Kryptologie II

Sommersemester 2008

Persönliche Mitschrift von
Yves Radunz

Inhaltsverzeichnis

Vorlesung am 15.04.2008

Kapitel 1

Hash-Verfahren

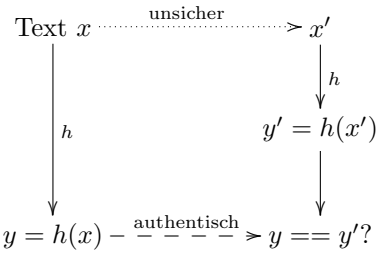
1.1 Einführung

Schutzziele

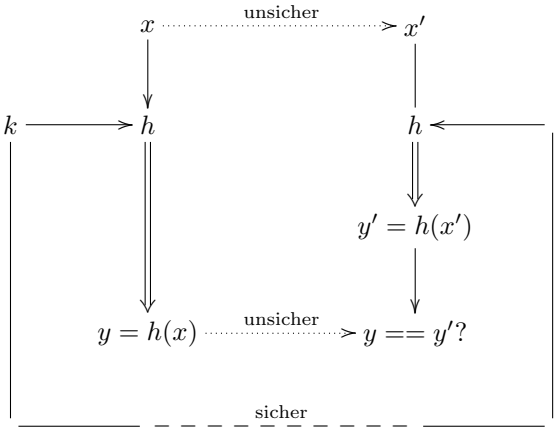
1. Vertraulichkeit
   - Geheimhaltung
   - Anonymität
   - Unbeobachtbarkeit
2. Integrität
   - von Nachrichten und/oder Daten
3. Zurechenbarkeit
   - Authentikation
   - Unabstreitbarkeit
   - Identifikation
4. Verfügbarkeit
   - von Daten
   - von Rechenressourcen
   - von Informationsdienstleistungen

Klassifikation von Hashverfahren

- schlüsselloses Hashverfahren (MDC: Manipulation Detection Code)



- symmetrischer Schlüssel (MAC: Message Authentication Code)



- Bei asymmetrischen Schlüsseln sprechen wir nicht von Hashverfahren, sondern von digitalen Signaturen.

Definition (gültiges Paar, Kollisionspaar)
Sei $h : X \to Y$ eine Hashfunktion.
Ein Paar $(x, y) \in X \times Y$ heißt *gültig*, falls $y = h(x)$ gilt.
Ein Paar $(x, x') \in X \times X$ mit $x \neq x'$ und $h(x) = h(x')$ heißt *Kollisionspaar*.
Die Anzahl $|Y|$ der Hashwerte bezeichnen wir mit m. Ist auch X endlich, d.h. $n = |X| < \infty$, so bezeichnen wir h als eine (n, m)-Hashfunktion. In diesem Fall verlangen wir meist, dass $m \leq \frac{n}{2}$ ist. h wird dann *Kompressionsfunktion* genannt.

Problem 1, Einweg-Hashfunktion
Gegeben seien $h : X \to Y$ und $y \in Y$. Gesucht ist $x \in X : h(x) = y$.
Falls es einen immensen Aufwand erfordert, dieses Problem zu lösen, so heißt h *Einweg-Hashfunktion* (auch *one-way hashfunction* oder *preimage resistant hashfunction*).

Problem 2, schwache Kollisionsresistenz
Gegeben seien $h : X \to Y$ und $x \in X$. Gesucht ist $x' \in X : h(x) = h(x'), x \neq x'$.
Falls es einen immensen Aufwand erfordert, dieses Problem zu lösen, so heißt h *schwach kollisionsresistent* (auch *weakly collision resistant* oder *second preimage resistant*).

Problem 3
Gegeben sei $h : X \to Y$. Gesucht sind $x, x' \in X$ mit $h(x) = h(x')$ und $x \neq x'$.
Falls es einen immensen Aufwand erfordert, dieses Problem zu lösen, so heißt h *stark kollisionsresistent*. Wir bezeichnen derartige Hashfunktionen auch einfach nur als *kollisionsresistent*.

Einweg-Hashfunktion ⇒ schwach kollisionsresistent?
Einweg-Hashfunktionen müssen nicht schwach kollisionsresistent sein:
Sei h eine Einweg-Hashfunktion. Dann ist auch $h'(xb) = h(x), b \in 2$ eine Einweg-Hashfunktion, aber nicht schwach kollisionsresistent, denn: $h'(x0) = h(x) = h'(x1)$ und $x = h'^{-1}(y)\text{div}2$.

Schwache Kollisionsresistenz ⇒ Einweg-Hashfunktion?
Schwach kollisionsresistente Hashfunktionen müssen keine Einweg-Hashfunktionen sein:
Sei h eine schwach kollisionsresistente Hashfunktion. Dann ist $h'(xb) = \begin{cases} x0 & b = 0 \\ h(x)1 & b = 1 \end{cases}$ eine schwach kollisionsresistente Hashfunktion, welche jedoch mit Wahrscheinlichkeit $\frac{1}{2}$ eine leichte Berechnung eines Urbildes zulässt.

Satz 1.1. (Starke Kollisionsresistenz ⇒ schwache Kollisionsresistenz) *Sei $h : X \to Y$ eine stark kollisionsresistente (n, m)-Hashfunktion. Dann ist h auch schwach kollisionsresistent.*

Beweis:
Wir nehmen an, h wäre nicht schwach kollisionsresistent. Sei A ein probabilistischer Algorithmus, der für einen zufällig gewählten Text $x \in X$ mit Erfolgswahrscheinlichkeit $\varepsilon > 0$ ein zweites Urbild zu $h(x)$ liefert. Dann findet der folgende Algorithmus B mit der selben Wahrscheinlichkeit ε ein Kollisionspaar:

```
X^2 B(h) {
    x' = A(x);                                   P(x ≠ x') = ε
    if (x' ≠ x)
        return (x, x');
    return ?;
}
```

Dies stellt einen Widerspruch zu der (starken) Kollisionsresistenz von h dar. □

Satz 1.2. (Starke Kollsisionsresistenz ⇒ Einwegfunktion) *Sei $h : X \to Y$ eine stark kollisionsresistente (n, m)-Hashfunktion. Dann ist h auch eine Einweg-Hashfunktion.*

Beweis:
Annahme: h ist keine Einweg-Hashfunktion. Sei A ein Invertierungsalgorithmus für h. Wir definieren nun einen Algorithmus B durch:

```
X^2 B(h) {
    x = select_random(X);
    y = h(x);
    x' = A(x);
    if (x ≠ x')
        return (x, x');
    return ?;
}
```

Behauptung: $\mathbf{P}(B = (x, x')) \geq \frac{1}{2}$.
Sei $C = \{h^{-1}(y) | y \in h(X)\}$.

$$\begin{aligned}\Rightarrow \mathbf{P}(B = (x, x')) &= \sum_{c \in C} \frac{|C|}{|X|} \cdot \frac{|C|-1}{|C|} = \sum_{c \in C} \frac{1}{n} \cdot (|C| - 1) \\ &= \frac{1}{n} \sum_{c \in C} |C| - 1 = \frac{1}{n}(n - |C|) = 1 - \frac{1}{n}|Y| = 1 - \frac{m}{n} \\ &\geq 1 - \frac{m}{2m} = \frac{1}{2}\end{aligned}$$

□

Bemerkung
Aus den vorhergehenden Sätzen folgt auch, dass es schwach kollisionsresistente Hashfunktionen gibt, die nicht stark kollisionsresistent sind. Analog gibt es Einweg-Hashfunktionen, die nicht (stark) kollisionsresistent sind.

Vorlesung am 17.04.2008

1.2 Zufallsorakelmodell - ZOM

Definition
Zufallsorakelmodell Der Gegner wird mit einer zufällig aus der Klasse Y^X gewählten Hashfunktion konfrontiert, d.h. die einzige Möglichkeit, Informationen über h zu gewinnen, besteht darin, für bestimmte Texte x die zugehörigen Hashwerte $h(x)$ zu erfragen.

Proposition 1.1. *Sei $X_0 = \{x_1, ..., x_k\} \subseteq X$ und seien $y_1, ..., y_k \in Y$. Dann gilt für eine zufällig aus $F(X, Y) = Y^X$ gewählten Funktion h und für jedes Paar $(x, y) \in (X \setminus X_0) \times Y$:*
$P(h(x) = y | h(x_i) = y_i, i = 1, ..., k) = \frac{1}{m}$.

Algorithmus: FindPreImage
Gegeben seien eine Hashfunktion h, das Bild y und eine maximale Anzahl an Fragen q.
Der Algorithmus A wählt zufällig $x_1, ..., x_q$ und berechnet $h(x_1), ..., h(x_q)$.
Dann finden A mit einer Wahrscheinlichkeit von $1 - (\frac{m-1}{m})^q \approx \frac{q}{m}$ ein Urbild von y. Es ist also $q \approx m\mathbf{P} = O(m)$ erforderlich, wobei $\mathbf{P}$ die gewünschte Erfolgswahrscheinlichkeit ist.

Analog erhalten wir derartige Wahrscheinlichkeiten für die Probleme 2 und 3 (Kollisionssuche).
Für das dritte Problem erhalten wir die Wahrscheinlichkeit $\mathbf{P} = 1 - \frac{(m-1)!}{(m-q)!m^{q-1}}$ durch den *Geburtstagsangriff*. Mit $1 - x \approx e^{-x}$ erhalten wir die Näherung $\mathbf{P} \approx \frac{q(q-1)}{2m} \approx \frac{q^2}{2m}$.
Wir erhalten damit $q \approx \sqrt{2m\mathbf{P}} = \sqrt{2\mathbf{P}}\sqrt{m} = O(\sqrt{m})$.
Demnach sollte man eine Hashwertlänge von etwa 128 (oder besser: 160) Bit verwenden.

Bemerkung (Geburtstagsangriff)
Der Name Geburtstagsangriff stammt von dem Geburtstagsparadoxon, welches nach der Anzahl der Personen fragt, ab der mit Wahrscheinlichkeit $\frac{1}{2}$ mindestens zwei von ihnen am gleichen Tag des Jahres Geburtstag haben.

Erweiterung einer Hashfunktion auf Texte beliebiger Länge
Gegeben sei eine Kompressionsfunktion $h : 2^{m+t} \to 2^m, t \geq 1$.
Wir suchen eine Erweiterung von h auf $\hat{h} : 2^* \to 2^m$.

Vorlesung am 22.04.2008

Beweis:
Annahme: Für $\hat{h}$ kann ein Kollisionspaar $(x, \tilde{x})$ gefunden werden, d.h. $x \neq \tilde{x}, \hat{h}(x) = \hat{h}(\tilde{x})$.
Wir zeigen, dass sich hieraus ein Kollisionspaar für h finden lässt.
Sei $y(x) = y_1y_2...y_k$ und $y(\tilde{x}) = \tilde{y}_1\tilde{y}_2...\tilde{y}_l$ mit $k \leq l$.

Vorlesung am 24.02.2008

Wir haben $x = x_1...x_k$ mit $|x_1| = ... = |x_{k-1}| = t - 1$ und $1 \leq d = |x_k| \leq t - 1$.
Im Fall $k = 0$ haben wir $y(\varepsilon) = 0^t$.
Im Fall $k = 1$ sei $y(x) = 0x0^{t-1-|x|}1(t - 1 - |x|)_{2,t-1}$.
Für $k > 1$ sei $y(x) = y_1...y_{k+1}$ mit $y_1 = 0x_1$, $y_i = 1x_i$ für $i = 2, ..., k - 1$, $y_k = 1x_k0^{t-d-1}$ und $y_{k+1} = 1(t - d - 1)_{2,t-1}$.

Satz 1.3. *Die oben definierte Funktion y ist suffixfrei.*

Beweis:
Seien $x \neq \tilde{x}$ zwei Texte mit $|x| \leq |\tilde{x}|$.
Dann gilt $|y(x)| \leq |y(\tilde{x})|$, d.h. wir müssen noch zeigen, dass $y(x) = y_1...y_{k+1}$ kein Suffix von $y(\tilde{x}) = \tilde{y}_1...\tilde{y}_{l+1}$ ist.
Im Fall $x = \varepsilon$ ist dies klar. Für $x \neq \varepsilon$ betrachten wir folgende Fälle:

1. $|x| \not\equiv_{t-1} |\tilde{x}|$

   Hier gilt $d \neq \tilde{d}$, d.h. $y_{k+1} \neq \tilde{y}_{l+1}$.

2. $|x| = |\tilde{x}|$

   Hier erhalten wir $|y(x)| = |y(\tilde{x})|$, d.h. $k = l$. Wegen $x \neq \tilde{x}$ gibt es ein $i \in \{1, ..., k\}$ mit $x_i \neq \tilde{x}_i$. Damit folgt jedoch auch $y_i \neq \tilde{y}_i$, d.h. $y(x) \neq y(\tilde{x})$.

3. $|x| \neq |\tilde{x}|, |x| \equiv_{t-1} |\tilde{x}|$

   Hier folgt $k < l$.

   Weiterhin gibt es in $y(x)$ genau einen Block, der mit 0 beginnt. Dieser steht bei $y(\tilde{x})$ ganz vorne, d.h. es ist $\tilde{y}_1$. Somit müsste $k = l$ gelten, damit $y(x)$ ein Suffix von $y(\tilde{x})$ sein kann. Dies stellt jedoch einen Widerspruch zu $k < l$ dar. □

1.3 Die MD und SHA-Hashfamilien

Hashfukt.	MD4	MD5	$SHA-1$	$SHA-224$	$SHA-256$	$SHA-384$	$SHA-512$
Bitlänge	128	128	160	224	256	384	512
Wortlänge	4	4	5	9	10	14	20

Benutzte Funktionen
Die Funktionen sind auf Worten, d.h. auf Mengen $2^{32} \times ... \times 2^{32}$ ($2 = \{0, 1\}$ wie in der Mengenlehre).

