v_{m+n-1} && \text{sind linear unabhängig} \\ \Leftrightarrow & v_0, v_1, \dots, v_{n-1} && \text{sind linear unabhängig} \end{aligned}$$

Zeige also $v_0, v_1, \dots, v_{n-1}$ sind linear unabhängig
Angenommen nicht:
Wähle $t \leq n-1$ minimal, sodass $v_0, v_1, \dots, v_t$ linear abhängig sind. Dann existieren $d_0, \dots, d_t \in \mathbb{Z}_2$ mit $d_0 v_0 + \dots + d_t v_t = 0$ und $d_t \neq 0$, d.h. $d_t = 1$, also $v_t = \sum\limits_{j=0}^{t-1} d_j v_j$.

Dann gilt $\forall i \geq 0 : v_{t+i} = C^i v_t = C^i \left(\sum_{j=0}^{t-1} d_j v_j \right) = \sum_{j=0}^{t-1} d_j C^i v_j = \sum_{j=0}^{t-1} d_j v_{j+i}$

$$\begin{pmatrix} a_{t+i} \\ \vdots \\ a_{t+i+n-1} \end{pmatrix} = d_0 \begin{pmatrix} a_i \\ \vdots \\ a_{i+n-1} \end{pmatrix} + \cdots + d_{t-1} \begin{pmatrix} a_{i+t-1} \\ \vdots \\ a_{i+t+n-2} \end{pmatrix}$$

Betrachte die erste Komponente:
$a_{t+i} = d_0 a_i + \cdots + d_{t-1} a_{i+t-1} \ \forall i \geq 0$
$\Rightarrow$ Die Folge wird erzeugt durch einen LSR der Länge $t \leq n-1$ (mit Koeffizienten $d_0, \ldots, d_{t-1}$) $\lightning$
Also gilt (2).

(2) $\Rightarrow$ (1) :
Angenommen (1) gilt nicht.
Sei $g(x_{r-1}, \ldots, x_0) = \sum_{i=0}^{r-1} d_i x_i$ dir Funktion eines kürzeren LSR, das $(a_0, a_1, \ldots)$ erzeugt, d.h. $r \leq n-1, \ d_0 \neq 0$.
Es ist $a_{r+j} = \sum_{i=0}^{r-1} d_i a_{i+j} \ \forall j \geq 0$. Dann folgt, dass $v_{m+r} = \sum_{i=0}^{r-1} d_i v_{m+i}$, d.h. $v_0, \ldots, v_{r-1}, v_r$ sind linear abhängig. $\lightning$
Also gilt (1).

2.6.8 Satz

$(a_0, a_1, \ldots)$ binäre periodische Folge der linearen Komplexität n. Dann lässt sich das zugehörige minimale LSR aus beliebigen $2n$ aufeinanderfolgenden Folgengliedern eindeutig bestimmen.

Beweis

Bekannt sei $a_r, \ldots, a_{r+2n-1}$. Seien $c_0, \ldots, c_{n-1}$ Koeffizienten des LSR der Länge n, das diese Folge erzeugt. $a_{t+n} = \sum_{i=0}^{n-1} c_i a_{t+i}, \ t = r, r+1, \ldots$

$$\begin{pmatrix} a_{r+n} \\ \vdots \\ a_{r+2n-1} \end{pmatrix} = \underbrace{\begin{pmatrix} a_r & \cdots & a_{r+n-1} \\ \vdots & \ddots & \vdots \\ a_{r+n-1} & \cdots & a_{r+2n-2} \end{pmatrix}}_{=:A} \begin{pmatrix} c_0 \\ \vdots \\ c_{n-1} \end{pmatrix}$$

Nach 3.7 ist $rg(A) = n$, d.h. A ist invertierbar:

$$A^{-1} \begin{pmatrix} a_{r+n} \\ \vdots \\ a_{r+2n-1} \end{pmatrix} = \begin{pmatrix} c_0 \\ \vdots \\ c_{n-1} \end{pmatrix}$$

Bei unbekanntem n:
Der Angreifer testet 3.7 mit $\lfloor\frac{r}{2}\rfloor, \lfloor\frac{r}{2}\rfloor - 1, \dots$ als Kandidaten für n. Sobald er bei einer Zahl m aufeinanderfolgende Zustandsvektoren gefunden hat, die linear unabhängig sind, so ist $m = n$
Beachte:
Wird periodische Folge von LSR der Länge n erzeugt, so auch von LSR'en der Länge $s\ \forall s \geq n$

2.6.9 Beispiel

Folge $\underbrace{101}\underbrace{101}\dots$ erzeugt von LSR mit $f(x_1, x_0) = x_1 + x_0$
<u>geg.</u>: 1101 $\Rightarrow A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$

$$\left(\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right) \to \left(\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right)$$
$$\to \left(\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{array}\right)$$
$$\Rightarrow A^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

2.6.10 Schieberegister zur Schlüsselstromerzeugung

a) Nicht-lineare Funktion
   (1) ein nicht-lineares LSR
   (2) nicht-lineare Funktion mit Input aus mehreren (linear angesteuerten) LSR

b) (1) Alternierender Stop-and-Go-Generator (Beth, Piper, 1984)
   LSR_2 wird getaktet, wenn LSR_1 Output 1 liefert
   LSR_3 wird getaktet, wenn LSR_1 Output 0 liefert
   Ausgabe: $LSR_2 \oplus LSR_3$.
   Beim nicht-getakteten LSR wird der vorherige Output wiederholt; zu Beginn Output 0 bei nicht getaktetem LSR.
   Große Periode, große lineare Komplexität

   (2) Shrinking-Generator (Coppersmith, Krawcyk, Mansur, 1993)
   Hat LSR_1 Output 1, so wird der Output von LSR_2 verwendet, sonst nicht ($\Rightarrow$ kein Output).

2.6.11 Spezielle Stromchiffren

a) A5/1, A5/2 (1982,1989)
   Beruhen auf taktgesteuerten Kombinationen von LSR
   z.B. Verschlüsselung Handy $\rightarrow$ Basisstation (GSM)
   Gelten als unsicher, insbesondere A5/2

b) SEAL* (1993, IBM)
   gilt als sicher

c) RC4* (Ron Rivest, 1987)
   bis 1994 geheim gehalten, gilt nicht mehr als sicher

* beruhen nicht auf LSR

d) EUECRYPT-Netzwerk eSTREAM (2004-2008)
   7 Stromchiffren (4 für Soft-. 3 für Hardware)
   z.B. Trivium (s. Paar, Pelzel, 2.3.3)

$\hookrightarrow$ später: Kryptographisch sichere Pseudozufallsfolgen-Generatoren

2.7 Symmetrische Blockchiffren

R=Klartextalphabet=Chiffretextalphabet
Klartext zerlegen in Blöcke der Länge n (evtl. Padding), Blöcke werden verschlüsselt.
Zunächst: Gleiche Blöcke werden gleich verschlüsselt (unabhängig von ihrer Position im Text)
Verschlüsselung: Permutation ($\rightarrow$ Schlüssel) der Blöcke der Länge n über R
Angenommen alle Permutationen der Blöcke sind als Schlüssel zugelassen: $R = \mathbb{Z}_2,\ 2^n$ Blöcke
Speichern eines Schlüssels erfordert $n \cdot 2^n$ bit
$\Rightarrow$ Nicht alle Blockpermutationen werden verwendet.

2.7.1 Affin-lineare Blockchiffren

$R = \mathbb{Z}_p$, p Primzahl, Blöcklänge n
Schlüssel (A, b) mit A ist invertierbare $n \times n$-Matrix über $\mathbb{Z}_p$, $b \in \mathbb{Z}_p^n$
Verschlüsselung von $v \in \mathbb{Z}_p^n$ (Zeilenvektor): $w = v \cdot A + b$
Entschlüsselung: $(w - b) \cdot A^{-1} = v$

Unsicher unter Known-Plaintext-Attack:
Benötigt:
$v_0, \ldots, v_n$ Klartextblöcke
$w_0, \ldots, w_n$ zugehörige Chiffretextblöcke

Sei $i \geq 1$.

$$\begin{aligned} w_i - w_0 &= (v_i \cdot A + b) - (v_0 \cdot A + b) \\ &= v_i A - v_0 A \\ &= (v_i - v_0) A \end{aligned}$$

$$V = \begin{pmatrix} v_1 - v_0 \\ \vdots \\ v_n - v_0 \end{pmatrix}, \; W = \begin{pmatrix} w_1 - w_0 \\ \vdots \\ w_n - w_0 \end{pmatrix} \;\Rightarrow\; V \cdot A = W$$

Angenommen A ist invertierbar. Dann: $A = V^{-1}W$ bekannt, $w_0 v_0 A + b \Leftrightarrow b = w_0 - v_0 A$ bekannt

2.7.2 Hintereinanderausführung von Blockchiffren, Diffusion und Konfusion

ggf. Erhöhung der Sicherheit durch mehrfaches Hintereinaderausführen einer Blockchiffre (evtl. mit verschiedenen Schlüsseln) ("Produkt","Überchiffrierung")
Bringt z.B. nichts bei affin-linearen Chiffren

$$(vA_1 + b_1)A_2 + b_2 = v \underbrace{A_1 A_2}_{\substack{\text{inv.} \\ n\times n- \\ \text{Matrix}}} + \underbrace{b_1 A_2 + b_2}_{\in \mathbb{Z}_p^n} \Rightarrow \text{affin-linear}$$

Kann sinnvoll sein zur Erhöhung von Diffusion und Konfusion (Shannon, 1949)

Diffusion

Statistische Auffälligkeiten im Klartext sollen im Chiffretext "verwischt" werden. Dazu muss ein Chiffretextzeichen von mehreren Klartextzeichen abhängen. Jede Änderung eines Klartextzeichens soll zur Änderung von mehreren Chiffretextzeichen führen.