1. $X \wedge Y$ bitweises UND
2. $X \vee Y$ bitweises ODER
3. $X \oplus Y$ bitweises XOR
4. $\neg X$ bitweises NOT
5. $X + Y$ Ganzzahl-Addition ohne Übertrag (modulo 2^{32})
6. $X \hookleftarrow s$ zirkulärer Linksshift um s Stellen (ROTATE LEFT, ROL)

Bemerkung (little/big endian)
Es gibt zwei Möglichkeiten, die einzelnen Bytes einer großen Zahl im Speicher abzulegen:

1. *little endian*: $a_0a_1a_2a_3$ repräsentiert die Zahl $a_3 2^{24} + a_2 2^{16} + a_1 2^8 + a_0$
2. *big endian*: $a_0a_1a_2a_3$ repräsentiert die Zahl $a_0 2^{24} + a_1 2^{16} + a_2 2^8 + a_3$

MD4 und MD5 verwenden bei der Addition die little endian Darstellung.

Definition (Hashfamilie)
Eine *Hashfamilie* $\mathcal{H}$ wird durch folgende Komponenten beschrieben:

1. X: eine (un)endliche Menge von Texten
2. Y: endliche Menge aller möglichen Hashwerte, $|Y| \leq |X|$
3. K: endlicher Schlüsselraum, wobei jeder Schlüssel $k \in K$ eine Hashfunktion $h_k : X \to Y$ spezifiziert.

Vereinbarung von Symbolen und Variablen
Wir verwenden $n = |X|$, $m = |Y|$ und $l = |K|$. $\mathcal{H}$ heißt dann (n, m, l)-*Hashfamilie*.

Definition (Berechnungsresistenz)
Eine Hashfamilie heißt *berechnungsresistent*, wenn sie die folgende Eigenschaft hat:
Selbst wenn eine Reihe $(x_i, h(x_i))_{i=1,...,r}$ von (unter demselben Schlüssel k) generierten Text- und Hashwert-Paaren bekannt ist, so erfordert es einen sehr großen Aufwand, ein Paar $(x, h(x))$ mit $x \notin \{x_1, ..., x_r\}$ zu finden.

Vorlesung am 29.02.2008

1.4 MACs

Angriffe gegen symmetrische Hashfunktionen

1. *Impersonation*

   Der Gegner kennt nur den benutzten MAC und versucht ein gültiges Paar, das als echt akzeptiert wird, d.h. $y = h_k(x)$, zu erzeugen.

2. *Substitution*

   Der Gegner kennt bereits ein Paar (x, y) mit $y = h_k(x)$ und versucht ein neues Paar (x', y') mit $y' = h_k(x'), x' \neq x$ zu finden.

3. Angriff bei bekanntem Text

   In diesem Fall kennt der Gegner sogar mehrere Paare, aus denen er versucht ein neues Paar zu erzeugen (siehe Berechnungsresistenz).

4. Angriff bei frei wählbarem Text

   Hier kann der Gegner die Wahl der Texte $x_1, ..., x_n$ selbst bestimmen.

5. Angriff bei *adaptiv wählbarem Text*.

   Der Gegner kann die Wahl des Textes x_i von den MAC- bzw. Hashwerten $h_k(x_1), ..., h_k(x_{i-1})$ abhängig machen.

Informationstheoretische Sicherheit von MACs
Modell:
Der Schlüssel k und der Text x werden gemäß der Wahrscheinlichkeitsverteilung $p(k, x) = p(k)p(x)$ gewählt, welche dem Gegner bekannt ist. O.B.d.A. gilt $p(k) > 0 < p(x)$ für alle $k \in K, x \in X$.

Erfolgswahrscheinlichkeit für Impersonation
Bezeichne $\alpha(x, y)$ die Erfolgswahrscheinlichkeit des Gegners, falls er das Paar (x, y) wählt.
Dann gilt $\alpha(x, y) = \sum_{k \in \mathcal{K}(x,y)} p(k)$ mit $\mathcal{K}(x, y) = \{k \in K | h_k(x) = y\}$.

Beispiel (Impersonation)
Seien $K = \{1, 2, 3\}, X = \{a, b, c, d\}$ und $Y = \{0, 1\}$.

Authentikationsmatrix

		$0,1$	$0,2$	$0,3$	$0,4$
		a	b	c	d
$0,25$	1	0	0	0	1
$0,3$	2	1	1	0	1
$0,45$	3	0	1	1	0

Erfolgswahrscheinlichkeit

α	a	b	c	d
0	$0,7$	$0,25$	$0,55$	$0,45$
1	$0,3$	$0,75$	$0,45$	$0,55$

Satz 1.4. (1) *Für jede (n, m, l)-Hashfunktion gilt $\alpha \geq \frac{1}{m}$.*

Beweis:
Sei $x \in X$ beliebig.
Dann gilt $\sum_{y \in Y} p(x \mapsto y) = 1$, d.h. es existiert ein $y \in Y$ mit $\alpha(x, y) = p(x \mapsto y) \geq \frac{1}{m}$. □

Bemerkung
Es gilt genau dann $\alpha = \frac{1}{m}$, wenn alle Summanden gleich sind, d.h. $p(x \mapsto y) = \sum_{k \in \mathcal{K}(x,y)} p(k) = \frac{1}{m}$.
Bei Gleichverteilung der Schlüssel muss demnach in jeder Spalte der Authentikationsmatrix jeder Hashwert gleich oft vorkommen.

Erfolgswahrscheinlichkeit für Substitution
Angenommen, B sendet das Paar (x, y) an A (d.h. $p(x \mapsto y) > 0$) und der Gegner ersetzt es durch (x', y') mit $x' \neq x$.
Dann ist die Erfolgswahrscheinlichkeit des Gegners $\beta(x, y, x', y') = P(x' \mapsto y' | x \mapsto y)$.
$\Rightarrow \beta(x, y, x', y') = \frac{P(x' \mapsto y' \wedge x \mapsto y)}{P(x \mapsto y)} = \frac{\sum_{k \in \mathcal{K}(x,y) \wedge \mathcal{K}(x',y')} p(k)}{\sum_{k \in \mathcal{K}(x,y)} p(k)}$
Falls B das Paar (x, y) sendet, kann der Gegner bei Verwendung der optimalen Strategie die Erfolgswahrscheinlichkeit $\beta(x, y) = \max_{(x',y'), x \neq x'} \beta(x, y, x', y)$ erreichen.
Da der Gegner keinen Einfluss auf die Wahl des Paares (x, y) hat, ist seine Erfolgswahrscheinlichkeit bei optimaler Strategie
$\beta = \sum_{(x,y)} p(x, y)\beta(x, y) = \sum_{(x,y)} p(x)p(y|x)\beta(x, y) = \sum_{(x,y)} p(x)p(x \mapsto y)\beta(x, y)$
$= \sum_{(x,y)} p(x) \underbrace{\max_{(x',y'), x' \neq x} p(x' \mapsto y' \wedge x \mapsto y)}_{=\beta'(x,y)}$

Beispiel

	$(a,0)$	$(a,1)$	$(b,0)$	$(b,1)$	$(c,0)$	$(c,1)$	$(d,0)$	$(d,1)$	$\beta'(x,y)$	$p(x)$	$\alpha(x,y)$	$\beta(x,y)$
$(a,0)$	–	–	0,25	0,45	0,25	0,45	0,45	0,25	0,45	0,1	0,7	0,643
$(a,1)$	–	–	0	0,3	0,3	0	0	0,3	0,3	0,1	0,3	1
$(b,0)$	0,25	0	–	–	0,25	0	0	0,25	0,25	0,2	0,25	1
$(b,1)$	0,45	0,3	–	–	0,3	0,45	0,45	0,3	0,45	0,2	0,75	0,6
$(c,0)$	0,25	0,3	0,25	0,3	–	–	0	0,55	0,55	0,3	0,55	1
$(c,1)$	0,45	0	0	0,45	–	–	0,45	0	0,45	0,3	0,45	1
$(d,0)$	0,45	0	0	0,45	0	0,45	–	–	0,45	0,4	0,45	1
$(d,1)$	0,25	0,3	0,25	0,3	0,55	0	–	–	0,55	0,4	0,55	1

$\Rightarrow \beta = 0,915$

Vorlesung am 06.05.2008

Satz 1.5. **($\forall x \neq x' \in X, y \in Y \exists y' \in Y : \beta(x, y, x', y') \geq \frac{1}{m}$ (2))** *Für alle $x, x' \in X$ und $y \in Y$ existiert ein $y' \in Y$ mit $\beta(x, y, x', y') \geq \frac{1}{m}$.*

Beweis:
Wegen $\sum_{y' \in Y} p(x' \mapsto y' | x \mapsto y) = 1$ muss ein $y' \in Y$ mit $p(x' \mapsto y' | x \mapsto y) \geq \frac{1}{m}$ existieren. □

Korollar 1.1. *Für alle (x, y) mit $p(x \mapsto y) > 0$ gilt $\beta(x, y) \geq \frac{1}{m}$.*
Insbesondere gilt $\beta = E(\beta(x, y)) \geq E(\frac{1}{m}) = \frac{1}{m}$.

Lemma 1.1. *Sei (X, Y, K, H) ein MAC mit $\beta = \frac{1}{m}$.*
Dann gilt $p(x' \mapsto y' | x \mapsto y) = \frac{1}{m}$ für alle $x, x' \in X, y, y' \in Y$. Insbesondere gilt $p(x \mapsto y) > 0$ für alle $x \in X, y \in Y$.

Beweis:

1. $\forall x \neq x' \in X, y, y' \in Y, p(x \mapsto y) > 0 : p(x' \mapsto y' | x \mapsto y) \leq \frac{1}{m}$

   (Folgt direkt aus der Definition von β.)

2. $\forall x \in X, y \in Y : p(x \mapsto y) > 0$

   Annahme: $\exists x', y' : p(x' \mapsto y') = 0$.

   Seien dann x, y ein beliebiges Paar mit $p(x \mapsto y) > 0$. Dann gilt $p(x' \mapsto y' | x \mapsto y) = 0$. Wegen $\sum_{y'' \in Y \setminus \{y'\}} p(x' \mapsto y'' | x \mapsto y) = 1$ existiert ein $y'' \in Y$ mit $p(x' \mapsto y'' | x \mapsto y) \geq \frac{1}{m-1} > \frac{1}{m}$, was einen Widerspruch zu 1. darstellt.

3. $\forall x \neq x' \in X, y, y' \in Y : p(x' \mapsto y' | x \mapsto y) \geq \frac{1}{m}$

   Annahme: Es existieren $x \neq x', y, y'$ mit $p(x' \mapsto y' | x \mapsto y) < \frac{1}{m}$.

   Wegen $\sum_{y' \in Y} (x' \mapsto y' | x \mapsto y) = 1$ muss ein y'' mit $(x' \mapsto y'' | x \mapsto y) > \frac{1}{m}$ existieren. Dies ist ein Widerspruch zu 1. □

Satz 1.6. *Ein MAC (X, Y, K, H) erfüllt genau dann $\beta = \frac{1}{m}$, wenn $p(x \mapsto y, x' \mapsto y') = \frac{1}{m^2}$ für alle $x \neq x', y, y'$ gilt.*

Beweis:

1. "⇒"

   Nach obigem Lemma gilt $p(x \mapsto y) > 0$ und $p(x' \mapsto y' | x \mapsto y) = \frac{1}{m}$ für alle $x \neq x', y, y'$. Daher ist für alle $x \neq x'$:

   $p(x' \mapsto y') = \sum_{y \in Y} p(x' \mapsto y', x \mapsto y) = \sum_{x,y} p(x \mapsto y) p(x' \mapsto y' | x \mapsto y)$
   $= \frac{1}{m} \sum_{y \in Y} p(x \mapsto y) = \frac{1}{m}$

   Folglich gilt $p(x' \mapsto y', x \mapsto y) = p(x' \mapsto y') \cdot p(x \mapsto y | x' \mapsto y') = \frac{1}{m^2}$.

2. "⇐"

   Es gilt $\beta'(x, y) = \max_{x' \neq x, y'} p(x' \mapsto y', x \mapsto y) = \frac{1}{m^2}$ und daher
   $\beta = \sum_{x \in X, y \in Y} p(x) \beta'(x, y) = \frac{1}{m^2} \sum_{x,y} p(x) = \frac{1}{m^2} \cdot m \cdot 1 = \frac{1}{m}$. □

Bemerkung
Falls der Schlüssel unter Gleichverteilung gewählt wird, ist ein MAC genau dann optimal (bezüglich Substitutionsangriffen), wenn der zugehörigen Authentikationsmatrix in je zwei Spalten $x \neq x'$ jedes Hashwertpaar (y, y') gleich oft vorkommt.

Definition (stark universal)
Ein MAC (X, Y, K, H) heißt *stark universal*, falls für alle $x \neq x', y, y'$ gilt:

$$|K(x, y) \cap K(x', y')| = \lambda = \frac{|K|}{m^2}.$$

Wir nennen (X, Y, K, H) dann auch einen (n, m, λ)-MAC mit $l = \lambda m^2$.
Bei $\lambda = 1$ ist nur $n \leq m + 1$ möglich, d.h. in diesem Fall haben wir nur eine sehr geringe Kompression.

Vorlesung am 08.05.2008

Beispiel ($(4, 3, 1)$-MAC)

	0	1	2	3
0	0	0	0	0
1	1	1	1	0
2	2	2	2	0
3	0	1	2	1
4	1	2	0	1
5	2	0	1	1
6	0	2	1	2
7	1	0	2	2
8	2	1	0	2

Satz 1.7. *Seien $n = m = p$ prim und $\mathcal{H} = \{h_{a,b} | a, b \in \mathbb{Z}_p\}$ die Familie der Hashfunktionen $h_{a,b}(x) = ax + b \mod p$.*
Dann ist $(\mathbb{Z}_p, \mathbb{Z}_p, \mathbb{Z}_p^2, \mathcal{H})$ ein $(p, p, 1)$-Mac.

Beweis:
Für alle $x \neq x' \in \mathbb{Z}_p$ und $y, y' \in \mathbb{Z}_p$ existiert genau ein $(a,b) \in \mathbb{Z}_p^2$ mit $h_{a,b}(x) = y$ und $h_{a,b}(x') = y'$.
Es gilt:

$$\begin{aligned}(a,b) \in K(x,y) \wedge K(x',y') &\Leftrightarrow h_{a,b}(x) = y \wedge h_{a,b}(x') = y' \\ &\Leftrightarrow ax + b \equiv_p y \wedge ax' + b \equiv_p y' \\ &\Leftrightarrow \begin{pmatrix} x & 1 \\ x' & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \equiv_p \begin{pmatrix} y \\ y' \end{pmatrix}\end{aligned}$$

Damit gibt es genau dann genau eine Lösung, wenn $\begin{pmatrix} x & 1 \\ x' & 1 \end{pmatrix}$ vollen Rang hat.

Wegen $x \neq x'$ ist dies der Fall. □

Bemerkung
Im obigen Beispiel ist der Schlüssel durch $k = 3a + b$ gegeben, was die Matrix für die ersten drei Spalten erzeugt. Für die letzte Spalte verwenden wir die Funktion $h_{a,b}(x) = a$.
Wir kodieren nun die Klartexte ebenfalls durch $x = 3x_1 + x_2$ und können damit die Hashfunktion einheitlich für alle 4 Spalten mit $h_{a,b}(x_1, x_2) = ax_1 + bx_2 \mod 3$ definieren.

Satz 1.8. *Für jeden $(n,m,1)$-MAC gilt $n \leq m+1$, d.h. $l = |K| = m^2 \geq (n-1)^2$.*

Beweis:
Sei A die Authentikationsmatrix eines $(n,m,1)$-MACs.

A	x_1	$\cdots$	x_n
k_1			
$\vdots$		y	
k_l			

O.B.d.A. sei $Y = \{0, ..., m-1\}$. Wenn wir die Einträge in einer Spalten durch eine Permutation $\pi : Y \to Y$ ersetzen (jeder Eintrag y wird durch $\pi(y)$ ersetzt), so bleibt die Eigenschaft der starken Universalität erhalten. (Wir können π zu einer Permutation auf einem Spaltenpaar durch $(y, y') \mapsto (\pi(y), y')$ erweitern, womit offensichtlich die Anzahl der Vorkommen jedes Paares (y, y') nicht verändert wird.)
Daher können wir o.B.d.A. annehmen, dass A in der ersten Zeile nur den Wert 0 enthält.

$\Rightarrow$

A	x_1	$\cdots$	x_n
k_1	0	$\cdots$	0
$\vdots$		y	
k_l			

In den übrigen Zeilen können dann nur noch höchstens eine Null pro Zeile vorkommen. In jeder Spalte kommen genau m Nullen vor (für jeden möglichen Hashwertpartner in einer anderen Spalte genau eine), also kommen in der gesamten Matrix genau nm Nullen vor.
Es gilt nun:
m^2 = Anzahl der Zeilen $\geq$ Anzahl der Zeilen mit mindestens einer Null $= nm - n + 1$
$\Rightarrow m^2 \geq nm - n + 1 = n(m-1) + 1$
$\Rightarrow (m+1)(m-1) = m^2 - 1 \geq n(m-1)$
$\Rightarrow m + 1 \geq n$
$\Rightarrow m \geq n - 1$
$\Rightarrow l = m^2 \geq (n-1)^2$ □

Satz 1.9. *Sei p prim und für $x = (x_1, ..., x_d) \in \{0,1\}^d$ und $k = (k_1, ..., k_d) \in \mathbb{Z}_p^d$ sei $h_k(x) = \langle k, x \rangle_p$ (Skalarprodukt modulo p).*
Dann ist $(X, Y, K, \mathcal{H})$ mit $X = 2^d \backslash \{0^d\}$, $Y = \mathbb{Z}_p$, $K = \mathbb{Z}_p^d$ und $\mathcal{H} = \{h_k | k \in K\}$ ein $(2^d - 1, p, p^{d-2})$-MAC.

Beweis:
Seien $x = (x_1, ..., x_d) \neq (x'_1, ..., x'_d) = x'$ mit den Hashwerten y und y' gegeben. Dann gibt es (wie im vorherigen Fall für $d = 2$) genau dann genau ein Vorkommen des Paares (y, y'), wenn die Matrix $\begin{pmatrix} x_1 & \dots & x_d \\ x'_1 & \dots & x'_d \end{pmatrix}$ den Rang 2 hat, d.h. wenn x und x' linear unabhängig sind. □

Satz 1.10. *Für jeden (n, m, λ)-MAC gilt $\lambda \geq \frac{n(m-1)+1}{m^2}$, d.h. $l = |K| \geq n(m-1)+1$.*

Beweis:
Wie zuvor betrachten wir die Authentikationsmatrix A und permutieren die Einträge wieder so, dass in der ersten Zeile nur Nullen vorkommen.
Sei $\#_i$ die Anzahl der Nullen in der Zeile k_i.
Dann ist die Anzahl der Nullen in A genau $\sum_{i=1}^{l} \#_i = n\lambda m$.
$\Rightarrow \sum_{i=2}^{l} \#_i = n(\lambda m - 1)$
Sei z die Anzahl der Vorkommen von Indexpaaren (j, j') mit $A[k_i, x_j] = A[k_i, x_{j'}] = 0$ in den Zeilen $i = 2, ..., l$. (Im Fall $\lambda = 1$ war diese Anzahl immer 0 gewesen.)
Es gilt $z = \sum_{i=2}^{l} \#_i(\#_i - 1) = \sum_{i=2}^{l} \#_i^2 - \sum_{i=2}^{l} \#_i = \sum_{i=2}^{l} \#_i^2 - n(\lambda m - 1)$.
Mit der Ungleichung zwischen arithmetischem und quadratischem Mittel erhalten wir die Abschätzung
$\sum_{i=2}^{l} \#_i^2 \geq \frac{(\sum_{i=2}^{l} a_i)^2}{l-1}$
$\Rightarrow z = \sum_{i=2}^{l} \#_i^2 - n(\lambda m - 1) \geq \frac{(\sum_{i=2}^{l} a_i)^2}{l-1} - n(\lambda m - 1) = \frac{n^2(\lambda m-1)^2}{l-1} - n(\lambda m - 1)$
Andererseits gibt es für jedes Spaltenpaar genau λ Paare der Form $(0, 0)$, d.h. $\lambda - 1$ derartige Paare, welche sich nicht in der ersten Zeile befinden. Also gilt insgesamt: $z = (\lambda - 1)n(n-1)$.

$$\begin{aligned}
(\lambda-1)n(n-1) &\geq \tfrac{n^2(\lambda m-1)^2}{l-1} - n(\lambda m - 1) \\
((\lambda-1)n(n-1) + n(\lambda m - 1))(l-1) &\geq n^2(\lambda m-1)^2 \\
((\lambda-1)n(n-1) + n(\lambda m - 1))(\lambda m^2 - 1) &\geq n^2(\lambda m-1)^2 \\
(\lambda n - n - \lambda + \lambda m)(\lambda m^2 - 1) &\geq n(\lambda m - 1)^2 \\
\lambda^2 nm^2 + \lambda^2 m^3 - \lambda nm^2 - \lambda^2 m^2 - \lambda n + n - \lambda - \lambda m &\geq n\lambda^2 m^2 - 2n\lambda m + n \\
\lambda^2 m^3 - \lambda^2 m^2 &\geq \lambda nm^2 + \lambda n - \lambda + \lambda m - 2\lambda nm \\
\lambda^2(m^3 - m^2) &\geq \lambda(nm^2 + n - 1 + m - 2nm) \\
\lambda^2(m^3 - m^2) &\geq \lambda(n(m^2 - 2m + 1) + m - 1) \\
\lambda^2(m^3 - m^2) &\geq \lambda(n(m-1)^2 + (m-1)) \\
\lambda m^2(m-1) &\geq (n(m-1)+1)(m-1) \\
\lambda m^2 &\geq n(m-1)+1 \\
\lambda &\geq \tfrac{n(m-1)+1}{m^2}
\end{aligned}$$

□

Vorlesung am 13.05.2008

Definition (Informationsgehalt, Entropie)
Wir definieren den *Informationsgehalt* einer Nachricht x_i mit Wahrscheinlichkeit $P(x_i) = p_i$ für positive Wahrscheinlichkeiten p_i durch $Inf(x_i) = \log_2 \frac{1}{p_i} = -\log_2 p_i$. Wenn $p_i = 0$ gilt, so setzen wir $Inf(x_i) = 0$.
Sei X eine Zufallsvariable mit einem endlichem Wertebereich $W(X) = \{x_1, ..., x_n\}$ und Wahrscheinlichkeiten $p_i = P[X = x_i]$.
Dann ist die *Entropie* $H(X)$ definiert durch

$$H(X) = E[Inf(X)] = \sum_{i=1}^{n} p_i Inf(x_i) = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i}.$$

Die *gemeinsame Entropie* zweier Zufallsvariablen X und Y mit Verteilung $P(X = x_i, Y = y_j) = p_{i,j}$ ist $H(X, Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} p_{i,j} \log_2 p_{i,j}$. Die *bedingte Entropie* definieren wir als
$H(X|Y) = -\sum_{i=1}^{m} p(y_i) \sum_{j=1}^{n} p(x_j|y_i) \log_2 p(x_j|y_i) = -\sum_{i,j} p(x_j, y_i) \log_2 p(x_j|y_i)$.

Diese können wir als den erwarteten Informationsgehalt von X, wenn Y bereits bekannt ist, verstehen.

Lemma 1.2. (Jensensche Ungleichung) *Sei X eine Zufallsvariable mit $W(X) \subseteq \mathbb{R}^+$. Dann gilt $E[\log(X)] \leq \log E[X]$, d.h. $\sum_{i=1}^{n} p_i \log x_i \leq \log \sum_{i=1}^{n} p_i x_i$.*

Beweis:
Übung. □

Satz 1.11. *Für jeden MAC $(X, Y, K, \mathcal{H})$ gilt $\alpha \geq \frac{1}{2^{H(\mathcal{K})-H(\mathcal{K}|\mathcal{X},\mathcal{Y})}}$. Hierbei sind $\mathcal{X}$ und $\mathcal{Y}$ und $\mathcal{K}$ Zufallsvariablen, welche die Verteilung der Nachrichten, Hashwerte und Schlüssel beschreiben.*

Beweis:
Wir zeigen $\log \alpha \geq H(\mathcal{K}|\mathcal{X},\mathcal{Y}) - H(\mathcal{K})$.
Zur Erinnerung: Es gilt $\alpha = \max_{(x,y)\in X\times Y} p(x \mapsto y)$ und $p(x \mapsto y) = \sum_{k\in K, h_k(x)=y} p(k) = p(y|x)$.
$\Rightarrow \alpha \geq \sum_{x,y} p(x,y) \cdot p(x \mapsto y) = E(\alpha(\mathcal{X},\mathcal{Y}))$
$\Rightarrow \log \alpha \geq \log E(\alpha(\mathcal{X},\mathcal{Y})) \geq E(\log(\alpha(\mathcal{X},\mathcal{Y}))) = \sum_{x,y} p(x,y) \log \alpha(\mathcal{X},\mathcal{Y})$
$\Rightarrow \log \alpha \geq \sum_{x,y} p(x,y) \log \alpha(\mathcal{X},\mathcal{Y}) = -\sum_{x,y} p(x)p(y|x) \log \alpha(\mathcal{X},\mathcal{Y}) = -H(\mathcal{Y}|\mathcal{X})$
Nun muss noch gezeigt werden, dass $-H(\mathcal{Y}|\mathcal{X}) \geq H(\mathcal{K}|\mathcal{X},\mathcal{Y}) - H(\mathcal{K})$ gilt. Wir beweisen sogar, dass an dieser Stelle immer die Gleichheit gilt. Hierfür gehen wir von der gemeinsamen Entropie aus. Diese lässt sich auf verschiedenen Wegen entwickeln:
$H(\mathcal{X},\mathcal{K}) + H(\mathcal{Y}|\mathcal{X},\mathcal{K}) = H(\mathcal{X},\mathcal{Y},\mathcal{Z}) = H(\mathcal{X}) + H(\mathcal{Y}|X) + H(\mathcal{K}|\mathcal{X},\mathcal{Y})$
$\Rightarrow H(\mathcal{X}) + H(\mathcal{K}) + 0 = H(\mathcal{X}) + H(\mathcal{Y}|X) + H(\mathcal{K}|\mathcal{X},\mathcal{Y})$
$\Rightarrow H(\mathcal{Y}|\mathcal{X}) = H(\mathcal{K}) - H(\mathcal{K}|\mathcal{X},\mathcal{Y})$ □

Vorlesung am 15.05.2008

Definition (CBC-MAC)
Wie auch bei einem Verschlüsselungsverfahren im CBC-Modus können wir einen *CBC-MAC* definieren, indem wir für eine Folge von Eingabeblöcken $x_1, ..., x_n$ und einen Initialisierungsvektor $IV = y_0$ jeweils $y_j = E_k(y_{j-1} \oplus x_j)$ für $j = 1, ..., n$ bestimmen und $y = y_n$ als Hashwert verwenden, d.h. es gilt $h_k(x) = y = y_n$ für $x = x_1...x_n$. Der Einfachheit halber können wir $y_0 = IV = 0^l$ verwenden, wobei $l = |x_j|$ die Blocklänge ist. Dabei ist zu beachten, dass alle Eingabetexte x die gleiche Blockanzahl haben müssen.
(Für unterschiedlich lange Texte gibt es den folgenden leichten Angriff:
$x_1 \mapsto y_1$ und $x_1x_2 \mapsto y_2$ werden bei frei wählbarem Text abgefragt.
Dann erhält man mit $(IV \oplus y_1 \oplus x_2, y_2)$ ein zulässiges Klartext-Hashwert-Paar.
Bei adaptiv wählbarem Text kann x_2 sogar so gewählt werden, dass $x = IV \oplus y_1 \oplus x_2$ ein zuvor frei wählbarer Text ist.)