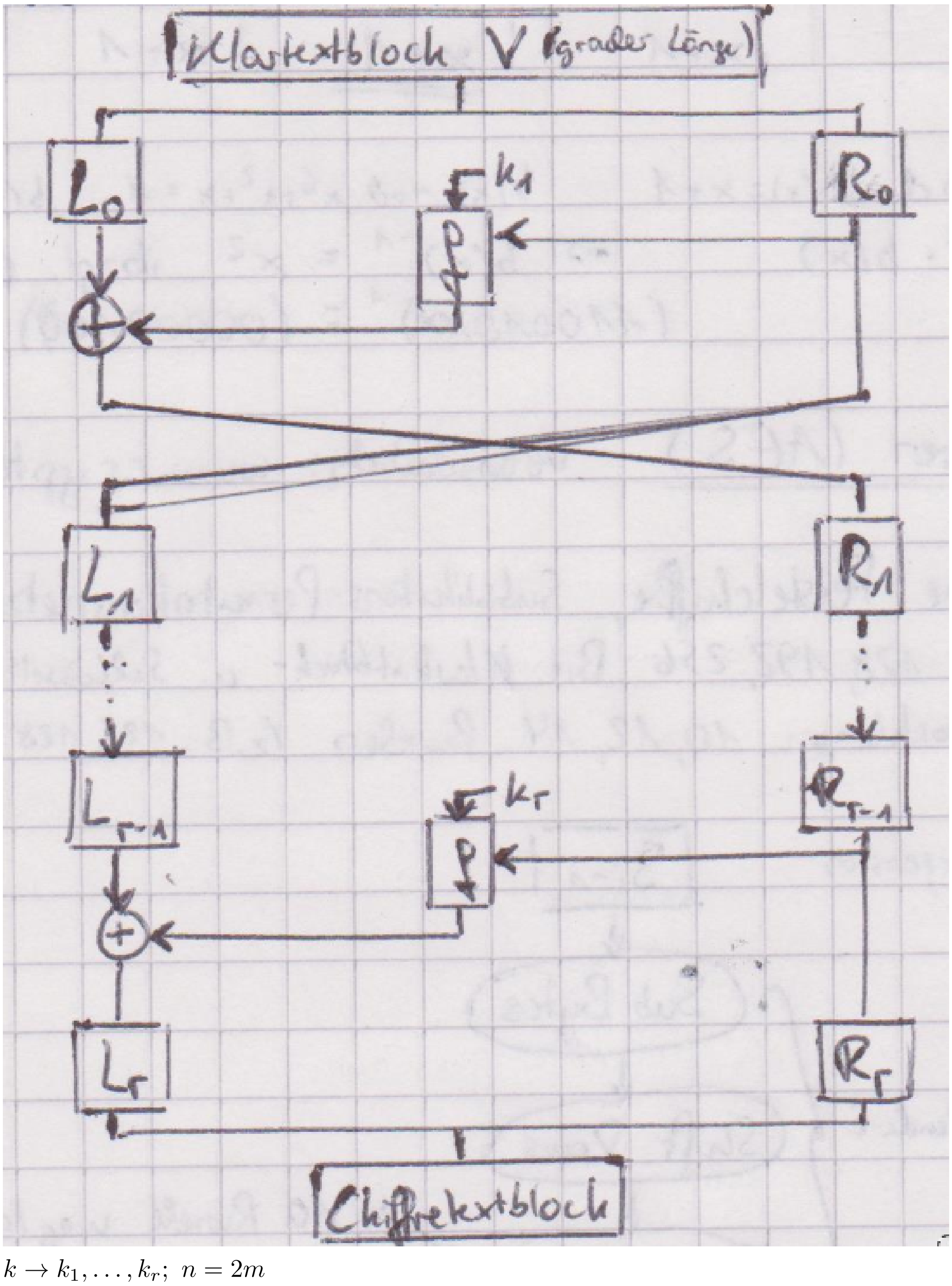
Konfusion

Aus statistischen Auffälligkeiten des Chiffretexts soll nicht auf den verwendeten Schlüssel zu schließen sein. Die Änderung eines Schlüsselbits muss zur Änderung mehrerer Chiffretextbits führen.

2.7.3 Feistel-Chiffre

Prinzip:

- Folge von Blocksubstitution und -transposition (mehrere Runden)
- kurzer Schlüssel liefert Rundenschlüssel $(k_1, \ldots, k_r)$



$k \to k_1, \ldots, k_r;\ n = 2m$
$R_i = L_{i-1} \oplus f_{k_i}(R_{i-1}),\ L_i = R_{i-1}\ \forall 0 < i \le r-1$
$L_r = L_{r-1} \oplus f_{k_r}(R_{r-1}),\ R_r = R_{r-1}$
Anmerkung: f_{k_i} darf nicht affin-linear sein, sonst ist es die ganze Feistel-Chiffre

Entschlüsselung: Selbe Prozedur mit umgekehrter Reihenfolge der Rundenschlüssel.

Auf diesem Prinzip beruht der DES (Data Encryption Standard):
Blocklänge: 64bit, Schlüssellänge: 56bit, sehr gutes Design, Funktion f realisiert durch S-Boxen, inzwischen anfällig gegen Brute-force-Angriffe.
Nachdem DES unsicher wurde schrieb NIST 1997 einen Wettbewerb für neuen Blockchiffren-Standard aus. Es gab 17 Kandidaten, von denen 5 in die Endauswahl kamen. Gewählt wurde 2000 Rijndael (Joan Daeman, Vincent Rijmen, Belgien); dieser wird 2002 zum neuen Standard (AES: Advanced Encryption Standard)

2.7.4 Endliche Körper

a) Körper (engl. field) K: $+,\ \cdot$
  - $(K, +)$ kommutative Gruppe mit neutralem Element 0
  - $(K \setminus \{0\}, \cdot)$ kommutative Gruppe mit neutralem Element 1
  - Distributivgesetz

  z.B. $\mathbb{Q},\ \mathbb{R},\ \mathbb{Z}_p = \{0, \dots, p-1\}$ mit Add. und Mult. $\mod p$

b) Ist Körper K endlich, so $|K| = p^a$ für eine Primzahl p und $a \in \mathbb{N}$

c) Sind K_1, k_2 endliche Körper mit $|K_1| = |K_2|$, so ist $K_1 \cong K_2$ ("isomorph"), also $|K| = p \Rightarrow K \cong \mathbb{Z}_p$

d) Konstruktion von endlichen Körpern der Ordnung $p^a,\ \mathbb{F}_{p^a}$:
$m(x) \in \mathbb{Z}_p[x]$ irreduzibles Polynom vom Grad a (es lässt sich zeigen: $m(x)$ existiert $\forall p, a$)
$\mathbb{F}_{p^a} = \{g(x) \in \mathbb{Z}_p[x] : \deg(g) < a\} = \{c_{a-1}x^{a-1} + \dots + c_1 x + 1 : c_i \in \mathbb{Z}_p\}$
Darstellung: $(c_{a-1} \dots c_1 c_0)$
Addition in $\mathbb{F}_{p^a}$: normale Addition + von Polynomen (koeffizientenweise)
Multiplikation in $\mathbb{F}_{p^a}$: normale Multiplikation von Polynomen mit anschließendem Reduzieren $\mod m(x) \to \odot$
$g_1, g_2 \in \mathbb{F}_{p^a},\ g_1 \odot g_2 = (g_1 \cdot g_2) \mod m(x)$
<u>Bsp.:</u> $\mathbb{F}_{2^8}$
$m(x) = x^8 + x^4 + x^3 + x + 1$, irreduzibel $\mathbb{Z}_2[x]$
$(x^6 + x^4 + x^2 + x + 1) \odot (x^7 + x + 1) = (x^7 + x^6 + 1)$
$\equiv (01010111) \odot (10000011) = (11000001)$

e) Wieso hat jedes Element $\neq 0$ in $\mathbb{F}_{p^a}$ ein Inverses bezüglich $\odot$?
Erweiterter Euklidischer Algorithmus:
In $\mathbb{Z}_p[x]$: $1 = ggT(m(x), g(x)) = u(x) \cdot m(x) + v(x) \cdot g(x)$ für geeignete Polynome $u(x), v(x) \in \mathbb{Z}_p[x]$

$$\begin{aligned} \Rightarrow\ 1 &= ((v(x) \mod m(x)) \cdot g(x)) \mod m(x) \\ &= \underbrace{(v(x) \mod m(x))}_{g^{-1}(x)} \odot g(x) \end{aligned}$$

$b(x) = x^7 + x^6 + x^3 + x + 1$ (Aufg.: $b^{-1}(x)$ bzgl. $\odot$ berechnen $\to$ EEA)
$a(x) = m(x)$

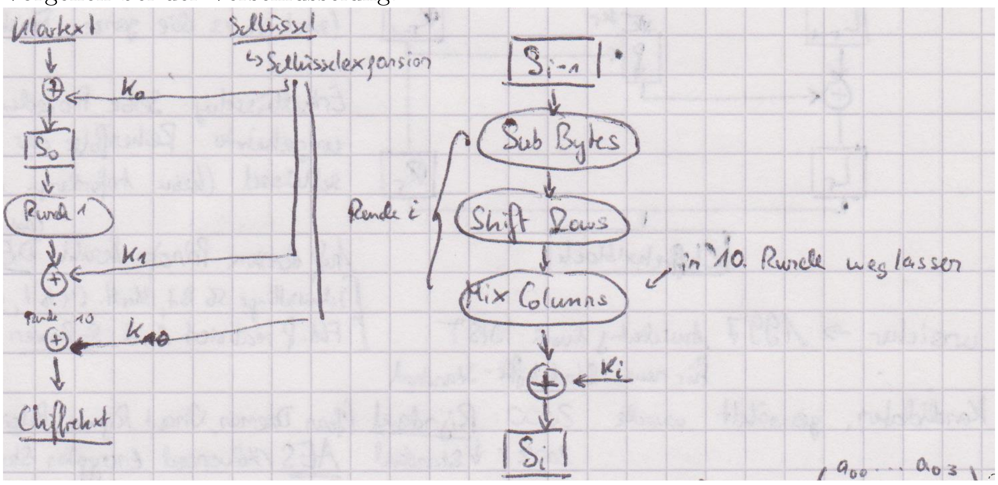
$s(x)$	$t(x)$	$u_1(x)$	$u_2(x)$	$u(x)$	$v_1(x)$	$v_2(x)$	$v(x)$
$m(x)$	$b(x)$	1	0	0	0	1	1
$b(x)$	$x^6 + x^2 + x$	0	1	1	1	$x+1$	$x+1$
$x^6 + x^2 + x$	1	1	$x+1$	$x+1$	$x+1$	x^2	x^2

$\Rightarrow 1 = (x+1)m(x) + x^2 b(x) \Rightarrow b^{-1}(x) = x^2$ bzgl. $\odot$
also $(11001011)^{-1} = (00000100)$

2.7.5 Rijndael-Verfahren (AES)

- Iterative Blockchiffre, *keine* Feistel-Chiffre
- Substitutions-Permutations-Netzwerk (PS-Netzwerk)
- AES lässt unabhängig von einander 128, 192, 256 bit Klartextblock- und Schlüssellänge zu
- abhängig von der Wahl der Blocklängen 10 (12, 14) Runden
- bei je 128 bit, 10 Runden

Vorgehen bei der Verschlüsselung:



- 128bit-Block wird als 4×4-Matrix von Bytes aufgefasst $a_{00}a_{10}\dots a_{30}a_{01}\dots a_{33} \leftrightarrow \begin{pmatrix} a_{00} & \dots & a_{03} \\ \vdots & \ddots & \vdots \\ a_{30} & \dots & a_{33} \end{pmatrix}$
- Sub Bytes: Jedes Byte $g = a_{ij}$ wird einzeln verändert:
  Fasse g als Element von $\mathbb{F}_{2^8}$ auf
  $g \mapsto g^{-1}$ in $\mathbb{F}_{2^8}$ bzw. $0 \to 0$ (*später:* anschließend affine Transformation)

- $g \mapsto g^{-1}$ durch TableLookup in 16×16-Matrix ($\Rightarrow$ effizient)
  Dabei gilt Zeilennummer=$b_7b_6b_5b_4$, Spaltennummer=$b_3b_2b_1b_0$, mit $b_7...b_0$ abzubildendes Byte.

- 'Shift Rows' und 'Mix Columns' dienen der Diffusion (und Konfusion): Zeilen- bzw. Spaltenoperationen an der Matrix (näheres im Skript)

- Der Rundenschlüssel wird mithilfe vorheriger Rundenschlüssel und der SubBytes-Transformation gebildet (genauer: Skript)

- Zur Dechiffrierung werden die Transformationen umgekehrt und die Schlüssel in umgekehrter Reihenfolge verwendet.

Sicherheit von AES

- Schnelle Diffusion: Nach Runden hängt jedes Zustandsbit von jedem Klartextbit ab.