CBC-MAC-Geburtstagsangriff
Ziel: Generiere ein Paar (x', y') mit $h_k(x') = y'$, wobei k der dem Gegner unbekannte Schlüssel ist. Hierbei darf der Gegner Hashwerte für von ihm gewählte Texte $x^j \neq x', j = 1, ..., q$ erfragen. Sei $q \approx 2^{\frac{l}{2}} = \sqrt{2^l}$ (d.h. der Geburtstagsangriff ist mit einer Wahrscheinlichkeit von etwa $\frac{1}{2}$ erfolgreich).
Wir wählen Texte der Form
$x^1 = x_1^1 x_2^1 x_3 x_4 x_5 ... x_n$
$x^2 = x_1^2 x_2^2 x_3 x_4 x_5 ... x_n$
$x^3 = x_1^3 x_2^3 x_3 x_4 x_5 ... x_n$
$\vdots$
$x^q = x_1^q x_2^q x_3 x_4 x_5 ... x_n$
mit $x^j \mapsto y^j$, wo die Einträge der ersten Spalte (d.h. die x_1^i) paarweise verschieden und die der zweiten Spalte (d.h. x_2^i) zufällig gewählt werden. Wenn wir eine Kollision, d.h. $i \neq j$ mit $y^i = y^j$ haben, konstruieren wir daraus das gesucht Paar (x', y').

Von der gefundenen Kollision ausgehend erhalten wir wegen der Bijektivität von E_k und der Gleichheit der Klartextblöcke 3 bis n, dass $y_1^i \oplus x_2^i = y_1^j \oplus x_2^j$ gilt. Wegen $x_1^i \neq x_1^j$ gilt $y_1^i \neq y_2^j$.
Jetzt fragen wir noch einmal den Hashwert y^{q+1} von $x^{q+1} = x_1^i(x_2^i \oplus z)x_3x_4...x_n$ mit $z \neq 0$ ab. Das gesuchte Paar ist $(x^{q+1} = x_1^i(x_2^i \oplus z)x_3x_4...x_n, y^{q+1})$.
(Wegen $y_1^i \oplus x_2^i = y_1^j \oplus x_2^j$ gilt $z \oplus y_1^i \oplus x_2^i = z \oplus y_1^j \oplus x_2^j$, woraus die Gleichheit der Hashwerte y' und y^{q+1} folgt.)

Vorlesung am 20.05.2008

Definition ((ε, q)-Fälscher)
Seien $0 \leq \varepsilon \leq 1$ und $q \in \mathbb{N}$. Ein (ε, q)-*Fälscher* für eine Hashfamilie $\mathcal{H} = (X, Y, K, H)$ ist ein probabilistischer Algorithmus A, der bis zu q Fragen $x_1, ..., x_q \in X$ stellt und aus den Antworten $y_i = h_k(x_i), i = 1, ..., q$ mit Wahrscheinlichkeit $\geq \varepsilon$ (bei zufällig gewähltem Schlüssel k) ein Paar $(x', y') \in X \times Y$ mit $x' \notin \{x_1, ..., x_q\}$ und $h_k(x') = y'$ ausgibt.
Wir unterscheiden zwischen *adaptiven* (d.h. der Text x_i darf von $h_k(x_1), ..., h_k(x_{i-1})$ abhängen) und *nicht-adaptiven* Fragen.
Zudem unterscheiden wir zwischen *selektiven* (d.h. der Gegner kann den Hashwert für einen Text seiner Wahl generieren) und *existenzielle* Fälschungen.

Beispiel
Der betrachtete Geburtstags-Angriff auf einen CBC-MAC beschreibt einen existenziellen $(\frac{1}{2}, q+1)$-Fälscher mit adaptiven Fragen und einen $(\frac{1}{2}, 2q)$-Fälscher mit nicht adaptiven Fragen. Dabei ist $q \approx 1,17\sqrt{2^l}$.

Definition ((ε, q)-Kollisionsangreifer)
Ein (ε, q)-*Kollisionsangreifer* für eine Hashfamilie $\mathcal{H} = (X, Y, K, H)$ ist ein probabilistischer Algorithmus A, der q Fragen $x_1, ..., x_q \in X$ stellt und aus den Antworten $h_k(x_1), ..., h_k(x_q)$ mit Wahrscheinlichkeit $\geq \varepsilon$ (bei zufällig gewähltem und dem Gegner nicht bekanntem Schlüssel k) ein Kollisionspaar (x, x') mit $h_k(x) = h_k(x')$ und $x \neq x'$ ausgibt. x und x' dürfen dabei auch unter x_1 bis x_q vorkommen.

Bemerkung
Da der Gegner den Schlüssel k nicht kennt, ist ein Kollisionsangriff gegen eine Hashfamilie $\mathcal{H}$ schwieriger zu realisieren, als gegen eine schlüssellose Hash-Funktion.

Definition (Nested-MAC, Komposition von Hashfamilien)
Seien $\mathcal{H}_1 = (X, Y, K_1, F)$ und $\mathcal{H}_2 = (Y, Z, K_2, G)$ zwei Hashfamilien.
Dann ist $\mathcal{H} = \mathcal{H}_1 \circ \mathcal{H}_2 = (X, Z, K, H)$ die *Komposition* von $\mathcal{H}_1$ und $\mathcal{H}_2$, wobei $K = K_1 \times K_2$ und $H = \{f_{k_1} \circ g_{k_2} | (k_1, k_2) \in K\}$ ist.

Beispiel
Waählt man für $\mathcal{H}_2$ eine (im optimalen Fall stark universale) Hashfamilie und für $\mathcal{H}_1$ eine schlüssellose Hashfunktion (etwa SHA-1) so erhält man einen so genannten HMAC (*Hash-MAC*). Der HMAC-SHA-1 benutzt beispielsweise die 512Bit-Konstanten $ipad = \underbrace{36...36}_{64-\text{mal}}$ und $opad = \underbrace{5C...5C}_{64-\text{mal}}$
und wird wie folgt berechnet:
HMAC-SHA-$1_k(x)$ =SHA-1$((k \oplus opad)$SHA-$1((k \oplus ipad)x)$

Satz 1.12. *Sei $\mathcal{H} = (X, Z, K, H) = \mathcal{H}_1 \circ \mathcal{H}_2$. Falls für $\mathcal{H}_1$ kein $(\varepsilon_1, q+1)$-Kollisionsangreifer und für $\mathcal{H}_2$ kein (ε_2, q)-Fälscher existieren, dann gilt für jeden (ε, q)-Fälscher für $\mathcal{H}$ $\varepsilon < \varepsilon_1 + \varepsilon_2$.*

Beweis:
Sei A ein (ε, q)-Fälscher für $\mathcal{H}$. Zu zeigen ist $\varepsilon < \varepsilon_1 + \varepsilon_2$.
Wir wählen zufällig einen Schlüssel $k = (k_1, k_2) \in K$ und simulieren A. Bei dieser Simulation stellt A bis zu q Fragen $x_1, ..., x_q \in X$, die wir mit $z_i = g_{k_2}(f_{k_1}(x_i)), i = 1, ..., q$ beantworten, und gibt am Ende mit Wahrscheinlichkeit $\geq \varepsilon$ ein Paar (x_{q+1}, z) mit $x_{q+1} \notin \{x_1, ..., x_q\}$ und $g_{k_2}(f_{k_1}(x_{q+1})) = z$ aus.

1. Fall: Es gibt $1 \leq i < j \leq q+1$ mit $f_{k_1}(x_i) = f_{k_1}(x_j)$.

   Da für $\mathcal{H}_1$ kein (ε_1, q)-Kollisionsangriff existiert, kann dieser Fall höchstens mit Wahrscheinlichkeit $< \varepsilon_1$ eintreten.

2. Fall: Es gilt $|\{f_{k_1}(x_1), ..., f_{k_1}(x_{q+1})\}| = q+1$ und $g_{k_2}(f_{k_1}(x_{q+1})) = z$.

   Weil Fall 1 mit Wahrscheinlichkeit $< \varepsilon_1$ eintritt, muss dieser Fall mit Wahrscheinlichkeit $> \varepsilon - \varepsilon_1$ eintreten. In diesem Fall muss A mit einer größeren Wahrscheinlichkeit als $\varepsilon - \varepsilon_1$ ein gültiges Paar (x_{q+1}, z) mit $g_{k_2}(f_{k_1}(x_{q+1})) = z$ ausgeben.

   Analog zur Analyse des ersten Falls existiert auch für $\mathcal{H}_2$ kein (ε_2, q)-Fälscher, d.h. es muss $\varepsilon - \varepsilon_1 < \varepsilon_2$ sein, d.h. es gilt $\varepsilon < \varepsilon_1 + \varepsilon_2$. □

Bemerkung:
A kann zu einem Angreifer auf $\mathcal{H}_1$ bzw. $\mathcal{H}_2$ umgewandelt werden, indem der fehlende Schlüsselteil (k_2 bzw. k_1) zufällig gewählt wird und die Fragen $x_1, ..., x_q$ entsprechend der Fragen von A gewählt werden.

1.5 Digitale Signaturverfahren

Handschriftliche Signaturen

1. Die durch die Unterschrift gekennzeichnete Person hat überprüfbar die Unterschrift geleistet.
2. Das signierte Dokument kann nachträglich nicht mehr unbemerkt verändert werden.
3. Die Unterschrift kann nicht von einem Dokument auf ein anderes übertragen werden.

Eine direkte Übertragung dieser Eigenschaften in die digitale Welt ist nicht möglich.
Lösung: Die digitale Signatur wird nicht physikalisch, sondern logisch (d.h. inhaltlich) an das digitale Dokument gebunden.

Definition
Ein digitales Signaturverfahren besteht aus folgenden Komponenten:

1. endliche Menge X von Dokumenten
2. endliche Menge Y von Unterschriften
3. endliche Menge K von Schlüsseln
4. Menge $S \subseteq K \times K$ von Schlüsselpaaren $(\hat{k}, k)$
5. Signatur / Signieralgorithmus $sig : K \times X \to Y$
6. Verifikationsalgorithmus $ver : K \times X \times Y \to 2 = \{0, 1\}$

   mit $ver(k, x, y) = 1 \Leftrightarrow y = sig(\hat{k}, x), (\hat{k}, k) \in S$

Vorlesung am 22.05.2008

Mögliche Angriffe gegen Signaturverfahren

1. *Angriff bei bekanntem Verifikationsschlüssel k*

   Der Gegner versucht, ohne Kenntnis des Signierschlüssels $\hat{k}$, ein Paar (x, y) mit $ver(k, x, y) = 1$ zu erzeugen.

2. *Angriff bei bekannter Signatur*

   Der Gegner kennt neben k noch eine Reihe von mittels $\hat{k}$ signierten Dokumenten (x_1, y_1), ..., (x_n, y_n) und versucht daraus ein Paar (x, y) mit $ver(k, x, y) = 1$ und $x \notin \{x_1, ..., x_n\}$ zu erzeugen.

3. *Angriff bei frei wählbaren Dokumenten*

   Der Gegner kann sich für eine gewisse Zeit für Dokumente seiner Wahl die zugehörigen Signaturen beschaffen, d.h. er kann die $x_1, ..., x_n$ frei wählen.

4. *Angriff bei adaptiv wählbaren Dokumenten*

   Der Gegner kann die Wahl des Textes x_i von den Signaturen $y_1, ..., y_{i-1}$ abhängig machen.

Erfolgskriterien für das Fälschen von Signaturen

1. *Uneingeschränktes Fälschungsvermögen*

   Der Gegner hat einen Weg gefunden, ohne Kenntnis von $\hat{k}$ die Funktion $x \mapsto sig(\hat{k}, x)$ effizient zu berechnen.

2. *Selektives Fälschungsvermögen*

   Der Gegner kann für Dokumente seiner Wahl die zugehörige Signatur bestimmen. (Evtl. mit Hilfe des legalen Unterzeichners.)

3. *Existentielles* bzw. *nicht selektives Fälschungsvermögen*

   Der Gegner kann nur für einige bestimmte Dokumente die zugehörige Signatur bestimmen.

Das RSA-Kryptosystem
Sei $n = pq$ mit großen $p, q \in \mathbb{P}$, d.h. n kann nicht effizient faktorisiert werden.
Der private Schlüssel sind ein Exponent $d \in \mathbb{Z}_n^*$ und n. Der öffentliche Schlüssel besteht aus $e \in \mathbb{Z}_n^*$ mit $ed \equiv_{\varphi(n)} 1$ und n. Die Verschlüsselung erfolgt durch durch $E(e, n, x) = x^e \mod n$ und die Entschlüsselung durch $D(d, n, y) = y^d \mod n$.