- Sicher gegen differentielle und lineare Kryptoanalyse

- verschiedene Ausgangsschlüssel haben nur wenige Rundenschlüssel gemeinsam

- Es existieren Timing-Angriffe:
  TableLookup-Zeiten sind abhängig vom Index (Bernstein 2005) (2^{27} Blöcke um Schlüssel zu knacken)

- Angriff mit verwandten Schlüsseln (Bivyukov, Kovratovich, 2009):
  Vorgabe von Schlüsseldifferenzen und Klartexten; z.B. 256-AES: 2^{119} verwandte Schlüssel/Klartextblöcke

- Biclique-Angriff (Bogdanov, Khovratovich, Rechberger, 2011):
  Um Faktor 4 schneller als vollständige Schlüsselsuche

2.7.6 Betriebsarten von Blockchiffren

Electronic Codebook Mode (ECB)

Jeder Block wird einzeln, mit demselben Schlüssel verschlüsselt.

Probleme:

- Gleiche Klartextblöcke $\rightarrow$ gleiche Chiffretextblöcke

- Angreifer kann Blöcke ersetzen durch Blöcke, die mit dem gleichen Schlüssel verschlüsselt wurden (z.B. Felder einer Datenbank)

Cipherblock Chaining Mode (CBC)

Klartextalphabet=Chiffretextalphabet=$\{0,1\}$, Blockchiffre mit Blocklänge n, Initialisierungsvektor $IV \in \{0,1\}^n$ (Geheimhaltung nicht nötig)
Verschlüsselung: Klartextblöcke $m_1, ..., m_t$, $c_0 = IV,\ c_j = E_k\,(c_{j-1} \oplus m_j)$, Chiffretext $c_1 \ldots c_t$
Entschlüsselung: $c_0 = IV,\ m_j = D_k(c_j) \oplus c_{j-1}$; falls IV nicht vereinbart wurde, muss m_1 Dummy-Block sein.

Vorteile:

- Gleiche Klartextblöcke werden i.d.R. unterschiedliche verschlüsselt
- Änderung von Chiffretextblock c_i führt zu falschem m_i und $m_i + 1$, ab dann korrekt (Selbstsynchronisation)
- Entsprechend, wenn c_i entfernt wird, analog beim Einfügen

Nachteil:

- Vor dem Verschlüsseln wird kompletter Block benötigt

Cipher Feedback Mode(CFB)

$R = \{0,1\}$, Aufteilung des Klartextes in Blöcke $P_1, P_2, \ldots$ der Länge$r \leq n$ (n Blocklänge der Chiffre, typisch: $r = 8$)
Verschlüsseln: $IV \in \{0,1\}^n$ muss von Sender und Empfänger vereinbart werden, $I_1 = IV$, $j \in \mathbb{N},\ L_r$: linke r Bits
$c_j = P_j \oplus L_r(\underbrace{E_k(I_j)}_{\text{Länge } n}),\ I_{j+1} = R_{n-r}(I_j)|c_j$, mit $|$ Konkatenation

Anmerkungen:

- CFB ist schnell
- Änderung eines c_j wirkt sich solange aus, bis c_j aus $I_j, I_{j+1}, ...$ “herausgeschoben” ist
- Einfügen und Entfernen wie bei CBC (selbstsynchronisierend)
- Bei $r = n$: $c_j = m_j \oplus E_k(c_{j-1})$ (vgl. CBC)

Output Feedback Mode (OFB)

Ähnlich wie CFB, nur $I_{j+1} = R_{n-r}(I_j)|L_r(E_k(I_j)),\ c_j = P_j \oplus L_r(E_k(I_j)),\ I_1 = IV$
Anmerkungen:

- I_j hängen nur vom Schlüssel k und von IV ab
- Änderung eines c_i wirkt sich nur auf P_i aus (bei Entschlüsselung)

- Entfernen und Einfügen verändert alles folgende (nicht selbstsynch.)
- OFB ist schnell
- Bei $r = n : I_{j+1} = E_k(I_j) = E_k^j(I_1),\ c_j = m_j \oplus E_k(I_j) = m_j \oplus E_k^j(I_1)$

3 Public Key Kryptographie (asymmetrische Versch.verfahren)

3.1 Grundidee (Diffie, Hellmann, 1976)

Teilnehmer A hat Paar von Schlüsseln.

1. öffentlicher Schlüssel (public key) p_A
2. geheimer Schlüssel (private key) g_A

Zu p_A gehört öffentlich bekannte Verschlüsselungsfunktion E_{p_A}.
Nachricht (B→ A) $m : m \mapsto E_{p_A}(m) = c$, A entschlüsselt c mit $g_A : \ c \mapsto D_{g_A}(c) = m$.
Damit das Verfahren funktioniert müssen zwei Forderungen erfüllt werden:

1) $E_{p_A}(m)$ ist schnell berechenbar aber schwer zu invertieren ($D_{g_A} = E_{p_A}^{-1}$) $\Rightarrow$ Einwegfunktion

2) A muss aus c schnell m berechnen können (mit g_A) $\Rightarrow$ injektive Einwegfunktion, die mit Zusatzinformation leicht invertierbar ist. Solche Funktionen heißen Geheimtür- oder Falltürfunktionen (trapdoor function)

Aus 1) und 2) folgt: g_A darf sich nicht leicht aus p_A bestimmen lassen.

Anmerkung: Umkehrung von pol. berechenbaren Funktionen liegt in NP, daher P$\neq$ NP notwendig für Existenz von Einwegfunktionen.

3.2 Modulare Potenzen und die RSA-Funktion (Rivest, Shamir, Adleman, 1977)

RSA-Verfahren beruht auf folgender Funktionenklasse:
p, q Primzahlen, $p \neq q$, $n = p \cdot q$, $e > 1$ mit $ggT(e, \varphi(n)) = 1$ (also $1 < e < \varphi(n)$), wobei φ : Euler'sche Funktion
$RSA_{e,n} : \mathbb{Z}_n \to \mathbb{Z}_n, x \mapsto x^e \mod n$ (RSA_e, statt $RSA_{e,n}$, falls n aus Kontext klar)

Euler'sche Phi-Funktion

$\varphi(n) = |\{i \in \mathbb{N} : i \leq n, ggT(i, n) = 1\}|$
$\varphi(n)$ für $n = p \cdot q$:
nicht teilerfremd sind: $p, 2p, 3p, \ldots, (q-1)p, q, 2q, \ldots, (p-1)q, pq(= n)$, also $p + q - 1$ Zahlen $\leq n$ sind nicht teilerfremd zu n
$\Rightarrow \varphi(n) = n - (p + q - 1) = pq - p - q + 1 = (p - 1)(q - 1)$

3.2.1 Satz

e, n wie in 3.2.
Dann ist RSA_{e_n} bijektive Funktion, $RSA_e^{-1} = RSA_d$, wobei d so gewählt ist, dass $e \cdot d \equiv 1 \pmod{\varphi}(n)$ $(0 < d < \varphi(n)$, so ist d eindeutig). d lässt sich mit Hilfe des EEA bestimmen (aus e und $\varphi(n)$)

Kleiner Satz von Fermat

Ist p Primzahl, $a \in \mathbb{Z}$ mit $ggT(a,p) = 1$, dann : $a^{p-1} \equiv 1 \pmod p$

Beweis (Satz)

$ggT(e, \varphi(n)) = 1 \Rightarrow \exists s, t \in \mathbb{Z} : 1 = ggT(e, \varphi(n)) = t \cdot e + s \cdot \varphi(n)$ $(t, s$ berechenbar mit EEA)
Dann: $1 = (t \mod \varphi(n) \cdot e \mod \varphi(n)) \mod \varphi(n) = (t \mod \varphi(n) \cdot e) \mod \varphi(n) = d \cdot e \mod \varphi(n)$
Sei $x \in \mathbb{Z}_n$. $RSA_{e,n}(RSA_{d,n}) = x^{e \cdot d} \mod n$, $e \cdot d = 1 + k \cdot \varphi(n) = 1 + k(p-1)(q-1)$, $k \in \mathbb{N}$
$x^{p-1} \equiv 1 \pmod p$, falls $p \nmid x$; $x^{p-1} \equiv 0 \pmod p$, falls $p | x$ (kl. Satz von Fermat)

$$x^{k(p-1)(q-1)} = \left(x^{p-1}\right)^{k(q-1)} \equiv \begin{cases} 1 \pmod p & \text{falls } p \nmid x \\ 0 \pmod p & \text{falls } p | x \end{cases}$$

$$x^{ed} \mod p = x \cdot x^{k(p-1)(q-1)} \mod p = \begin{cases} x \mod p & \text{falls } p \nmid x \\ 0 \mod p & \text{falls } p | x \\ & (\Rightarrow 0 = x \mod p) \end{cases} = x \mod p$$

genauso: $x^{ed} \equiv x \pmod q \Rightarrow p, q | x^{ed} - x \overset{p,q \text{ prim}}{\Rightarrow} n = p \cdot q | x^{ed} - x \Rightarrow x^{ed} \mod n = x \mod n = x$ (da $x < n$)
$RSA_{e,n}(RSA_{d,n}) = x \; \forall x \in \mathbb{Z}_n$

3.2.2 Square and Multiply

$e \in \mathbb{N}, e = \sum\limits_{i=0}^{k} e_i \cdot 2^i$, $e_i \in \{0, 1\}$ Binärdarstellung von e, $e_k = 1$.

$$m^e = m^{2^k} \cdot m^{e_{k-1} \cdot 2^{k-1}} \cdot \dots \cdot m^{e_0} = \left(\left(\left(n^2 \cdot n^{e_{k-1}}\right)^2 \cdot n^{e_{k-1}}\right)^2 \cdot \dots \cdot n^{e_1}\right)^2 \cdot n^{e_0}$$

$\rightarrow e - 1$ Multiplikationen vs. k Quadrierungen und $\leq k$ Multiplikationen $\Rightarrow \mathcal{O}(k) = \mathcal{O}(\log(e))$ Multiplikationen
Bei $m^e \mod n$ nach jeder Multiplikation $\mod n$ rechnen.

3.2.3 RSA-Verfahren (Basisversion)

Das RSA-Verfahren ist etwa um den Faktor 100-1000 langsamer als AES.

1) Schlüsselerzeugung von A:

a) Wähle zwei große Primzahlen p, q mit $p \neq q$, berechne $n = p \cdot q$ (Dazu Primzahltests; p,q geheim!)
b) Wähle $1 < e < \varphi(n) = (p-1)(q-1)$ mit $ggT(e, \varphi(n)) = 1$ ($\varphi(n)$ geheim)
   $\Rightarrow$ öffentlicher Schlüssel (n,e)
c) EEA: bestimme $1 < d < \varphi(n)$ mit $e \cdot d \equiv 1 \pmod{\varphi}(n)$
   $\Rightarrow$ geheimer Schlüssel d

($p, q, \varphi(n)$ löschen!)