Satz 1.13. $ed \equiv_{\varphi(n)} 1 \Rightarrow \forall x \in \mathbb{Z}_n : x^{ed} \equiv_n x$

Beweis:
Siehe "Kryptologie I" oder "Linare Algebra I" oder sonst irgendwas in der Art. □

1.6 Das RSA-Signaturverfahren

Definition (RSA-Signaturverfahren)
Wir wählen zwei große Primzahlen $p, q \in \mathbb{P}$ und bestimmen $n = pq$. Der Signierschlüssel ist (d, n) mit $d \in \mathbb{Z}_n^*$ und der Verifikationsschlüssel (e, n) mit $de \equiv_{\varphi(n)} 1$. Die Signierfunktion ist durch $sig(d, n, x) = x^d \mod n$ und die Verifikation durch $ver(e, n, x, y) = \begin{cases} 1 & \text{, wenn } y^e \equiv_n x \\ 0 & \text{sonst} \end{cases}$ gegeben. Die Korrektheit folgt direkt aus der Korrektheit des RSA-Kryptosystems.

Existentielle Fälschung bei bekanntem Verifikationsschlüssel
Die Dokumente $x = 0$ und $x = 1$ werden immer durch 0 bzw. 1 signiert, also werden diese von vornherein ausgeschlossen.
Gesucht sind Paare (x, y) mit $ver(e, n, x, y) = 1$.
Einfach zu realisieren: Wir wählen eine beliebige Signatur y und berechnen dazu das Dokument $x = y^e \mod n$.
Mögliche Gegenmaßnahmen:

1. Anstelle des Dokuements x wird nur der Hashwert $h(x)$ signiert.
2. x wird mit genügend Redundanz ausgestattet (z.B. $x = x'x'$).

Signierung des Hashwertes von x statt x selbst

1. Die Hashfunktion h sollte die Einweg-Eigenschaft besitzen, da der Gegner sonst für den Hashwert $y' = y^e \mod n$ ein Dokument x mit $h(x) = y'$ finden kann.
2. Falls bereits signierte Dokumente (x, y) bekannt sind, so darf der Gegner kein weiteres Urbild x' von $h(x)$ finden, d.h. h muss zumindest schwach kollisionsresistent sein.

Angriffsmöglichkeiten
Der Gegner kann zunächst 2 Dokumente x und x' verfassen, von denen x harmlos ist und vom legalen Unterzeichner ohne weiteres signiert wird, und x' das Dokument ist, für welches der Gegner eine Signatur erzeugen möchte.
Der Gegner kann nun die Texte x und x' leicht abwandeln und Texte $x_1, ..., x_n$ sowie $x'_1, ..., x'_m$ erzeugen, unter denen er ein Kollisionspaar (x_i, x'_j) auswählt, x_i signieren lässt und x'_j mit der Signatur von x_i versieht.
h sollte also sogar (stark) kollisionsresistent sein.

Existentielle Fälschung bei bekannten Dokumenten
Falls der Gegner die Signaturen y_1 und y_2 von x_1 und x_2 kennt, so hat $x_1 \cdot x_2$ die Signatur $y_1 \cdot y_2$, da
$(y_1 \cdot y_2)^e = y_1^e \cdot y_2^e = x_1 \cdot x_2$.

Selektive Fälschung bei frei wählbaren Dokumenten
Wir betrachten wieder das RSA-Signaturverfahren ohne Hashfunktion. Um sich die Signatur für ein Dokument x zu beschaffen, kann der Gegner wie folgt vorgehen:
Er wählt ein y_2 beliebig und bestimmt $x_2 = y_2^e$. Dann lässt er $x_1 = x \cdot (x_2)^{-1}$ signieren und erhält die Signatur $y_1 = x_1^d$. Damit erhält er für x die Signatur $y_1 \cdot y_2$.

Kapitel 2

Der diskrete Logarithmus

2.1 Allgemeines

Definition (Ordnung eines Gruppenelements)
Sei $(G, \cdot)$ eine endliche Gruppe mit neutralem Element 1 und $\alpha \in G$ mit $ord_G(\alpha) = |\langle a \rangle| = n$. Dabei ist $\langle a \rangle = \{a^n | n \in \mathbb{N}\}$ die von α in G erzeugte Untergruppe.
Es gilt $ord(\alpha) = \min\{i \in \mathbb{N} | i > 0, \alpha^i = 1\}$.

Beispiel

1. Seien $G = \mathbb{Z}_p^*, p \in \mathbb{P}$ und $\alpha \in \mathbb{Z}_p^* \setminus \{1\}$. Dann gilt $ord(\alpha) = p - 1$.
2. Seien $G = \mathbb{Z}_p^*, p \in \mathbb{P}$, q ein (Prim-)Teiler von $p - 1$ und α ein Erzeuger von $\mathbb{Z}_p^*$.

   Dann ist $\beta = \alpha^{\frac{p-1}{q}}$ ein Element der Ordnung $ord(\beta) = q$.

Naive Algorithmen zur Berechnung des diskreten Algorithmus

1. Eingabe: $G, n = ord(\alpha), \alpha, \beta$, Gesucht: $\log_\alpha \beta = \min\{i \in \mathbb{N} | \beta = \alpha^i\}$.

   Potenziere α so lange, bis $\alpha^i = \beta$ gilt und gib i aus.

   Dieses Verfahren hat einen Zeitaufwand von $O(n)$ und einen Speicherplatzaufwand von $O(1)$.
2. **Precomputation:** Sortiere die Paare $(i, \alpha^i), i = 0, ..., n - 1$ nach der zweiten Komponente und schreibe diese Paare in dieser Reihenfolge in eine Tabelle.

   Zeitaufwand: $O(n \log n)$

   Speicheraufwand: $O(n)$

   Computation: Ermittle mittels Binärsuche den Eintrag (i, α^i) mit $\beta = \alpha^i$.

   Zeit: $O(\log n)$

   Platz: $O(n)$

Vorlesung am 27.05.2008

SHANKS
Wir werden versuchen, die Vorteile beider Algorithmen zu verbinden, indem wir für $d = \lceil \sqrt{n} \rceil$ nur die Potenzen α^{dj} mit $j = 0, ..., d-1$ vorausberechnen und speichern. Anschließend suchen wir von β ausgehend nach dem ersten dieser vorausberechneten Werte, d.h. wir berechnen $\beta, \beta\alpha^{-1}, \beta\alpha^{-2}, ...$ Wenn wir $\beta\alpha^{-k} = \alpha^{dj}$ erhalten, wissen wir somit, dass $\beta = \alpha^{dj+k}$ gilt.

Beispiel
Seien $G = \mathbb{Z}^*_{809}, \alpha = 3, ord(\alpha) = 808$ und $\beta = 525$.
Es gilt $d = \lceil\sqrt{808}\rceil = 29$.
Wir berechnen die Tabelle der Paare (i, α^{di}) mit $(0,1), (23,15), (12,26), ..., (10,644), ..., (14,800)$ (sortiert nach dem zweiten Koeffizienten). Dann bestimmen wir $525, 525 \cdot 3^{-1}, ..., 525 \cdot 3^{-19} = 644$.
$\Rightarrow \log_3 525 = 10 \cdot 29 + 19 = 309$

ϱ-Faktorisierungsalgorithmus von Pollard
Gegeben sei n. O.B.d.A. sei n ungerade und enthalte mindestens zwei verschiedene Primfaktoren $p \neq q$. (Wenn n eine Primzahl ist, so lässt sich dies mit einem Primzahltest effizient feststellen. Wenn n eine Primzahlpotenz ist, so lässt sich durch Bestimmung der Wurzeln $\sqrt{n}, \sqrt[3]{n}, ..., \sqrt[\lceil\log_2 n\rceil]{n}$ die Basis der Potenz ebenfalls effizient berechnen.)
Wählen wir nun eine zufällige Teilmenge $X \subseteq \mathbb{Z}_n$ der Größe $|X| \sim \sqrt{p}$, so enthält X mit Wahrscheinlichkeit $\frac{1}{2}$ zwei Elemente $x \neq x'$ mit $x \equiv_p x'$. In diesem Fall können wir den nichttrivialen Teiler $d = \text{ggT}(x - x', n)$ von n bestimmen. Da wir jedoch alle Paare (x, x') betrachten müssen (wir kennen ja p zunächst noch nicht), ist die Komplexität noch $O(p)$.
Wir wählen also anstelle von X eine pseudozufällige Menge $X_j = \{x_0, ..., x_{j-1}\}$, wobei x_0 zufällig und $x_{i+1} = f(x_i)$ durch eine Funktion $f : \mathbb{Z}_n \to \mathbb{Z}_n$ berechnet wird. Bei geeigneter Wahl von f (z.B. $f(x) = x^2 + 1 \mod n$) tritt innerhalb von X_j für $j \approx \sqrt{p}$ eine Kollision $x_i \equiv_p x_j$ für ein $i < j$ mit $x_i \neq x_j$ auf.
Wie lässt sich nun effizient ein Kollisionspaar (x_i, x_j) in X_j bestimmen, ohne p zu kennen?
Werden zur Berechnung von f nur $\cdot$ (Multiplikation) und $+$ (Addition) sowie ganzzahlige Konstanten aus $\mathbb{Z}_n$ verwendet, dann gilt $x_i \equiv_p x_j \Rightarrow x_{i+1} \equiv_p x_{j+1}$. Hieraus folgt dann $x_k \equiv_p x_{k+d(j-i)}$ für alle $k \geq i, d \geq 1$.
Es gilt also $x_k \equiv_p x_{2k}$, falls $k \geq i$ und $k \equiv_{j-i} 0$.
Ein solches k lässt sich im Intervall $i, i+1, ..., j = i + (j - i)$ finden. Damit reicht es dann auch aus, nur Paare der Form (x_i, x_{2i}) auf Kollisionen zu testen.

Bemerkung
Der Name des Algorithmus stammt von der graphischen Darstellung der Folge $x_0, x_1, ...$, weil das Anfangsstück wir eine Linie aussieht, die dann auf einen Kreis führt (ähnlich zum Buchstaben ϱ). Der Kreis ist die Veranschaulichung der Periodiziät der Folge ab einem bestimmten i.

Algorithmus: ϱ-Faktorisierung von Pollard

```
integer Pollard(n, zusammengesetzt, ungerade, keine Primzahlpotenz){
    x =select({0, ..., n − 1}, random);
    y = x;
    while(true) {
        x = f(x) mod n;
        y = f(y) mod n;
        y = f(y) mod n;
        if (x ≠ y) {
            g = ggT(x − y, n);
            if (g > 1)
                return g;
        }
    }
}
```

Vorlesung am 29.05.2008

ϱ-DLP-Algorithmus von Pollard
Gegeben ist eine Gruppe $(G, \cdot, 1)$, $\alpha \in G$ mit $ord(\alpha) = n$ und $\beta \in \langle\alpha\rangle$.
Gesucht ist $\log_\alpha \beta \in \mathbb{Z}_n$.

Sei (S_1, S_2, S_3) eine Partition von G mit $|S_1| \approx |S_2| \approx |S_3|$ gegeben.
Wir betrachten die Funktion $f : \langle\alpha\rangle \times \mathbb{Z}_n \times \mathbb{Z}_n \to \langle\alpha\rangle \times \mathbb{Z}_n \times \mathbb{Z}_n$ mit

$$f(x,a,b) = \begin{cases} (\beta x, a, b+1), \text{ wenn } x \in S_1 \\ (x^2, 2a, 2b) \quad , \text{ wenn } x \in S_2 \\ (\alpha x, a+1, b), \text{ wenn } x \in S_3 \end{cases}.$$

Dann gilt $x = \alpha^a\beta^b \Rightarrow f(x,a,b) = (x', a', b')$ mit $x' = \alpha^{a'}\beta^{b'}$.
Angenommen, wir finden in der Folge $z_1, z_2, ...$ mit $z_1 = (1,0,0)$ und $z_{i+1} = f(z_i)$ ein Tripel $z_i = (x_i, a_i, b_i)$ sodass $x_i = x_{2i}$. Dann gilt $x_i = \alpha^{a_i}\beta^{b_i} = x_{2_i} = \alpha^{a_{2i}}\beta^{b_{2i}}$. Dies ist äquivalent zu $a_i + cb_i \equiv_n a_{2i} + cb_{2i} \Leftrightarrow c(b_i - b_{2i}) \equiv_n a_{2i} - a_i$ (mit $c = \log\alpha\beta$). D.h. im Fall $\text{ggT}(b_i - b_{2i}, n) = 1$ lässt sich c eindeutig bestimmen: $c = (a_{2i} - a_i)(b_i - b_{2i})^{-1} \mod n$.

Beispiel
Seien $G = \mathbb{Z}_p^*, p = 809, \alpha = 89, ord(\alpha) = n = 101$ und $\beta = 618$. Mit $S_i = \{x \in \mathbb{Z}_{809}^* | x \equiv_3 a_i\}$ und $z_1 = (1,0,0)$ erhalten wir die Doppel-Tripel-Folge

i	z_i	z_{2i}
1	$(618,0,1)$	$(76,0,2)$
2	$(76,0,2)$	$(46,0,3)$
3	$(46,0,3)$	$(113,0,4)$
4	$(113,0,4)$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
10	$(422,5,11)$	$(422,11,15)$

$\text{ggT}(15-11, 101) = 1 \Rightarrow \log_{89,\mathbb{Z}_{809}^*} 618 = (11-5)(11-15)^{-1} \mod 101 = 49$

Bemerkung
Die Zeitkomplexität liegt ungefähr bei $O(\sqrt{n})$ und der Speicheraufwand ist $O(1)$.

DLP-Algorithmus von Pohlig und Hellman (PH-DLP)
Sei $n = \prod_{i=1}^k p_i^{c_i}$ die (bekannte) Primfaktorisierung von n, wobei $n = ord(\alpha)$ ist.
Der PH-Algorithmus berechnet für $i = 1, ..., k$ die Zahlen $a_i = \log_\alpha \beta \mod p_i^{c_i}$. Durch den chinesischen Restsatz können wir dann den diskreten Logarithmus $\log_\alpha \beta$ bestimmen.

Hilfsproblem
Gegeben sind n, ein Primfaktor p von n und $c \in \mathbb{N}$ mit $p^c | n$ aber $p^{c+1} \nmid n$. Weiterhin sei $\sum_{i=0}^{c-1} a_i p^i$ die p-adische Darstellung von $\log_\alpha \beta \mod p^c$ in der Basis p.
Gesucht ist $(a_0, ..., a_{c-1})$.
Berechnung von a_0:
Es gilt $\alpha^l = \beta$ für $l = \log_\alpha \beta \mod p^c = \sum_{i=0}^{c-1} a_i p^i$.
$\Rightarrow \alpha^{\frac{a_0 n}{p}} = \alpha^{\frac{a_0 n}{p}} \cdot \alpha^{n\sum_{i=1}^{c-1} a_i p^i} = \alpha^{\frac{n}{p}\sum_{i=0}^{c-1} a_i p^i} = (\alpha^l)^{\frac{n}{p}} = \beta^{\frac{n}{p}}$
Somit können wir für $j = 0, ..., p-1$ die Werte $\alpha^{\frac{jn}{p}}$ berechnen und $\alpha_0 = j$ für das j mit $\alpha^{\frac{jn}{p}} = \beta^{\frac{n}{p}}$ wählen.
Sei nun $\beta_j = \beta_{j-1}\alpha^{-a_{j-1}p^{j-1}} = \alpha^{p^j(\sum_{i=j}^{c-1} a_i p^{i-j})}, j \geq 1, \beta_0 = \beta$.
Dann gilt $\beta_j^{\frac{n}{p^{j+1}}} = \alpha^{\frac{n}{p}\sum_{i=j}^{c-1} a_i p^{i-j}} = \alpha^{a_j \frac{n}{p}}$, womit wir dann iterativ a_j bestimmen können.

Komplexität
Wir müssen für jeden Primfaktor p die c Koeffizienten a_i bestimmen. Mit Hilfe des Algorithmus von Shanks können wir diese in $O(\sqrt{p})$ bestimmen. Für jede Primzahlpotenz haben wir also einen Aufwand von $O(c\sqrt{p})$.
Als Gesamtaufwand ergibt sich $O(\sum_{i=1}^{k} c_k\sqrt{p_k})$ mit der Primfaktorzerlegung $n = \prod_{i=1}^{k} p_k^{c_k}$. Der Aufwand ist also $O(\sqrt{p}\log n)$, wobei p der größte Primfaktor ist.

Vorlesung am 03.06.2008

Untere Schranke für das DLP
Das DLP für Gruppen $G = (\mathbb{Z}_m, +, 0)$ ist sehr einfach zu lösen:
Eingabe: $\alpha, \beta \in \mathbb{Z}_m, n = ord_{\mathbb{Z}_m}(\alpha) = \frac{m}{\text{ggT}(\alpha,m)}, \beta \in \langle\alpha\rangle$
Gesucht: $\log_{\alpha,\mathbb{Z}_m} \beta = \min\{e \geq 0 | e \cdot \alpha \equiv_m \beta\}$
Sei $\alpha' = \frac{\alpha}{\text{ggT}(\alpha,m)}$, $\beta' = \frac{\beta}{\text{ggT}(\alpha,m)}$ und $m' = \frac{m}{\text{ggT}(\alpha,m)}$. e ist dann eine Lösung von $x\alpha \equiv_m \beta$. Dies ist genau dann der Fall, wenn e eine Lösung von $x\alpha' \equiv_{m'} \beta'$ ist.
Wegen $\text{ggT}(\alpha', m') = 1$ lässt sich α' in $\mathbb{Z}_{m'}$ invertieren, d.h. wir erhalten $e = \beta'(\alpha')^{-1}$. Dieses e ist dann modulo m auch die kleinste Lösung.
$\Rightarrow e = \log_{\alpha,\mathbb{Z}_m} \beta$

2.2 Generisches DLP

Idee
Das Problem sollte unabhängig von der Repräsentation der Gruppe G sein.
Wir kodieren daher das DLP wie folgt: Sei $\alpha \in G$ die Basis, 1 das Einselement und $n = ord_G\alpha$.
Sei $\delta : \mathbb{Z}_n \to \{0,1\}^*$ mit $|\delta(i)| = O(\log n)$ für $0, ..., n-1$ eine zufällig gewählte Injektion. Dann können wir auf dem Wertebereich $\delta(\mathbb{Z})$ ein isomorphes Bild $\hat{G}$ von $\mathbb{Z}_n$ erzeugen:
$\hat{1} = \delta(0), \delta(i) \circ \delta(j) = \delta(i + j \mod n)$

Das generische DLP lässt sich nun wie folgt formulieren:
Gegeben seien $\delta_1 = \delta(1) = \alpha$ und $\delta_2 = \delta(a) = \beta$ mit $a \in \mathbb{Z}_n$.
Gesucht ist $a = \log_\alpha \beta$.

Definition
Ein DLP-Algorithmus heißt *generisch*, falls er das DLP für beliebige Kodierungen δ (wie oben definiert) löst, indem er auf $\hat{G}$ Gruppenoperationen ausführt.

Beispiel
Alle bisher betrachteten DLP-Algorithmen (außer für die Gruppe $G = (\mathbb{Z}_m, +, 0)$) sind generisch.

Bemerkung
Da ein generischer DLP-Algorithmus alle berechneten Gruppenelemente aus den Eingabe-Elementen α, β generieren muss, können wir die Ausführung von Gruppenoperationen auch durch ein Orakel modellieren, das auf die Frage (c, d) die Antwort $\alpha^c\beta^d$ gibt. Ein generischer DLP-Algorithmus A berechnet also bei Eingabe $\delta_1 = \alpha, \delta_2 = \beta$ eine Folge $\delta_1, \delta_2, \delta_3, ...$ von Elementen in $\hat{G}$ mit $\delta_i = \delta(c_i + ad_i \mod n) = \alpha^{c_i}\beta^{d_i}$ und gibt am Ende a oder ? aus.

Frage
In welchen Fällen gibt A die Ausgabe a aus?
Antwort: In der Folge $\mathcal{C} = (c_1, d_1), (c_2, d_2), ...$ der von A gestellten Fragen gibt es zwei Fragen (c_i, d_i) und (c_j, d_j) auf die das Orakel die gleiche Antwort gibt τ gibt.
In diesem Fall gilt $\alpha^{c_i}\beta^{d_i} = \tau = \alpha^{c_i}\beta^{d_j}$.
$\Leftrightarrow c_i + ad_i \equiv_n c_j + ad_j$
$\Leftrightarrow a(d_i - d_j) \equiv_n c_j - c_i$
D.h. im Fall, dass n prim ist, kann A genau dann den Wert von a mit Sicherheit herausfinden, wenn A auf zwei unterschiedliche Fragen dieselbe Antwort erhält.
Falls alle Orakelantworten paarweise verschieden sind, so kann daraus der Wert a nur bestimmen, wenn es genau ein Element gibt, welches nicht in der Menge $Good(\mathcal{C}) = \{(c_j - c_i)(d_i - d_j)^{-1} \mod n | 1 \leq i < j \leq m\}$ liegt.
Wir wählen eine zufällige Kodierung $\delta : \mathbb{Z}_n \to \{0,1\}^{O(\log n)}$ und $a \in \mathbb{Z}_n$. Dann starten wir A mit der Eingabe $n, \alpha = \delta(1), \beta = \delta(a)$.
Falls A ein generischer Las-Vegas-Algorithmus für das DLP ist, gilt im Fall $\binom{m}{2} < n - 1$:
$\frac{1}{2} \leq P[A(n, \alpha, \beta) \text{ ist korrekt}] = \frac{|Good(\mathcal{C})|}{n} \leq \frac{\binom{m}{2}}{n}$.
$\Rightarrow \binom{m}{2} \geq \frac{n}{2} \Leftrightarrow m(m-1) \geq n$
$\Rightarrow m = \Omega(\sqrt{n})$

Vorlesung am 05.06.2008

2.3 Die Index-Calculus-Methode

Definition
Gegeben sind $G = \mathbb{Z}_p^*$ und ein Erzeuger α von $\mathbb{Z}_p^*$.
Der Algorithmus besteht aus 2 Phasen:

1. Berechne die Werte $\log_\alpha p_i$ für $i = 1, ..., k$, wobei $B = \{p_1, ..., p_k\}$ die sogenannte Faktorbasis ist.

2. Berechne $\log_\alpha \beta$ für $\eta \in \mathbb{Z}_p^* = \langle a \rangle$.

Berechnung von Phase 1
Wie lassen sich in Phase 1 die diskreten Logarithmen $\log_\alpha p_i$ von den Primzahlen $p_i \in B$ berechnen?

Antwort:
Wähle $l \geq k$ Kongruenzen der Form $\alpha^{x_j} \equiv_p p_1^{\alpha_{j_1}} \cdot ... \cdot p_k^{a_{j_k}}$, d.h. $x_j \equiv_{p-1} a_{j_1} \log_\alpha p_1 + ... + a_{j_k} \log_\alpha p_k$.
Hierzu wählen wir ein zufälliges $x_j \in \mathbb{Z}_{n-1}$ und berechnen $\alpha^{x_j} \mod n$. Falls diese Zahl über der Faktorbasis B faktorisiert werden kann, lassen sich die Exponenten $a_{j_1}, ..., a_{j_k}$ leicht bestimmen.

Beispiel (Phase 1)
Seien $p = 10007$ (prim), $\alpha = 5$ und $B = \{2, 3, 5, 7\}$ (d.h. wir wissen sogar schon $l_3 = \log_\alpha 5 = 1$).
Wähle zufällig $x_1 = 4063, x_2 = 5136$ und $x_3 = 5985$.

$$\begin{array}{ll} & 5^{4063} \mod p = 42 = 2 \cdot 3 \cdot 7 \\ \Rightarrow & 5^{5136} \mod p = 54 = 2 \cdot 3^3 \\ & 5^{9865} \mod p = 189 = 3^3 \cdot 7 \end{array}$$

$$\begin{array}{ll} & l_1 + l_2 + l_4 \equiv_{p-1} 4063 \\ \Rightarrow & l_1 + 3l_2 \equiv_{p-1} 5163 \\ & 3l_2 + l_4 \equiv_{p-1} 9865 \end{array}$$

$$\Rightarrow \begin{array}{l} l_1 = 6578 \\ l_2 = 6190 \\ l_3 = 1 \\ l_4 = 1301 \end{array}$$

Phase 2
Wir wählen ein zufälliges $s \in \mathbb{Z}_{p-1}$ und berechnen $\gamma = \beta\alpha^s \mod p$. Falls sich γ über B zerlegen lässt, d.h. $\gamma = \prod_{i=1}^{k} p_i^{c_i}$, dann lässt sich $\log_\alpha \beta$ durch $\log_\alpha \beta = (\sum_{i=1}^{k} c_i l_k) - s$ bestimmen.

Vorlesung am 10.06.2008

Vorlesung am 12.06.2008

Lemma 2.1. *Sei $p \equiv_4 3$ und $S = 1$.*
Dann ist die Berechnung von $L_1(\beta)$ genau so schwer wie die Berechnung von $\log_\alpha \beta$.

Beweis:
$L_0(\beta)$ ist ohne Benutzung des L_1-Orakels effektiv berechenbar. $L_1(\beta)$ kann mit einer Orakelfrage β bestimmt werden.
Fragen: Wie lässt sich die Berechnung von $L_2(\beta)$ auf $L_1(\beta)$ reduzieren? Wie lässt sich die Berechnung von $L_n(\beta)$ mit Hilfe des Orakels auf die Berechnung von L_0 reduzieren?
Gesucht ist also $\alpha^{(a_l \dots a_1 a_0)_2} \mapsto a^{(a_l \dots a_1)_2}$.
Annahme: Wir können zusätzlich zu $\beta \in QN(p)$ ein γ mit $\gamma^2 = \beta$ bestimmen. Sei $\gamma = \sqrt{\frac{\beta}{\alpha^{a_0}}}$. Dann können wir γ und $-\gamma$ darauf testen, ob $\gamma = \alpha^{a_l \dots a_1}$ oder $-\gamma^{a_l \dots a_1}$ gilt. Für genau eines der beiden ist das niederwertigste Bit 1, weil $-1 = \alpha^{\frac{p-1}{2}}$ ($\frac{p-1}{2}$ ist ungerade) ist und $-\gamma = \gamma\alpha^{\frac{p-1}{2}}$ gilt. Durch das $L_1(\beta)$-Orakel können wir das letzte Bit von der bestimmten Wurzel mit dem vorletzten Bit von $\log_\alpha \beta$ vergleichen. Es muss also nur noch gezeigt werden, dass sich die Wurzeln leicht berechnen lassen:
Sei $\gamma = \beta^{\frac{p+1}{4}}$. Dann gilt $\gamma^2 \equiv \beta^{\frac{p+1}{2}} \equiv \alpha^{2a\frac{p+1}{2}} \equiv \alpha^{a(p+1)}$, d.h. $\gamma^2 \equiv \alpha^{a(p-1)} \cdot \alpha^{2a} \equiv 1 \cdot \beta = \beta$. □