2) Wir verschlüsseln Zahlen $m < n$ (Blockzerlegung in $\log(n)-1$ Blöcke nach Binärkodierung)
   $c = m^e \mod n$ Chiffretext
3) Entschlüsselung: $c^d \mod n = m$ Klartext (nach Satz 3.2.1)

3.2.4 Beispiel

$p = 13,\ q = 23,\ n = p \cdot q = 299,\ \varphi(n) = 12 \cdot 22 = 264 = 2^3 \cdot 2 \cdot 11$
$e = 5$ (kleinstmögl. e teilerfremd zu $\varphi(n)$) $\Rightarrow$ öffentl. Schlüssel (299,5)
d bestimmen mit EEA oder suche kleinstes $k \in \mathbb{N}$ mit $e | k \cdot \varphi(n) + 1$ $\left(d = \frac{k \cdot \varphi(n)}{e}\right)$
$\Rightarrow$ hier $k = 1,\ d = \frac{265}{5} = 53$, geheimer Schlüssel: 53

Verschlüsselung

$m = 212$:

$$\begin{aligned} m^5 \mod 299 &= \left(\left(212^2 \mod 299\right)^2 \mod 299\right) \cdot 212 \mod 299 \\ &= (94^2 \mod 299) \cdot 212 \mod 299 \\ &= 165 \cdot 212 \mod 299 \\ &= 296 \\ \Rightarrow c &= 296 \text{ Chiffretext} \end{aligned}$$

Entschlüsselung

$296^{53} \mod 299$, $(53)_{10} = (110101)_2$
$c^{53} \mod 299 = \left(\left(\left(\left(296^2 \cdot 296\right)^2\right)^2 \cdot 296\right)^2\right)^2 \cdot 296$, jeweils $\mod 299$
$\rightarrow$8 Multiplikationen

3.2.5 Sicherheit von RSA

Hauptfrage: Ist RSA eine Einwegfunktion? *Offen!*
Ebenfalls offen: Ist Invertierung von $RSA_{e,n}$ (e-te Wurzel $\mod n$) genauso schwierig,

wie die Bestimmung des geheimen Schlüssels d? ("RSA-Vermutung")
Bestimmung von d ist "genauso schwierig" wie Faktorisierung von n in p und q.

3.2.6 Satz

Gibt es zur Lösung eines der folgenden Probleme einen polynomialen probabilistischen Algorithmus[1] (in $\log(n)$), so auch für jedes der anderen:

(1) Bestimmung der Faktorisierung von n

(2) Bestimmung von $\varphi(n)$

(3) Bestimmung von d

Beweis

(1) $\Rightarrow$ (2) p, q bekannt, so auch $\varphi(n) = (p-1)(q-1)$✓

(2) $\Rightarrow$ (3) d zu (n, e) bestimmen, bekannt $\varphi(n)$: EEA (wie bei Schlüsselerzeugung)✓

(3) $\Rightarrow$ (1) schwierig (Skript), probabilistisch; hier deshalb (2) $\Rightarrow$ (1)

(2) $\Rightarrow$ (1) $\varphi(n) = (p-1)(q-1) = pq - p - q + 1 = n - p - q + 1$

$$\begin{aligned} \Leftrightarrow p &= n - \varphi(n) - q + 1 \\ \Leftrightarrow n &= (n - \varphi(n) - q + 1) \cdot q \\ \Rightarrow q^2 &= (n - \varphi(n) + 1)q \end{aligned}$$

$\Rightarrow$ Quadratische Gleichung für q, $p = \frac{n}{q} \Rightarrow$ Faktorisierung✓

3.2.7 Bemerkung

Wie schnell lässt sich n in die beiden Faktoren p, q zerlegen? (Faktorisierungsproblem)
Unbekannt, ob hierfür ein polynomialer Algorithmus (in $\log(n)$) existiert.
Naiv: Suche Teiler bis $\sqrt{n}$ (p oder $q \leq \sqrt{n}$) $\Rightarrow \mathcal{O}(2^{\frac{1}{2}\log(n)})$
Besser: Quadratisches Sieb (3.2.8)
Noch besser: Zahlenkörpersieb $\Rightarrow \mathcal{O}\left(e^{(c+\mathcal{O}(1))\log(n)^{frac13}(\log(\log(n)))^{\frac{2}{3}}}\right)$, $c = \sqrt[3]{\frac{64}{9}}$

RSA-Challenge (1991-2007)

- Dez. 2009: 768-Bit-Zahl (232 Dezimalstellen) faktorisiert (2000 CPU-Jahre)
- 1024-bit: $\sim$ 1000 mal länger

[1] Alg. liefert mit Wahrscheinlichkeit ≥ 0.5 Lösung; Alg. macht Zufallswahlen

Aufwand zur Faktorisierung von 1024-bit-Zahl$\widehat{=}$ Aufwand Brute-force-Suche von 80-bit-Schlüssel (2048$\widehat{=}$112, 3072$\widehat{=}$128)
$\Rightarrow$ Empfehlung 2048-RSA, besser 4096-RSA

Bernstein, T.Lange (Nov. 2014): Faktor von einem Teil von Zahle aus einer großen Menge ist schneller als Faktorisierung einer einzelnen fest gegebenen Zahl n.

3.2.8 Quadratisches Sieb

Idee: $n = a^2 - b^2 = (a+b)(a-b)$, berechne $n+1^2, n+2^2, n+3^2, \dots$ bis Quadratzahl entsteht (schnell, wenn p, q nahe beieinander)
Finde a, b mit $n | a^2 - b^2 = (a+b)(a-b)$, teste $ggT(n, a+b),\ ggT(n, a-b)$
Wähle t fest (klein, z.B. 10^6). Seien $p_1, \dots, p_t$ die ersten t Primzahlen. Suche Zahlen $a_1, \dots, a_k$ mit $a_i^2 \mod n = p_1^{e_{i,1}} p_2^{e_{i,2}} \dots p_t^{e_{i,t}}$; $e_{i,j} \geq 0$, $2 \left| \sum_{i=1}^{k} e_{i,j} \right. \forall 1 \leq j \leq t, j \in \mathbb{N}$
$a = a_1 \cdot \dots \cdot a_k$, $b = p_1^{\frac{\sum_{i=1}^k e_{i,1}}{2}} p_2^{\frac{\sum_{i=1}^k e_{i,2}}{2}} \dots p_t^{\frac{\sum_{i=1}^k e_{i,t}}{2}}$
$a^2 \mod n = a_1^2 \cdot \dots \cdot a_k^2 \mod n = b^2 \mod n \Rightarrow n | a^2 - b^2 = (a+b)(a-b)$
Berechne $ggT(a+b, n), ggT(a-b, n)$; falls $1, n$: Pech, sonst p, q (durch EEA)
$\mathcal{O}\left(e^{c (\log n)^{\frac{1}{2}} (\log\log n)^{\frac{1}{2}}}\right)$

3.2.9 Bemerkungen

(a) Falls $m^e < n$ (kleines m, e klein), so berechne normale e-te Wurzel (z.B. binäre Suche) $\Rightarrow$ RSA-Basisversion unsicher
Lösung: Padding

(b) auch für kleine d Angriffsmöglichkeit ($d < m^{0.293}$)

3.3 Bestimmung großer Primzahlen

3.3.1 Grundsätzliches Vorgehen

Zufallswahlen + Primzahltests

- Wann kann man erwarten eine Primzahl gefunden zu haben?
  <u>Primzahlsatz</u> (Hadamard, de la Vallé Poussin, 1896)

  - $\Pi(x) = |\{p : p \text{ Primzahl}, p \leq x\}|$
  - $\Pi(x) \sim \frac{x}{\ln x}$

  $\Rightarrow$ $\frac{\Pi(x)}{x} \sim \frac{1}{\ln x} \Rightarrow$ W-keit für Primzahl $\to 0$ für $x \to \infty$
  - z.B. 150-stellige(dezimal) Primzahl gesucht: Erwarte eine solche nach $\ln(10^{150}) \approx$ 350 Wahlen

- Wie testet man auf Primzahleigenschaft?
  zunächst testen ob n durch kleine Primzahl ($\leq 10^6$) teilbar ist, dann eigentlicher Primzahltest.

3.3.2 Primzahltests

(a) Einfachster Test: Fermat Test:
p Primzahl, $ggT(a,p) = 1$, kl. Satz von Fermat: $a^{p-1} \equiv 1 \pmod p$
Fermat-Test von n: $1 < a < n,\ a \in \mathbb{N}$
Berechen $ggT(a,n)$:

$ggT(a,n) \neq 1$ n zusammengesetzt, a ist Teiler

$ggT(a,n) = 1$ $a^{n-1} \equiv 1 \pmod p$?

ja: wähle neues a, starte neu

nein: n ist zusammengesetzt

Allerdings: Es gibt unendliche viele zusammengesetzte Zahlen, die den Fermat-Test für alle a mit $ggT(a,n)$ bestehen (Carmichael-Zahlen)

(b) Verfeinerung des Fermat-Test: Miller-Rabin-Test
Beruht auf folgender Eigenschaft von Primzahlen:
p ungerade Primzahl, $ggT(a,p) = 1,\ p-1 = 2^d \cdot m,\ 2 \not| m,\ d \geq 1$
kleiner Satz von Fermat:

$$a^{p-1} = a^{2^d \cdot m} \equiv 1 \pmod p \text{ bzw. } a^{2^d \cdot m} = \left(a^{2^{d-1} \cdot m}\right)^2 \equiv 1 \pmod p$$

$a^{2^{d-1} \cdot m} \mod p$ ist Nullstelle von $x^2 - 1 \in \mathbb{Z}_p[x]$, also $a^{2^{d-1} \cdot m} \equiv \begin{cases} 1 \pmod p \\ -1 \pmod p \end{cases}$

Angenommen $a^{2^{d-1} \cdot m} \equiv -1 \pmod p$, $d > 1$, $\left(a^{2^{d-2} \cdot m}\right)^2 \ldots$

Folglich ist p ungerade Primzahl, $a \in \mathbb{N},\ ggT(a,p) = 1,\ p-1 = 2^d \cdot m,\ 2 \not| m$, so ist entweder $a^m \equiv \pmod p$ oder $a^{2^i \cdot m} \equiv -1 \pmod p$ für ein i mit $0 \leq i \leq d-1$.
Miller-Rabin-Test testet obige Eigenschaft mit nn anstelle von p. Teste zunächst ob $a^m \equiv 1$:

ja: neues a wählen

nein: teste $a^{2m} \equiv -1 \pmod n \wedge a^{2^2 m} \equiv -1 \pmod n \wedge \cdots \wedge a^{2^{d-1} m} \equiv -1 \pmod n$

ja: neues a wählen

nein: n ist zusammengesetzt

$\Rightarrow$ Miller-Rabin-Test für ein a : $\mathcal{O}\left((\log n)^3\right)$

Vergleich der Tests

Warum ist die Chance beim Miller-Rabin-Test größer als beim Fermat-Test, eine zusammengesetzte Zahl als solche zu erkennen?
Chinesischer Restsatz: $n = n_1 \cdot \dots \cdot n_r,\ ggT(n_i, n_j) = 1,\ a_1, \dots, a_r \in \mathbb{Z}$

Dann existiert $1 \leq x < n$, eindeutig bestimmt, mit $x \equiv a_i \pmod{n}_i \forall i$.