TODO
Algorithmus

Beispiel (El-Gamal-Signatur)
Seien $p = 467, \alpha = 2$ und $a = 127$, d.h. $\beta = \alpha^a \mod p = 2^{127} \mod 467 = 132$. Mit $x = 100$ und $r = 213$ erhalten wir $\mathrm{ggT}(213, 466) = 1$ und $r^{-1} \equiv_{p-1} 431$.
$\Rightarrow \hat{k} = (\alpha, a, p) = (2, 127, 467), k = (\alpha, \beta, p) = (2, 132, 467)$
$\Rightarrow \gamma = \alpha^r \mod p = 2^{213} \mod 467 = 29, \delta = (x - a\gamma)r^{-1} \mod (p-1) = (100 - 127 \cdot 29) \cdot 431 \mod 466 = 51$
$\Rightarrow y = sig_{\hat{k}}(x, r) = (\gamma, \delta) = (29, 51)$
$\Rightarrow ver_k(x, y) = \beta^\gamma \gamma^\delta \mod p = 132^{29} \cdot 29^{51} \mod 467 = 189 = 2^{100} \mod 467 = \alpha^x \mod p$

Mögliche Angriffe

1. Zunächst Wahl von $\gamma \Rightarrow$ Berechnung von $\delta = \log_\gamma \alpha^x \beta^{-\gamma}$
   $\Rightarrow$ DLP $\Rightarrow$ schwer.
2. Zunächst Wahl von $\delta \Rightarrow$ Lösung für $\beta^\gamma \gamma^\delta \equiv \alpha^x$ gesucht
   In diesem Fall wurde bisher noch kein Lösungsalgorithmus gefunden. Die Schwierigkeit konnte jedoch auch noch nicht nachgewiesen werden.
3. Zunächst Wahl von γ und $\delta \Rightarrow$ Suche nach x
   $\Rightarrow$ DLP $\Rightarrow$ schwer

4. Simultane Suche von γ, δ und x

   (Existentielle Fälschung)

   Vorgehen: Seien α, β und p gegeben. Wir wählen $i, j \in \mathbb{N}$ mit $0 \leq i, j \leq p-2$ und setzen $\gamma = \alpha^{i\beta j} \mod p$.

   $\Rightarrow \alpha^x \equiv_p \beta^\gamma \alpha^{i\delta} \beta^{j\delta}$

   $\Leftrightarrow \alpha^{x-i\delta} \equiv_p \beta^{\gamma+j\delta} \Leftrightarrow x - i\delta \equiv_{p-1} a(\gamma + j\delta)$

   Nun reicht es aus, $x \equiv_{p-1} i\delta$ und $\gamma \equiv_{p-1} -j\delta$ zu finden.

   Für gegebene i, j ergibt sich:

   $\gamma = \alpha^{i\beta j} \mod p, \delta = -\gamma j^{-1} \mod (p-1)$ und $x = i\delta \mod (p-1)$.

   Es reicht also $\mathrm{ggT}(j, p-1) = 1$ aus.

Vorlesung am 17.06.2008

Vorlesung am 19.06.2008

Definition (El-Gamal-Signatur-Verfahren)
Wir setzen $X = \mathbb{Z}_p^*, Y = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}, \hat{k} = (p, \alpha, a)$ mit $\langle\alpha\rangle \mathbb{Z}_p^*, a \in \mathbb{Z}_{p-1}$ und $k = (p, \alpha, \beta$ mit $\beta = \alpha^a \mod p$.

Signaturerzeugung: Mit einer zufälligen Zahl $r \in \mathbb{Z}_{p-1}^*$ bestimmen wir:
$sig_{\hat{k}}(x, r) = (\gamma, \delta)$ mit $\gamma = \alpha^r \mod p$ und $\delta = (x - a\gamma)r^{-1} \mod (p-1)$.

Verifikation: $ver_k(x, \gamma, \delta) = (\beta^\gamma \gamma^\delta \equiv_p x)$.

2.4 Existentielle Fälschungen

Key only attack
Für $0 < i, j \leq p-2, \mathrm{ggT}(j, p-1) = 1$ gibt es die Fälschung (x, γ, δ) mit $\gamma = \alpha^i \beta^j \mod p$, $\delta = -\gamma j^{-1} \mod (p-1)$ und $x = -\gamma i j^{-1} \mod (p-1)$.

Weitere Sicherheitsbetrachtungen

1. Die zur Signierung benutzte Zufallszahl r muss geheim gehalten werden:
   $a = (x - \delta r)\gamma^{-1} \mod (p-1)$
2. r darf nicht zu Signierung mehrerer Dokumente verwendet werden. (siehe Übung)

Definition
Schnorr-Signaturverfahren Seien $p, q \in \mathbb{P}$ mit $p = mq + 1$, $\alpha \in \mathbb{Z}_p^*, \mathrm{ord}\,\alpha = q$ und $h : 2^* \to \mathbb{Z}_q$ eine Hashfunktion. Weiterhin seien$X = 2^*$ und $Y = \mathbb{Z}_q^2$.
Wir setzen $\hat{k} = (p, q, \alpha, a)$ für $a \in \mathbb{Z}_p$ und $k = (p, q, \alpha, \beta)$ mit $\beta = \alpha^a \mod p$.

Signaturerstellung:
Wir wählen $r \in \mathbb{Z}_q^*$ zufällig und berechnen $sig_{\hat{k}}(x, r) = (\gamma, \delta)$ mit $\gamma = h(x(\alpha^r \mod p)_2), \delta = r + a\gamma \mod q$. Hierbei sei $(Z)_2$ die Binärdarstellung der Zahl Z.

Verifikation: $ver(x, \gamma, \delta) = (h(x(\alpha^r \mod p)_2) = \gamma)$

Beispiel
Wir betrachten $q = 101, p = 87 \cdot q + 1 = 7879$ und $\alpha_0 = 3$ als Erzeuger von $\mathbb{Z}_p^*$.
Damit erhalten wir $\alpha = \alpha_0^{78} \mod p = 3^{78} \mod 7879 = 170$.
Seien nun $a = 75$ und $r = 50$.
$\Rightarrow \alpha^r \mod p = 170^{50} \mod 7879 = 2518, \beta = \alpha^{a_0} \mod p = 170^{75} \mod 7879 = 4567$
$\Rightarrow sig_{\hat{k}}(x, r) = (\gamma, \delta), h(x(2518)_2) := 96, \delta = r + a\gamma \mod q = 50 + 75 \cdot 96 \mod 101 = 74$

Verifikation:
$h(x(\underbrace{\alpha^{\delta}\beta^{-\gamma} \mod p}_{=170^{74} \cdot 4567^{-96} = 2518})_2) = 96 = \gamma$

Definition (Digital Signature Algorithm (DSA))
Es sei $q \in \mathbb{P}$ mit einer Länge von 160 Bits gewählt.
Weiterhin sei $p \in \mathbb{P}$ mit einer Länge $L \equiv_{64} 0$ und $512 \leq L \leq 1024$, sodass $p = mq + 1$.
Der Signierschlüssel ist $\hat{k} = (p, q, \alpha, a)$ mit $\alpha \in \mathbb{Z}_p^*$ und $\operatorname{ord} \alpha = a \in \mathbb{Z}_q$.
Der Verifikationsschlüssel ist $k = (p, q, \alpha, \beta)$ mit $\beta = \alpha^a \mod q$.
Wie beim Schnorr-Verfahren wird eine Hashfunktion verwendet, nämlich SHA-1. Ausgehend vom El-Gamal-Signatur-Verfahren setzen wir:
$\delta' = (x + a\gamma)r^{-1}$.
$\Rightarrow \gamma' = (\underbrace{\alpha^r \mod p}_{=\gamma}) \mod q, \delta' = ((x + a\gamma)^{-1} \mod (p - 1)) \mod q$
$\Rightarrow \beta^{\gamma'}\alpha^x \equiv_p \gamma^{\delta'}$
Wenn $\operatorname{ggT}(\delta', q) = 1$ gilt, so folgt $\gamma' = (\beta^{\gamma'\delta'^{-1}}\alpha^{x\delta'^{-1}}) \mod q$.

Signaturerstellung:
Wieder wählen wir ein zufälligs $r \in \mathbb{Z}_q^*$ und berechnen $sig_{\hat{k}}(x, r) = (\gamma, \delta)$ mit $\gamma = (\alpha^r \mod p) \mod q$ und $\delta = (SHA - 1(x) + a\gamma)r^{-1} \mod q$.
Im Fall $\gamma = 0$ oder $\delta = 0$ muss ein neues r gewählt werden.

Verifikation: $ver_k(x, \gamma, \delta) = ((\alpha^{SHA-1(x)\delta^{-1}}\beta^{\gamma\delta^{-1}}) \mod p) \mod q$

Bemerkung
Dieses Verfahren wird in den USA seit 1999 als Standard verwendet.

Vorlesung am 24.06.2008

Beispiel
Seien $q = 101, p = 78q + 1 = 7879$ und $\alpha_0 = 3$. Dann erhalten wir $\alpha = 3^{78} \mod 7879 = 170$.
$a = 75 \Rightarrow \beta = \alpha^a \mod 7879 = 4567$
$\Rightarrow \hat{k} = (p, q, \alpha, a) = (7879, 101, 170, 75)$
$k = (p, q, \alpha, \beta) = (7879, 101, 170, 4567)$

Signierung:
Um den Hashwert $SHA - 1(x) = 22$ zu signieren, wählt Alice eine Zufallszahl $r \in \mathbb{Z}_q^*$ (z.B. $r = 50$, d.h. $r^{-1} = 99 \mod 101$) und berechnet die Signatur $sig_{\hat{k}}(x, r) = (\gamma, \delta)$, wobei $\gamma = (\alpha^r \mod p) \mod q$ sowie $\delta = (SHA - 1(x) + a\gamma)r^{-1} \mod q$:
$\gamma = (170^{50} \mod 7879) \mod 101 = 2518 \mod 101 = 94$
$\delta = (22 + 75 \cdot 94) \cdot 99 \mod 101 = 97$

Verifikation:
$ver_k(x, (\gamma, \delta)) = ((\alpha^{e_1}\beta^{e_2} \mod p) \equiv_q \gamma)$ mit $e_1 = SHA - 1(x)\delta^{-1} \mod q$ und $e_2 = \gamma\delta^{-1} \mod q$:
$e_1 = 22 \cdot 97^{-1} \mod 101 = 22 \cdot 45 \mod 101 = 45$
$e_2 = 94 \cdot 25 \mod 101 = 27$
$\Rightarrow (170^{45} \cdot 4567^{27} \mod 7879) \mod 101 = 2518 \mod 101 = 94$

2.5 Elliptische Kurven über $\mathbb{R}$

Definition (Ellipitische Kurve)
Seien $a, b \in R$. Eine (*reellwertige*) *elliptische Kurve* E ist die Menge der Lösungen $(x, y) \in \mathbb{R}^2$ der Gleichung $y^2 = x^3 + ax + b$ zuzüglich des Punktes $\mathcal{O}$.
Im Fall $4a^3 + 27b^2 = 0$ heißt E *singulär*, sonst *nicht-singulär*.

Definition (Gruppenoperation auf elliptischen Kurven)
Auf den nicht singulären Punkten lässt sich eine Gruppenoperation wie folgt definieren: Gegeben sind zwei Punkte P und Q. Wir bilden die Gerade durch P und Q. Diese schneidet die elliptische Kurve in genau einem weiteren Punkt R (bei Tangenten zählen die Berühungspunkte doppelt). Das Spiegelbild $\overline{R}$ von R an der x-Achse ist der Punkt $P + Q$.
Das neutrale Element ist $\mathcal{O}$ (man kann diesen Punkt als unendlich weit nach oben verschoben betrachten). Es gilt also $P + \mathcal{O} = \mathcal{O} + P = P$ für alle $P \in E$.
Seien nun $P = (x_1, y_1) \neq \mathcal{O} \neq Q = (x_2, y_2)$. Dann definieren wir:

1. Fall: $x_1 \neq x_2$

   Sei $g = \{(x, y) \in \mathbb{R}^2 | y = \lambda x + \mu\}$ mit $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ und $\mu = y_2 - \lambda x_2 = y_1 - \lambda x_1$.

   Wir zeigen zuerst, dass $E \cap g = \{P, Q, R\}$ mit $R = (x_3, y_3)$ gilt.

   Für alle $(x, y) \in E \cap g$ gilt $x^3 - 4x = y^2 = (\lambda x + \mu)^2 = \lambda^2 x^2 + 2\lambda\mu x + \mu^2$. Dies ist äquivalent zu $p(x) = x^3 - \lambda^2 x^2 + (a - 2\lambda\mu)x + b - \mu^2 = 0$.

   Über $\mathbb{C}$ lässt sich $p(x)$ vollständig faktorisieren: $p(x) = (x - x_1)(x - x_2)(x - x_3)$. Weil x_1 und x_2 reell sind, muss auch x_3 reell sein.

   Durch Koeffizientenvergleich erhalten wir $x_1 + x_2 + x_3 = \lambda^2$, d.h. $x_3 = \lambda^2 - x_1 - x_2$. y_3 lässt sich dann durch einsetzen in die Geradengleichung (oder besser: in die Gleichung zur Berechnung von λ) bestimmen: $y_3 = \lambda(x_3 - x_1) + y_1$.

2. Fall: $x_1 = x_2, y_1 = -y_2$ Hier setzen wir $P + Q = \mathcal{O}$.

3. Fall: $P = Q$, $y_1 \neq 0$

   In diesem Fall sei t die Tangente an E in P. Die Steigung λ von t erhalten wir durch implizite Differentiation:

   $\lambda = \frac{dx}{dy} = \frac{-\frac{\partial F}{\partial x}(x_1, y_)}{\frac{\partial F}{\partial y}(x_1, y_1)} = \frac{3x_1^2 + a}{2y_1}$

   Sei $T(x, y)$ die Tangentialebene an $F(x, y) = y^2 - x^3 - ax - b$ im Punkt (x_1, y_1).