Angenommen n ist zusammengesetzt, aber keine Primzahlpotenz (schnell feststellbar, so $n = n_1 \cdot n_2,\ ggT(n_1, n_2) = 1$. Dann hat das Polynom $x^2 - 1 \in \mathbb{Z}_n[x]$ mind. 4 verschiedene Nullstellen in $\mathbb{Z}_n$:
$1, -1 \mod n = n-1, x_1 \equiv 1 \pmod{n}_1, x_1 \equiv -1 \pmod{n}_2$ (Chinesischer Restsatz), d.h. $n_1, n_2 | x_1^2 - 1 \Rightarrow n_1 \cdot n_2 | x_1^2 - 1 \Rightarrow n | x_1^2 - 1$, d.h. x_1 ist Nullstelle von $x^2 - 1 \in \mathbb{Z}_n[x],\ x_1 \neq 1, -1$, genauso $x_2 \equiv -1 \pmod{n}_1, x_2 \equiv 1 \pmod{n}_2$
Angenommen $a^{2^d m} \equiv 1 \pmod{n}$, $a^{2^{d-1} m}$ Nullstelle von $x^2 - 1 \in \mathbb{Z}_n[x]$. Wahrscheinlichkeit ist $\geq \frac{1}{2}$, dass $a^{2^{d-1} m} \mod n \neq 1, -1 \mod n$

Man kann zeigen: Von den $\varphi(n)$ vielen a mit $1 \leq a \leq n$ mit $ggT(a, n) = 1$ haben mindestens $\frac{3}{4}$ die Eigenschaft, dass n nicht den Miller-Rabin-Test bezüglich a besteht, falls n zusammengesetzt.
Besteht n den Miller-Rabin-Test für mindestens $\frac{1}{4}\varphi(n) + 1$ as mit $ggT(a, n) = 1$, so ist n Primzahl!

$\Rightarrow$ deterministischer Test, aber nicht polynomial (in der Praxis testet man $1 - 10$ as $\Rightarrow$ probabilistisch)

(c) Es existiert ein deterministischer, polynomialer Primzahltest:
AKS-Test (Agrawal, Kayal, Saxena, 2002)

3.4 Rabin-Public-Key-System (M.O. Rabin, 1979)

A: Zwei große Primzahlen $p \neq q$, $p \equiv 3 \pmod 4, q \equiv 3 \pmod 4$
Berechne $n = p \cdot q \to$ öffentlicher Schlüssel n, geheimer Schlüssel p, q

Versch.: $B \to A, m < n:\ m^2 \mod n = c$ (nicht injektiv, 4 Lösungen!)

Entsch.: A berechnet die Quadratzahlen von $c \mod p$ und $c \mod q$ (Davon gibt es jeweils 2: Nullstelle von $x^2 - (c \mod p)$ in $\mathbb{Z}_p$)
$m_p := c^{\frac{p+1}{4}} \mod p,\ m_q := c^{\frac{q+1}{4}} \mod q$ (Exponent $\in \mathbb{N}$ nach Voraussetzung für p und q)
$m_p^2 \mod p = c^{\frac{p+1}{2}} \mod p = c^{\frac{p-1}{2}} c \mod p = \underbrace{m^{p-1}}_{\equiv 1 \pmod p} c \mod p = c \mod p$,
analog: $m_q^2 \mod q = c \mod q$
$\pm m_p \mod p, \pm m_q \mod q$ sind die Quadratzahlen von $c \mod p$ bzw. $c \mod q$.
Daraus ergeben sich (i.A.) 4 verschiedene Wurzeln von $c \mod n (= p \cdot q)$.
mit dem Chinesischen Restsatz: EEA: $y_p \cdot p + y_q \cdot q = 1, y_p, y_q \in \mathbb{Z}$
$r = (y_p \cdot p \cdot m_q + y_q \cdot q \cdot m_p) \mod n, s = (y_p \cdot p \cdot m_q - y_q \cdot q \cdot m_p) \mod n$

$r^2 \mod p = c \mod p, r^2 \mod q = c \mod q, r^2 \mod n = c$
$\Rightarrow$ 4 Quadratwurzeln von $c \mod n : \pm r \mod n, \pm s \mod n$. Eine davon ist m. Zur Identifikation von m benötigt man Redundanzschema, z.B. letzte 64 bit duplizieren. Nur eine der Quadratwurzeln wird dieses Schema erfüllen.

3.4.1 Satz

$n = p \cdot q,\ p \neq q$ Primzahlen, $p, q \equiv 3 \pmod 4$
Bestimmung aller Quadratwurzeln $\mod n$ einer Zahl c mit $ggT(c, n) = 1$, die Quadrat $\mod n$ sind, ist genauso schwierig, wie die Faktorisierung von n zu bestimmen.

Beweis

Faktorisierung von n bestimmbar $\overset{3.4}{\Rightarrow}$ Quadratwurzeln von c bestimmbar
Quadratwurzeln $\mod n$ von c: $r, n-r, s, n-s$ (alle $< n$)
$n \nmid r - s$, denn sonst $r - s = 0$, d.h. $r = s$↯
$n \nmid r + s$, denn sonst $r + s = n$, d.h. $r = n - s$↯
$\Rightarrow ggT(n, r + s) = p$ oder $q \rightarrow$ Faktorisierung

3.5 Diskreter Logarithmus

p Primzahl, $\mathbb{Z}_p\{0, \ldots, p-1\}$, Addition und Multiplikation $\mod p$: Körper
$\left(\mathbb{Z}_p^*, \odot\right)$ Gruppe, ist zyklisch, d.h. $\exists g \in \mathbb{Z}_p^* = \{1, \ldots, p-1\}$ mit $\mathbb{Z}_p^* = \{g^0 = 1, g^1 = g, \ldots, g^{p-2}\}$
g heißt Primitivwurzel $\mod p$. Es gibt genau $\varphi(p-1)$ Primitivwurzeln $\mod p$, nämlich alle g^i mit $ggT(i, p-1) = 1$.
$EXP_{p,g} : \mathbb{Z}_{p-1} \rightarrow \mathbb{Z}_p^* : i \mapsto g^i \mod p$ (diskrete Exponentialfunktion $\mod p$ zur Basis g)
Umkehrabb: $Log_{p,g} : \mathbb{Z}_p^* \rightarrow \mathbb{Z}_{p-1} : g^i \mod p \mapsto i$ diskreter Logarithmus
$\{EXP_{p,g}\}$ Familie von Kandidaten für Einwegfunktionen ($EXP_{p,g}$ einfach, $Log_{p,g}$ schwierig (?))

3.5.1 Diffie-Hellmann-Verfahren (1976)

Ursprüngliche Anwendung des diskreten Logarithmus: Schlüsselvereinbarung über unsicheren Kanal
A und B einigen sich auf (große) Primzahl p und Primitivwurzel $g \mod p$ (p, g dürfen öffentlich bekannt sein)
A wählt zufällig $a \in \{2, \ldots, p-2\}$ und berechnet $g^a \mod p = x$
B wählt zufällig $b \in \{2, \ldots, p-2\}$ und berechnet $g^b \mod p = y$
(a, b geheim, x, y öffentlich)
A berechnet $y^a \mod p = g^{ab} \mod p = K$, B berechnet $x^b \mod p = g^{ab} \mod p = K$, K ist gemeinsamer Schlüssel.

3.5.2 Bestimmung von p, g

Gegeben p, wie bestimmt man g?
g Primitivwurzel $\mod p \Leftrightarrow g^i \mod p \neq 1 \forall i \in \mathbb{Z}_{p-1}^*$
Falls alle Primteiler q von $p-1$ bekannt sind, dann: g Primitivwurzel $\mod p \Leftrightarrow g^{\frac{p-1}{q}} \mod p \neq 1 \forall q$
Es gibt $\mathcal{O}(\log p)$ Primteiler von $p-1 \Rightarrow$ polynomialer Test
Wähle p (Primzahl!) von der Form $2r+1$, r Primzahl; dann $p-1 = 2r \Rightarrow$ Primteiler $2, r$ (r ist Sophie-Germain-Primzahl)
$\Rightarrow$ Zufallswahl von g ist mit Wahrscheinlichkeit $\sim \frac{1}{2}$ Primitivwurzel.

Sophie-Germain-Primzahlen

Offene Frage: Gibt es unendlich viele Sophie-Germain-Primzahlen?
Es gibt 26,569,515 Sophie-Germain-Primzahlen $\leq 10^{10}$
Zum Vergleich: $\Pi(10^{10}) = 455,052,511$ ($\Rightarrow \sim 5\%$ sind Sophie-Germain-Primzahlen)

3.5.3 Sicherheit von Diffie-Hellmann-Verfahren

a) Angreifer kennt $p, q, g^a \mod p, g^b \mod p$. Er will $g^{ab} \mod p$ (Diffie-Hellmann-Problem)
   Diffie-Hellmann-Problem ist höchstens so schwer wie Diskreter-Logarithmus-Problem. Gleich schwer?

b) Es gibt viele Algorithmen für das Diskreter-Logarithmus-Problem; es sind jedoch keine polynomialen bekannt.

3.5.4 Babystep-Giantstep-Algorithmus (Shanks, 1971)

geg. Primzahl p, Primitivwurzel $g \mod p$, $x \in \mathbb{Z}_p^*$
Suche $a \leq p-1$ mit $g^a \mod p = x$.
Setze $t := \lceil \sqrt{p-1} \rceil$. Dann $a = q \cdot t + r$, $0 \leq r < t$ (Division mit Rest)

$$x = g^a = g^{qt+r} = g^{qt} g^r \Leftrightarrow g^{qt} = g^{-r} x \tag{3.1}$$

Algorithmus sucht q, r, sodass (3.1) gilt.
2 Listen werden angelegt

- Babystep-Liste: $xg^{-r} \mod p$, $0 \leq r < t$
- Giantstep-Liste: $g^{qt} \mod p$, $0 \leq q < t$

Beachte: $q < t$, da sonst $qt \geq t^2 \geq p-1$ mit $a \geq qt \Rightarrow a \geq p-1 ↯$
Speicherplatzbedarf: $\mathcal{O}(\sqrt{p})$. (Giantstep-Liste nur einmal erzeugen, da unabhängig von x)
Suche zwei gleiche Einträge in der Babystep- und Giantstep-Liste:
$x \cdot g^{-r} \mod p = g^{qt} \mod p$, $a = qt + r$ ✓

3.5.5 Man-in-the-Middle-Angriff auf Diffie-Hellmann-Verfahren

$$\begin{tikzcd} A \arrow[r, bend left, "g^a \mod p"] & M \arrow[l, bend left, "g^c \mod p"] \arrow[r, bend left, "g^c \mod p"] & B \arrow[l, bend left, "g^b \mod p"] \end{tikzcd}$$

$g^{ac} \mod p = K_A \leftrightarrow g^{bc} \mod p = K_B$
A,B glauben gemeinsamen Schlüssel vereinbart zu haben, tatsächlich haben sie jedoch jeweils unterschiedliche Schlüssel mit M vereinbart. Damit läuft alle Kommunikation über M, welcher Nachrichten abfangen, verändern und erfinden kann.
Man-in-the-Middle-Angriffe sind durch Authentifizierung vermeidbar.

3.5.6 ElGamal-Public-Key-Verfahren (T.ElGamal, 1984)

Anmerkung: Beruht auf Diffie-Hellmann-Verfahren

<u>Schlüsselerzeugung:</u>
A wählt Primzahl p, Primitivwurzel $g \mod p$, zufällig $a \in \mathbb{Z}_{p-1} \setminus \{0,1\}$ geheim, berechnet $x = g^a \mod p$.
$\rightarrow$ Öffentlicher Schlüssel: (p, g, x), geheimer Schlüssel: a
<u>Verschlüsselung:</u>
B$\xrightarrow{m}$A, $m \in \mathbb{Z}_p^*$
B wählt zufällig $b \in \mathbb{Z}_{p-1} \setminus \{0,1\}$; berechnet $y = g^b \mod p$
B berechnet $f = x^b \cdot m \mod p$, Chiffretext (y, f)
<u>Entschlüsselung:</u>
A berechnet: $(y^a)^{-1} \cdot f \mod p = m$, da $y^a \equiv x^b \equiv g^{ab} \pmod p$
Dabei y^{-a} mittels EEA oder $y^{-a} = y^{p-1-a}$, $p - 1 - a > 0$, da $a \leq p - 2$

Effizienz

Vergleich RSA - ElGamal:
n, p gleiche Länge (mind. 1024 bit)
1 Expon. $\mod n$ - 2 Expon. $\mod p$ und 1 Mult. $\mod p$
Falls b fest, so nur einmal $g^b \mod p$ und $x^b \mod p$.
Chiffretext hat bei ElGamal doppelte Länge des Klartextes.

Sicherheit

(a) ElGamal ist genauso sicher wie Diffie-Hellmann

(b) Wenn B bei wiederholter Kommunikation mit A immer neues b wählt, so wird Kryptoanalyse erschwert. Es wird auch folgender Angriff wirkungslos:
Angenommen Angreifer E kennt (y, f) und zugehöriges m (Known-Plaintext). Angenommen m' wird mit gleichen b verschlüsselt, dann ist auch y unverändert.

$f' = x^b \cdot m' \mod p, f = x^b \cdot m \mod p$ bekannt $\Rightarrow fm^{-1} \mod p = x^b \mod p$ bekannt $\Rightarrow \left(x^b\right)^{-1} f' \mod p = m'$

(c) Gelegentlich wird g nicht als Primitivwurzel $\mod p$ gewählt. dann bilden Potenzen von g kleinere Untergruppe von $\mathbb{Z}_p^*$, wobei $|\langle g\rangle|\,|p-1$. Kein Problem, falls $|\langle g\rangle|$ groß genug.
Im Falle von Sophie-Germain-Primzahl q mit $p = 2q+1$:
$g \in \mathbb{Z}_p^* \setminus \{1, p-1\} \Rightarrow |\langle g\rangle| = q \vee |\langle g\rangle| = 2q = p-1$

3.6 Elliptische Kurven Kryptografie

Modifikation von Diffie-Hellmann bzw. ElGamal

(a) Diffie-Hellmann und ElGamal funktionieren mit jeder zyklischen Gruppe $G =< g >$. $g^{-a} = g^{|G|-a},\ g^{|G|} = 1$, z.B. $G = (\mathbb{Z}_p, \oplus)$ zyklische Gruppe. Statt Potenzen Vielfache: $g \cdot m \mapsto c$.
Allerdings unsicher: g^{-1} durch EEA effizient berechenbar, $g^{-1}c = m$

(b) Als genügend für D-H und ElGamal haben sich Punktgruppen elliptischer Kurven (anstelle von $\mathbb{Z}_p$) erwiesen.
Was ist eine elliptische Kurve über Körper K? (Voraussetzung: $1{+}1 \neq 0, 1{+}1{+}1 \neq 0$)
Nach geeigneten Transformationen: Elliptische Kurve $y^2 = x^3 + ax + b,\ a, b \in K$
Lösungsmenge dieser Gleichung für $K = \mathbb{R}$ ist beispielsweise die Wurzel aus den positiven Teilen der Funktion $y = x^3 + ax + b$

3.6.1 Addition auf elliptischen Kurven

Auf den Punkten einer elliptischen Kurve $\cup\{\mathcal{O}\}$ lässt sich eine Addition $\oplus$ definieren. Bezüglich $\oplus$ wird $E \cup \{\mathcal{O}\}$ eine kommutative Gruppe mit neutralem Element $\mathcal{O}$. $\oplus$ wird wie folgt definiert:

(1) $P \oplus \mathcal{O} = \mathcal{O} \oplus P = P\ \forall P \in E \cup \{\mathcal{O}\}$

(2) $P, Q \in E \cup \{\mathcal{O}\}, P = (x_1, y_1), Q = (x_2, y_2),\ x_1 \neq x_2$:
$P \oplus Q = (x_3, y_3)$ mit
$x_3 = \left(\frac{y_2-y_1}{x_2-x_1}\right)^2 - x_1 - x_2,\ y_3 = \left(\frac{y_2-y_1}{x_2-x_1}\right)^2 (x_1 - x_3) - y1$

(3) $P = (x_1, y_1), Q = (x_1, -y_1)$ ($\Rightarrow y_1 = 0$ oder $y_1 \neq -y_1$, da $1 + 1 \neq 0$)
$P \oplus Q = \mathcal{O}$

(4) $P(x_1, y_1), y_1 \neq 0$
$P \oplus P = (x_3, y_3), x_3 = \left(\frac{3x_1^2+a}{2y_1}\right)^2 - 2x_1,\ y_3 = \left(\frac{3x_1^2+a}{2y_1}\right)^2 (x_1 - x_3) - y_1$

Inverse bzgl. $\oplus$:
$-\mathcal{O} = \mathcal{O}$
$P = (x_1, y_1) \in E \Rightarrow -P = (x_1, -y_1) \in E$

3.6.2 Anmerkungen

- Wenn $|K| = q$, so $|E \cup \{\mathcal{O}\}| = |G| \leq 2q + 1$
  $q + 1 - \sqrt{q} \leq |G| \leq q + 1 + 2\sqrt{q}$ (Satz von Hasse)

- Es gibt Algorithmen für das Diskreter-Logarithmus-Problem, die in jeder Gruppe funktionieren:
  - Brute-force $\mathcal{O}(|G|)$
  - Babystep-Giantstep $\mathcal{O}(\sqrt{|G|})$

  In $\mathbb{Z}_p^*$ gibt es spezielle Algorithmen, die deutlich schneller sind als $\mathcal{O}(\sqrt{p})$ (Indexkalkül), jedoch nicht polynomial in $\log(p)$.
  In Punktgruppen von elliptischen Kurven über $\mathbb{Z}_p$ sind keine DL-Algorithmen bekannt, die besser als $\mathcal{O}(\sqrt{p})$ sind.

- Elliptische Kurve über $\mathbb{Z}_p$, p 256-Bit-Primzahl, ist ähnlich sicher wie D-H oder ElGamal auf $\mathbb{Z}_q^*$ mit q 3072-Bit-Primzahl
  $\Rightarrow$ deutlich effizienter

- Benötigt werden elliptische Kurven, die Punkte P enthalten, sodass $|\langle P \rangle| \in \Theta(p)$.
  Solche Kurven existieren (Standardisierung FIPS 186-3, 2009)

4 Digitale Signaturen und kryptographische Hashfunktionen

4.1 Signaturen

4.1.1 Anforderungen

Signatur von A

- niemand außer A kann Dokumente mit Signatur von A versehen, selbst wenn er Signatur von A aus anderen Dokumenten kennt. A kann dann nicht abstreiten, das Dokument signiert zu haben.
- Signatur passt nur zu signiertem Dokument und zu keinem anderen.
- jeder Empfänger muss die Signatur verifizieren können.

4.1.2 Schema (vereinfacht)

A hat Signaturfunktion s_A(geheim), Verifikationsfunktion v_A(öffentlich). s_A lässt sich aus v_A nicht herleiten.
A sendet $(m, s_A(m))$, Empfänger prüft mit Hilfe von v_A Echtheit der Signatur. Oft: $v_A = s_A^{-1}, v_A(s_A(m)) = m$
Dann muss v_A Einwegfunktion sein.
$\rightarrow$ Signaturschema mit Hilfe von PublicKey-System
<u>Wichtig:</u> Verlässlichkeit der öffentlichen v_A-Verzeichnisse ($\rightarrow$ PublicKey-Infrastucture PKI)

<u>Problem:</u> Wenn Verschlüsselungsfunktion zur Bildung von s_A verwendet wird, so ist die Signatur so lang wie das Dokument. Daher: Bilder Hashwert $H(m)$ und signiere diesen $(m, s_A(H(m)), H$ öffentlich bekannt. Verifikation dann: $H(m) \stackrel{?}{=} v_A(s_A(H(m)))$

4.2 Hashfunktionen

4.2.1 Definition

R endl. Alphabet (oft $\{0, 1\}$)
<u>Hashfunktion</u> $H : R^* \rightarrow R^n$, n fest, effizient berechenbar
Also: Hashfunktionen sind nie injektiv, es existiert immer $m, m' \in R^*, m \neq m'$ mit $H(m) = H(m')$

4.2.2 Kryptographische Situation

- $(m, s_A(H(m)))$ wird von A versandt. Jeder Angreifer kann mit v_a $H(m)$ ermitteln. Gelingt es ihm ein $m' \neq m$ zu finden mit $H(m') = H(m)$, dann ist $(m', s_A(H(m)))$ gültige Signatur von m' durch A.
- Angreifer wählt zufällig y und berechnet $v_A(y) = z$. Angenommen er findet Nachricht m mit $H(m) = z$, so ist (m, y) eine gültige Signatur von m durch A: $s_A(H(m)) = s_A(z) = s_A(v_A(y))$

4.2.3 Definition

Eine kryptographische Hashfunktion ist eine Hashfunktion, die folgende Bedingungen erfüllt:

(1) H ist Einwegfunktion

(2) H ist schwach kollisionsresistent, d.h. für geg. $m \in R^*$ ist es nicht effizient möglich, ein $m' \in R^*,\ m' \neq m$ zu finden mit $H(m') = H(m)$

(3) H ist stark kollisionsresistent, d.h. es ist nicht effizient möglich $x, x' \in R^*,\ x \neq x'$ zu finden mit $H(x) = H(x')$ (Verschärfung von (2))

Wegen (1) ist nicht bekannt, ob kryptographische Hashfunktionen existieren.

4.2.4 Satz (Geburtstagsparadoxon)

Ein Merkmal komme in l verschiedenen Ausprägungen vor. Jedes Element von Menge M besitze genau eine dieser Ausprägungen.
Ist $|M| = k \geq \frac{1+\sqrt{1+8l\cdot\ln(2)}}{2} \approx 1.18l$, so enthält M mit der Wahrscheinlichkeit $\geq \frac{1}{2}$ zwei Elemente mit gleicher Merkmalsausprägung.

Beweis

Ereignisse: $(g_1, \dots, g_k) \in \{1, \dots, l\}^k$; $M = \{1, \dots, k\}, g_i$ Merkmalsausprägung von $i \in M$
Gesamtzahl: l^k, alle g_i verschieden $\prod\limits_{i=0}^{k-1}(l-i)$
Wahrscheinlichkeit, dass *keine* zwei Elemente aus M gleiche Merkmalsausprägung haben:

$$q = \frac{\prod\limits_{i=0}^{k-1}(l-i)}{l^k} = \prod_{i=0}^{k-1}\left(1 - \frac{i}{l}\right)$$

$$
\begin{aligned}
e^x &\geq 1+x \\
\Rightarrow 1-\frac{i}{l} &\leq e^{-\frac{i}{l}} \\
\Rightarrow q &\leq e^{-\sum_{i=0}^{k-1}\frac{i}{l}} && = e^{-\frac{1}{l}\frac{(k-1)k}{2}} \\
\Rightarrow \ln(q) &\leq -\frac{1}{l}\frac{(k-1)k}{2}
\end{aligned}
$$

Für $q \leq \frac{1}{2}$:

$$
\begin{aligned}
-\ln(2) &\leq -\frac{1}{l}\frac{(k-1)k}{2} \\
\Leftrightarrow \ln(2) &\leq \frac{1}{l}\frac{(k-1)k}{2} \\
\Leftrightarrow k &\geq \frac{1+\sqrt{1+8l\cdot\ln(2)}}{2}
\end{aligned}
$$

4.2.5 Geburtstagsattacke

$H : \mathbb{Z}_2^* \to \mathbb{Z}_2^n$ Hashfunktion
Angreifer erzeugt $2^{\frac{n}{2}}$ Hashwerte. Mit Wahrscheinlichkeit $\sim 50\%$ Kollisionen.
$\Rightarrow n = 64$ nicht sicher, mind. $n = 128, 160$

4.3 Konstruktion von Hashfunktionen

4.3.1 Serielles Hashing mit Kompressionsfunktionen

Kompressionsfunktion $K : R^l \to R^n, l > n$ (fest)
Konstruktion einer Hashfunktion $H : R^* \to R^n$
$m \in R^*$: Zerlege m in Blöcke $m_1, \ldots, m_t$ der Länge $l-n$ (eventuell Padding). Wähle Initialisierungsvektor $IV =: h_0 \in R^n$ (öffentl. bekannt)
$h_1 := K(m_1, h_0), h_2 := K(m_2, h_1), \ldots, h_t := K(m_t, h_{t-1}) \to H(m) = h_t$

4.3.2 Hashfunktionen unter Verwendung von Blockchiffren

Realisiere 4.3.1 durch Kompressionsfunktion, die auf Blockchiffren beruht:
Blockchiffre über R: Blocklänge=Chiffreblocklänge n, Schlüssellänge s, Verschlüsselungsfunktion E
$\underbrace{(\overbrace{k}^{\in R^s}, \overbrace{x}^{\in R^n})}_{\in R^{s+n}} = E(k,x) = E_k(x) \in R^n$, Kompressionsfunktion oder andere Kompressionsfunktion daraus ableiten.

4.4 Spezielle Hashfunktionen

- MD5 (Message Digit 5, Ron Rivest, 1992) $n = 128$
- SHA-1 (Secure Hash Algorithm, NSA, NIST, 1992/93, 1998) $n = 160$)
- SHA-2-Familie (SHA-224,256,384,512; NIST 2003/04)

Angriffe gegen $MD5, SHA-1$ (im Wesentlichen bzgl. starker Kollisionsresistenz, $2^{51} - 2^{57}$ statt 2^{80} Werte benötigt)

- 2007: NIST-Ausschreibung zu SHA-3
  $\rightarrow$ 2012 Keccak (Bertoni, Daemen, Peeters, van Assche)

4.5 Signaturschema mit Hashfunktionen

geg. s_A, v_A (wie 4.1.2) und öffentlich bekannte (kryptographisch sichere) Hashfunktion H
A sendet an B $(m, s_A(H(m)))$
B verifiziert $v_A(s_A(H(m))) = H(m)$

4.6 RSA-Signatur

$m \in \mathbb{Z}_2^*, (n, e)$ öffentlicher Schlüssel, d geheimer Schlüssel von A (RSA)
$H : \mathbb{Z}_2^* \to \mathbb{Z}_2^t, 2^t \leq n$ kryptographisch sichere Hashfunktion (öffentlich bekannt)
A sendet an B $(m, (H(m))^d \mod n)$
B verifiziert $\left((H(m))^d \mod n\right)^e \mod n = H(m)$
<u>Praxis:</u> Expansionsfunktion auf $H(m)$, das wird verschlüsselt mit d (PKCS # 1)

4.7 ElGamal-Signatur (1985)

A hat öffentlichen ElGamal-Schlüssel (p, g, x); geheimen Schlüssel a ($x = g^a \mod p$)
Signatur von $m \in \mathbb{Z}_2^*$:
öffentlich bekannte Hashfunktion $H : \mathbb{Z}_2^* \to \{1, \ldots, p-1\}$
A wählt Zufallszahl $k \in \{1, \ldots, p-1\}, ggT(k, p-1) = 1$
A berechnet $r = g^k \mod p, s = k^{-1}(H(m) - ar) \mod (p-1)$
Signatur: $(m, s_A(m)), s_A(m) = (r, s)$

B verifiziert:

(1) $1 \leq r \leq p - 1$

(2) $x^r \cdot r^s \equiv g^{H(m)} \pmod p$

<u>Korrektheit:</u> (1)✓(2) $x^r r^s \mod p = g^{ar} g^{ks} \mod p = g^{H(m)} \mod p$

4.7.1 Variante von ElGamal

DSA (Digital Signature Algorithm) (1994, FIPS168 $\to$ DSS)
Primzahl p: 1024/2048/<u>3072</u> Bit, Exponentationen erfolgen in Untergruppe von $\mathbb{Z}_p^*$ von Primzahlordnung q (160/224/256 Bit)
Signaturlänge: 1024+160 Bit

4.8 Message Authentification Code (MAC)

MAC = schlüsselabhängige kryptographische Hashfunktion
Familie $\{H_k, k \in \mathcal{K}\}$ von kryptographischen Hashfunktionen, $\mathcal{K}$ Schlüsselraum
A,B tauschen $k \in \mathcal{K}$ auf sicherem Wege aus (k geheim). A sendet B $(m, H_k(m))$. Verifizierbar durch B.
<u>Problem:</u> A und B können beide $(m, H_k(m))$ erstellen ($\Rightarrow$ keine Signatur)
<u>Vorteil:</u> i.d.R. schnell

4.8.1 Konstruktionsmöglichkeiten für MACs

(a) CBC-MAC (Cipherblock-Chaining-Mode)
Beruht auf symmetrischen Blockchiffren (Blocklänge n), $\mathcal{K}$ Schlüsselraum für Blockchiffre
m zerlegen in Blöcke $m_0, \ldots, m_t$ der Länge n (evtl. Padding); A,B einigen sich auf Initialisierungsvektor IV der Länge n (oft Nullstring)
CBC-Mode: $E_k(c_i \oplus m_i)c_{i+1},\ c_0 = IV,\ H_k = c_{t+1}$

(b) HMAC (Bellare, Canetti, Krawczyk, 1996)
H Hashfunktion (z.B. SHA-1,-2,-3)
Zwei Konstanten $ipad$, $opad$ (Länge abh. von Schlüssellänge), k Schlüssel
HMAC von m zu k:

$$H(k \oplus opad | H(k \oplus ipad | m)), \ | \text{ Konkatenation}$$

Versandt werden kann ($E(m) = c,\ E$ Verschlüsselungsfunktion):

- $(c, MAC(c))$
- $(c, MAC(m))$
- $E((m, MAC(m)))$

5 Radomisierte Public-Key-Verschl. und Padding-Schemes

5.1 Meet-in-the-Middle-Angriff auf RSA (Boneh, Joax, Nguyen, 2000)

$(m_1 m_2)^e \mod n = m_1^e m_2^e \mod n$, Angreifer Eve weiß dann, dass $m < 2^t < n$ (z.B. m Passwort $\Rightarrow t$ klein)
Mit nicht vernachlässigbarer Wahrscheinlichkeit ist dann $m = m_1 \cdot m_2, m_1, m_2 \leq 2^{\frac{t}{2}}$ (z.B. $m < 2^{64}, t = 64, 18\%$ Wahrscheinlichkeit)
Eve erstellt $\left\{1^e, 2^e, \ldots, \left(2^{\frac{t}{2}}\right)^e\right\} \pmod n$ (oBdA teilerfremd zu n, sonst Teiler von n gefunden $\rightarrow$ Verfahren geknackt) und $\left\{(1^e)^{-1}, (2^e)^{-1}, \ldots, \left(\left(2^{\frac{t}{2}}\right)^e\right)^{-1}\right\} \pmod n$
Test: $c \cdot (k^e)^{-1} \mod n = j^e \mod n$
Falls ja: $c = k^e \cdot j^e \mod n = (\underbrace{kj}_{<n})^e \mod n \Rightarrow kj = m$
Angriff funktioniert, da RSA hier deterministisch verschlüsselt.

5.2 Optimal asymmetric encryption padding (OAEP) (Bellare, Rogaway, 1994)

Teil des PKCS#1
Kann für alle PK-Verfahren angewandt werden. Verschlüsselungsfunktion PK-Verfahren, Blocklänge l Bytes, Hashfunktion H, Länge der Hashwerte k Bytes ($k < l$)
Mask-Generating-Function$G : \mathbb{Z}_2^{8k} \rightarrow \mathbb{Z}_2^{8(l-k-1)}$, Länge $l - k - 1$ Bytes
Nachricht m, Länge $\leq l - k - 2$, $\tilde{m} = \overbrace{\underbrace{00|\ldots|00}_{optional}|01|m}^{\text{Hex-Darstellung}}$, Länge $l - k - 1$, Zufallsstring r, Länge k Bytes

Bilde und verschlüssle:

$$\overbrace{00|r \oplus H(\tilde{m} \oplus G(r))}^{Padding}\,|\,\overbrace{\tilde{m} \oplus G(r)}^{Maskierung} \rightarrow \text{Länge: } 1 + k + (l - k - 1) = l$$

Entschlüsselung

$s = \tilde{m} \oplus G(r)$, $t = r \oplus H(\tilde{m} \oplus G(r))$
$t \oplus H(s) = r \rightarrow G(r)$
$s \oplus G(r) \rightarrow \tilde{m} \rightarrow m$

Sicherheit

Man kann zeigen: Unter gewissen Voraussetzungen für H, G gilt:
Ein Angreifer kann nur dann ein Bit des Klartextes ermitteln, wenn er die RSA-Funktion invertieren kann ("Alles-oder-Nichts Prinzip").

Typische Werte (RSA)

$l = 2048 Bit = 256 Byte$
$k = 160 Bit = 20 Byte$
Länge von $m \leq 235 Byte \approx 92\%$ der Gesamtwortlänge

6 Authentifizierung und Zero-Knowledge-Beweise

Authentifizierung (Identifizierung):
Überprüfung, dass jemand derjenige ist, für den er sich ausgibt.
Forderung: Nur A darf sich als A ausgeben können. Auch darf sich nach erfolgter Authentifizierung gegenüber B nicht B als A ausgeben können.
Möglichkeiten: Authentifizierung durch:

- <u>Wissen</u>
- biometrische Merkmale
- Besitz

6.1 Passwörter

A wählt Passwort w, bei B ist $f(w)$ gespeichert, f Einwegfunktion (UNIX: Verschlüsselung des Nullstrings mit Schlüssel w)
Gefahren:

- unsichere Passwörter
- zu Beginn Übertragung des Passwortes über unsicheren Kanal (z.B. Key-Logger)

Verbesserungen:

- Einmal-Passwörter (TAN-Listen)
- Camport (1994): A wählt $n \in \mathbb{N}$, Wort w_A, B veröffentlicht Einwegfunktion f
  A berechnet $f(w_A), f^2(w_a), \ldots, f^n(w_a) =: w_0$, sendet w_0 an B. w_0 muss nicht verschlüsselt sein, B muss aber sicher sein, dass es von A kommt.

  Spätere Authentifizierung:
  A schickt $w_1 = f^{n-1}(w_A)$ an B, B testet ob $f(w_1) = w_0$, nächstes Mal $w_2 = f^{n-2}(w_A)$, Test $f(w_2) = w_1$ usw.

6.2 Challenge-Response-Verfahren

A erhält von B Aufgabe (Challenge), die nur sie mit ihrem geheimen Wissen erfüllen kann (Response). B muss die Antwort verifizieren können.
Einfachste Art:
Mit PK-Verfahren: Signatur einer Zufallszahl

Beispiel RSA

A (n, e) öffentlich, d geheim
$B \to A$ Zufallszahl z, A: $z^d \mod n \to B$
B verifiziert $\left(z^d \mod n\right)^e = z$
<u>Gefahr:</u> Angenommen C hat Nachricht m mit A's Schlüssel (n, e) verschlüsselt, B hat $m^e \mod n$ abgefangen. Falls er bei Authentifizierung A $z = m^e \mod n$ als Zufallszahl sendet, erhält er m zur Nachricht C$\to$ A
$\Rightarrow$ verschiedene Schlüssel für Authentifizierung und Verschlüsselung !

6.3 Zero-Knowledge-Verfahren

Konzept "Zero-Knowledge-Proof" (Goldwasser, Micali, Rackoff, 1985/89)

Beruhen auf interaktiven Beweissystemen (hier) (nicht-interaktiv: Blum, Feldmann, Micali)
Protokoll zwischen A und B. A kennt Geheimnis, B nicht.
In mehrfach durchzuführender Schleife (Interaktion) wird Kommunikation durchgeführt, die 2 Kriterien erfüllt:

- A überzeugt B, dass sie dasGeheimnis kennt
- Betrüger, der Geheimnis nicht kennt, kann B nicht überzeugen

(Erfüllt durch Challenge-Response-Verfahren)
Zusätzlich bei <u>Zero-Knowledge-Eigenschaft</u>:

- B erfährt nur, dass A das Geheimnis kennt, aber nichts weiter (egal, welche Strategie er verfolgt)

6.3.1 Fiat-Shamir-Verfahren (1985)

A (Beweiserin) wählt zwei große Primzahlen $p, q, p \neq q$, sie bildet $n = p \cdot q$. Sie wählt zufällig $s \in \{1, \ldots, n-1\}$ mit $ggT(s, n) = 1$. A berechnet $v = s^2 \mod n$ (v, n) öffentlich, s geheim (nach 3.4.1 ist s aus (v, n) zu bestimmen "genauso schwierig" wie die Faktorisierung von n)
Authentifizierung: A beweist B, dass sie die Quadratwurzel von $v \mod n$ kennt.

Protokoll

1. A wählt zufällig gleichverteilt $r \in \{1, \ldots, n-1\}, ggT(n, r) = 1$. Sie berechnet $x = r^2 \mod n$
2. A sendet x an Verifizierer B
3. B wählt mit W-keit $\frac{1}{2}$ $e \in \{0, 1\}$ und sendet e an A (Challenge)

4e $= 0$: A sendet r an B (Response)

$e = 1$: A sendet $y = r \cdot s \mod n$ an B

5 $e = 0$: B verifiziert $r^2 \mod n = x$

$e = 1$: B verifiziert $y^2 \mod n = x \cdot v \mod n$

Schritte 1-5 werden wiederholt (jedes Mal mit neuem r)

Analyse

(1) Alice kennt s und kann daher alle Challenges beantworten.

(2) Betrüger M kann aus (n, v) nicht effizient s berechnen.

(3) Was passiert, wenn M versucht, sich als A auszugeben?
Er kann nicht beide möglichen Fragen beantworten, denn sonst kennt er r und $y = rs \mod n$ und könnte s berechnen. ↯
Er hat zwei Möglichkeiten:

- er verhält sich gemäß Protokoll
  $\Rightarrow$ erfolgreich bei $e = 0$, nicht bei $e = 1$
- er wählt y und sendet $x = y^2 v^{-1} \mod n$ an B
  $\Rightarrow$ erfolgreich bei $e = 1$, nicht bei $e = 0$

Da M x angeben muss, bevor er e erhält, ist seine Erfolgswahrscheinlichkeit $\frac{1}{2}$. Bei k Durchgängen also $\frac{1}{2^k}$

(4) B erfährt nichts über das Geheimnis s von A (außer, dass A es kennt) - gleichgültig, ob er sich an das Protokoll hält oder nicht (intuitiv: Wenn r zufällig gleichverteilt ist, so auch $y = r \cdot s \mod n$)

$\Rightarrow$ Zero-Knowledge-Beweise sind für alle NP-Probleme möglich

7 Secret Sharing Schemes

Aufteilung von Geheimnissen

7.1 Definition

T Menge von Teilnehmern.
Zugriffsstruktur Z auf T ist Menge von Teilmengen von T, d.a. $Z \subseteq \mathcal{P}(T)$. $X \in Z$ heißt zulässige Konstellation. $Y \in \mathcal{P}(T) \setminus Z$ heißt unzulässige Konstellation

7.2 Definition

Secret Sharing Scheme (SSS) zu T, Z (wie oben), g Geheimnis (aus einer Menge potentiell möglicher Geheimnisse)
In SSSs wird durch eine vertrauenswürdige Instanz (Dealer) an jeden Teilnehmer Elemente aus einer Menge S (Teilgeheimnisse, shares) verteilt, so dass gilt:

(1) Jede zulässige Konstellation kann mit Hilfe ihrer Teilgeheimnisse (mit einem bekannten Algorithmus) das Geheimnis rekonstruieren

(2) Jede unzulässige Konstellation, kann es nicht

SSS heißt perfekt, falls jede unzulässige Konstellation mit ihren Teilgeheimnissen genauso viel über g erfährt, wie ohne ihre Teilgeheimnisse.

7.3 Spezialfall: Schwellenwertsysteme

7.3.1 Definition

$T = \{1, \ldots, n\}, k \leq n, Z = \{U \subseteq T : |U| \geq k\}$
SSS zu einer solchen Zugriffsstruktur heißt Schwellenwertsystem (threshold system)

7.3.2 Shamirs Konstruktion eines Schwellenwertsystems (1979)

$|T| = n, 2 \leq k \leq n$
geg.: große Primzahl p (Größe bestimmt Sicherheitsniveau), $n + lep$, $g \in \mathbb{Z}_p$ Geheimnis

Dealer

Wählt zufällig Zahlen $a_1, \dots, a_{k-1} \in \mathbb{Z}_p, a_{k-1} \neq 0$ und bildet $f(x) = g + a_1 x + \dots + a_{k-1}x^{k-1} \in \mathbb{Z}_p[x]$ (a_i, g geheim)
Wählt $x_1, \dots, x_n \in \mathbb{Z}_p \setminus \{0\}, x_i \neq x_j$ (öffentlich)
Teilnehmer i erhält $(x_i, g_i = f(x_i))$

Rekonstruktion

Ang. k Teilnehmer, oBdA $1, \dots, k$
f ist durch $(x_1, g_1), \dots, (x_k, g_k)$ eindeutig bestimmt, z.B. mit Lagrange-Interpolation:

$$f(x) = \sum_{j=1}^{k} g_j \frac{\prod\limits_{\substack{i=1\\i\neq j}}^{k} (x - x_i)}{\prod\limits_{\substack{i=1\\i\neq j}}^{k} (x_j - x_i)}$$

Mehr als k Teilnehmer erzeugen dasselbe Polynom, $k' < k$ Teilnehmer interpolieren Polynom f' vom Grad $\leq k' - 1$. Jedes $f'(0)$ ist gleich wahrscheinlich $\Rightarrow$ perfektes SSS

7.4 Monotone SSS

7.4.1 Definition

$Z \subseteq \mathcal{P}(T)$ heißt monotone Zugriffsstruktur, wenn gilt:
$X \in Z, X \subseteq U \subseteq T \Rightarrow U \in Z$
(Schwellenwertsysteme sind monoton)

7.4.2 Monotone SSS nach Simmons

Lemma

Sei Y d-dimensionaler Vektorraum, $v_1, \dots, v_d$ Basis von Y. Dann existiert $v_{d+1} \in Y$, so dass $v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_{d+1}$ Basis von Y ist für alle $i \in \{1, \dots, d+1\}$

Beweis

z.B. $v_{d+1} = v_1 + \dots + v_d$
$i \leq d : v_i \in \langle v_1, \dots, v_{i-1}, v_{i+1} \rangle = Y$

Konstruktion

(1) $\mathcal{U}_{max} = \{U \in \mathcal{U} : U \subsetneq V \subseteq T \Rightarrow V \in Z\}$, wobei $\mathcal{U} = \mathcal{P} \setminus Z$ $|\mathcal{U}_{max}| = d$, $\mathcal{U}_{max} = \{U_1, \dots, U_d\}$, p Primzahl (groß), $V = \mathbb{Z}_p^{d+1}$
Wähle in V:

- 2-dim Unterraum X, das ist der öffentlich bekannte Geheimnisraum
- In X 1-dim Unterraum $G =< w >$, das Geheimnis (in Y gibt es $\frac{p^2-1}{p-1}$ 1-dim Unterräume)
- d-dim Unterraum Y von V mit $Y \cap X = G$ (Wähle Basis von X: w, x_1; Ergänze zu Basis von $V: w, x_1, y_1, ..., y_{d-1}; x_1 \notin Y =< w, y_1, \ldots, y_{d-1} >$)
- Ergänze $w, x_1, y_1, ..., y_{d-1}$ durch $y_d \in Y$, so dass ja d Vektoren aus $w, x_1, y_1, ..., y_d$ linear unabhängig sind, d.h. Basis von Y (vgl. Lemma)

(2) Verteilung der Teilgeheimnisse:
$t \in T = \{t_1, \ldots, t_n\}$: Sei $t \notin U_{i_1}, \ldots U_{i_r}, t \in U_j \forall j \in \{1, \ldots, d\} \setminus \{i_1, \ldots, i_r\}$.
Dann erhält t als Teilgeheimnismenge $\{Y_{i_1}, \ldots, Y_{i_r}\}$

(3) Rekonstruktion:
$S \subseteq T$. Die Teilnehmer aus S bilden den Unterraum, der von all ihren Vektoren erzeugt wird und schneiden diesen mit X

- 1. Fall: $S \in Z$
  Dann gilt $S \not\subseteq U_i \forall i \in \{1, \ldots, d\}$ (da Zugriffsstruktur monoton). Sei $i \in \{1, \ldots, d\}$. Dann existieren Teilnehmer $t \in S$ mit $t \notin U_i$ (abh. von i). Also hat t in seiner Teilgeheimnismenge den Vektor y_i. In der Vereinigung aller Teilmengen aus S kommen $y_1, \ldots y_d$ vor.. TN aus S erzeugen mit ihren Vektoren ganz Y, $Y \cap X = G$
- 2. Fall: $S \notin Z$, d.h. $S \in U$
  Dann existiert maximal unzulässige Teilmenge U_j mit $S \subseteq U_j$.
  Kein Teilnehmer aus S hat den Vektor y_j erhalten. Teilnehmer aus S erzeugen Unterraum $Y_0 \subseteq< y_1, \ldots, y_{j-1}, y_j, \ldots, y_d >$. $Y_0 \cap X \subseteq Y \cap X =< w >$.
  Angenommen $Y_0 \cap X =< w >$:
  $\Rightarrow w \in Y_0 \subseteq< y_1, \ldots, y_{j-1}, y_j, \ldots, y_d >$
  $\Rightarrow w, y_1, \ldots, y_{j-1}, y_j, \ldots, y_d$ sind lin. abh. $\lightning$ zu (1)
  $\Rightarrow Y_0 \cap X = \{0\}$. Sie erfahren nichts über das Geheimnis $\Rightarrow$ perfektes SSS.