   Dann gilt $T(x, y) = cx + dy + e$ mit $c = \frac{\partial F}{\partial x}(x_1, y_1)$ und $d = \frac{\partial F}{\partial y}(x_1, y_2)$.

   t ist der Schnitt von $T(x, y)$ mit der (x, y)-Ebene: $(x, y) \in t \Leftrightarrow cx + dy + e = 0 \Leftrightarrow y = \frac{-cx - e}{d}$. Der Anstieg von t ist also $\frac{-cx}{d}$.

   Analog zum Fall 1 ergibt sich $x_3 = \lambda^2 - 2x_1$ und $y_3 = \lambda(x_3 - x_1) + y_1$.

Satz 2.1. *E bildet mit $\mathcal{O}$ als neutralem Element und $+$ als Verknüpfung eine abelsche Gruppe.*

Beweis:
Kein Beweis □

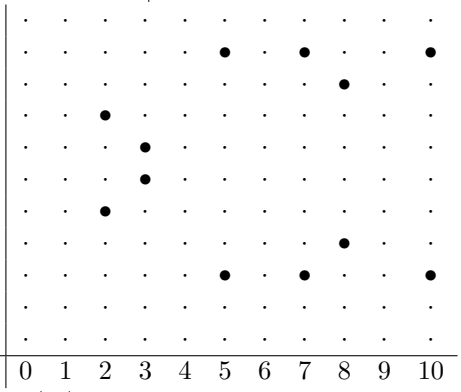
Bemerkung
Das Inverse eines Elements $P = (x, y)$ ist der Punkt $\overline{P} = (x, -y)$. Demnach sind genau die Punkte mit $y = 0$ sowie der Punkt $\mathcal{O}$ selbstinvers.

2.6 Elliptische Kurven über endlichen Körpern

Definition
Seien $3 \leq p \in \mathbb{P}$ und $a, b \in \mathbb{Z}_p$.
Dann heißt $E = \{(x,y) \in \mathbb{Z}_p^2 | y^2 \equiv_p x^3 + ax + b\} \cup \{\mathcal{O}\}$ *elliptische Kurve* über $\mathbb{Z}_p$. Im Fall $4a^3 + 27b^2 \equiv_p 0$ heißt E *singulär* und sonst *nicht singulär*.

Beispiel
$p = 11, y^2 = x^3 + x + 6$

x	0	1	2	3	4	5	6	7	8	9	10
$z = x^3 + x + 6$	6	8	5	3	8	4	8	4	9	7	4
$z \in QR_{11}$	−	−	+	+	−	+	−	+	+	−	+
y	−	−	4, 7	5, 6	−	2, 9	−	2, 9	3, 8	−	2, 9



Es gilt $|E| = 13$, d.h. $E \cong \mathbb{Z}_{13}$.
Sei $A = (2, 7)$.
Dann gilt $2A = (\underbrace{\lambda^2 - 2 - 2}_{=8^2-2-2=5}, \underbrace{\lambda(2-5) - 7}_{=2})$ mit $\lambda = (3x_1^2 + a)(2y_1)^{-1} = (3 \cdot 2^2 + 1)(2 \cdot 7)^{-1} = 8$.

Vorlesung am 26.06.2008

Satz 2.2. (Hasse) *Sei E eine elliptische Kurve über $\mathbb{Z}_p, 3 \leq p \geq \in \mathbb{P}$.*
Dann gilt für die Anzahl der Elemente von E: $p + 1 - 2\sqrt{p} \leq \#E \leq p + 1 + 2\sqrt{p}$.

Beweis:
Kein Beweis. □

Satz 2.3. *Sei E eine elliptische Kurve über $\mathbb{Z}_p, 3 \leq p \in \mathbb{P}$.*
Dann gilt $E \cong \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ mit $n_1 \geq 1 \leq n_2$ und $n_2 | \operatorname{ggT}(n_1, p-1)$.

Beweis:
Kein Beweis. □

Kompression der Punktdarstellung
Um die Punkte auf E kompakt darzustellen, definieren wir die folgende Funktion:
$PointCompress : E \setminus \{0\} \to \mathbb{Z}_p \times \mathbb{Z}_2$ mit $(x, y) \mapsto (x, y \mod 2)$.

Dekompressionsalgorithmus
$z = x^3 + ax + b \mod p$;
if $(z \in QR_p)\{$
  $y = \sqrt{z} \mod p$;
  if $(y \not\equiv_2 b)$
    $y = p - y$;
  return (x, y);
}

Definition (ECDSA)
Sei p eine Primzahl und sei E eine elliptische Kurve über $\mathbb{Z}_p$. Weiterhin sei $A \in E$ ein Punkt der Ordnung $q \in \mathbb{P}$, sodass das DLP in E zur Basis q schwer zu lösen ist.
Signierschlüssel: $\hat{k} = (E, p, q, A, m), m \in \mathbb{Z}_q$
Verifikationsschlüssel: $k = (E, p, q, A, B), B = mA$
Signaturerstellung: Wähle zufällig $r \in \mathbb{Z}_q^*$ und bestimme $sig_{\hat{k}}(x, r) = (y, z)$ mit $r \cdot A = (u, v)$, $y = u \mod q$ und $z = (SHA - 1(x) + my)r^{-1} \mod q$. Im Fall $y = 0$ oder $z = 0$ muss r neu gewählt werden.
Verifikation: $ver_k(x, y, z) = u \mod q = y$, wobei $e_1 A + e_2 B = (u, v) =, e_1 = SHA - 1(x)z^{-1} \mod q$ und $e_2 = yz^{-1} \mod q$.

Lemma 2.2. (Korrektheit) $(y, z) = sig_{\hat{k}}(x, r), r \in \mathbb{Z}_q, y \neq 0 \neq z \Rightarrow ver_k(x, y, z) = 1$

Beweis:
$e_1 = \underbrace{SHA - 1(x)}_{x'} z^{-1} \mod q, e_2 = yz^{-1} \mod q$
$e_1 A + e_2 B = (x'z^{-1})A + (yz^{-1}m)A = (x' + my)z^{-1}A = rA$ □

Beispiel
Wir betrachten wieder die elliptische Kurve aus dem vorherigen Beispiel ($p = 11, y^2 = x^3 + x + 6$).
Seien $A = (2, 7)$ und $m = 7$ mit $q = ord_E(A) = 13$
Damit erhalten wir $\hat{k} = (E, 11, 13, A, 7)$ und $k = (E, 11, 13, A, B)$ mit $B = 7 \cdot A = (7, 2)$.
Seien $x' = SHA - 1(x) = 4$ und $r = 3$.
$\Rightarrow rA = 3 \cdot (2, 7) = (8, 3) = (u, v)$
$\Rightarrow y = u \mod q = 8, z = (x' + my)r^{-1} \mod q = (4 + 7 \cdot 8)3^{-1} \mod 13 = 7$
$\Rightarrow sig_{\hat{k}} = (8, 7)$

Verifikation:
$e_1 = x'z^{-1} \mod q = 4 \cdot 7^{-1} \mod 13 = 8$
$e_2 = y \cdot z^{-1} = 8 \cdot 7^{-1} \mod 13 = 3$
$e_1 A + e_2 B = 8 \cdot (2, 7) + 3 \cdot (7, 2) = (8, 3)$
$u = 8 \equiv_q y \Rightarrow ver_k(x, 8, 7) = 1$

Vorlesung am 01.07.2008

2.7 Effiziente Berechnung von Vielfachen von Punkten auf E

Definition
$(c_{l-1}, ..., c_0) \in \{-1, 0, 1\}^l$ heißt *SBR-Darstellung* (signed binary representation) einer Zahl $c \in \mathbb{Z}$, falls $\sum_{i00}^{l-1} c_i 2^i = c$ ist.
Ist von je zwei benachbarten c_i's mindestens eines 0, so heißt $(c_{l-1}, ..., c_0)$ *NAF-Darstellung* von C.

Beispiel
$C = 11$ hat die SBR-Darstellungen $(0, 1, 0, 1, 1)$ und $(1, 0, -1, 0, -1)$, denn $1 + 2 + 8 = 11$ und $-1 - 4 - 16 = 11$.

Satz 2.4. *Jede Binärzahl hat eine eindeutige NAF-Darstellung.*

Beweis:
Übung. □

Algorithmus: DoubleAdd (klassische Variante)

```
Point DoubleAdd(P, c_{l-1}, ..., c_0) {
    Q = 0;
    for (i = l - 1; i >= 0; i--) {
        Q = 2Q;
        if (c_i == 1)
            Q = Q + P;
    }
    return Q;
}
```

Algorithmus: DoubleAddSub (für SBR-Darstellung)

```
Point DoubleAdd(P, c_{l-1}, ..., c_0) {
    Q = 0;
    for (i = l - 1; i >= 0; i--) {
        Q = 2Q;
        if (c_i == 1)
            Q = Q + P;
        else if (c_i == -1)
            Q = Q - P;
    }
    return Q;
}
```

Bemerkung (Geschwindigkeitsvergleich)
Da eine l-Bitzahl im Durchschnitt $\frac{l}{2}$ Nullen in Binärdarstellung und $\frac{2l}{3}$ Nullen in NAF-Darstellugn enthält, ist DoubleAddSub durchschnittlich um den Faktor $\frac{l+\frac{l}{2}}{l+\frac{l}{3}} = \frac{9}{8}$ schneller, d.h. wir haben einen Geschwindigkeitsgewinn von etwa 12.5%.

2.8 One-time Signatur (Lompat)

Sei $X = 2^n$ und $Y = U^n$ mit $l : U \hookrightarrow V$ eine injektive Einwegfunktion ist.
Signierschlüssel: $\hat{k} = (u_{i,b})_{i=1,...,n,b\in 2}$, wobei $u_{i,b}, (i,b) \in \{1,...,n\} \times 2$ $2n$ paarweise verschiedene Elemente aus U sind.
Verifikationsschlüssel: $k = (v_{i,b})_{i=1,...,n,b\in 2}$, wobei $v_{i,b} = f(u_{i,b})$.
Signaturerstellung: Sei $x = x_1...x_n \in 2^n$ gegeben.
$sig(\hat{k}, x) = \underbrace{u_1...u_n}_{=y}$, wobei $u_i = u_{i,x_i}$.
Verifikation: $ver(k, x, y) = (\forall i = 1, ..., n : f(u_i) = v_{i,x_i})$

Beispiel
Wir können als Einwegfunktion eine funktion $f : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ mit $f(u) = g^u \mod p$ und $\langle g \rangle = \mathbb{Z}_p^*$ verwenden. Wir wählen $p = 7879$ und $g = 3$. Weiterhin sei $n = 3$.
Der Signierschlüssel sei $\hat{k} = (\underbrace{5831}_{u_{1,0}}, \underbrace{803}_{u_{1,1}}, \underbrace{4285}_{u_{2,0}}, \underbrace{735}_{u_{2,1}}, \underbrace{2467}_{u_{3,0}}, \underbrace{6449}_{u_{3,1}})$.
Der Verifikationsschlüssel ist dann $k = (\underbrace{2009}_{v_{1,0}}, \underbrace{4672}_{v_{1,1}}, \underbrace{268}_{v_{2,0}}, \underbrace{3810}_{v_{2,1}}, \underbrace{4721}_{v_{3,0}}, \underbrace{5731}_{v_{3,1}})$.
$\Rightarrow sig(\hat{k}, \underbrace{110}_{=x}) = (u_1u_2u_3) = (803, 735, 2467)$
$\Rightarrow ver(k, 110, u_1u_2u_3) = 1$, da:
$f(u_1) = 3^{803} \mod p = 4672 = v_{1,1}$

$f(u_2) = 3^{735} \mod p = 3810 = v_{2,1}$
$f(u_2) = 3^{2467} \mod p = 4721 = v_{3,0}$

Definition (Algorithmus für den Sicherheitsbeweis)
Zum Nachweis der Sicherheit des Signaturverfahrens nehmen wir an, dass $f : U \to V$ eine Einwegpermutation ist (also bijektiv) ist und dass ein Algorithmus *Lampat-Fälschung* k existiert, der bei Eingaben eines Verifikationsschlüssels k eine nicht selektive Fälschung (x, y) mit $verk(k, x, y) = 1$ ausgibt. Wir betrachten den folgenden probabilistischen Las-Vegas-Algorithmus (d.h. wir geben nie etwas falsches aus):

```
int Lampat-Urbild(int v) {
    wähle zufällig paarweise verschiedene Funktionswerte v_{1,0}, ..., v_{n,1}, die ungleich v sind;
    v_{i,b} = v für ein zufällig gewähltes Paar (i, b);
    (x, y) =Lampat-Fälschung(k);
    if (x_j == a);
        return u_j;
    else
        return ?;
}
```

Satz 2.5. *Unter den genannten Voraussetzungen gibt Lampat-Urbild(v) für einen zufällig gewählten Funktionswert $v \in V$ mit Wahscheinlichkeit $\frac{1}{2}$ ein Urbild u mit $f(u) = v$ aus.*

Beweis:
Es reicht zu zeigen, dass die Wahrscheinlichkeit des Eintretens des ?-Falles kleiner oder gleich $\frac{1}{2}$ ist.
Seien $S = \binom{V}{2n}$ die Menge aller möglichen Verifikationsschlüssel k und $s = |S|$. Für $v \in V$ seien S_v die Menge aller $k \in S$, in denen v vorkommt, und T_v die Menge aller $k \in S_v$, bei deren Wahl der Algorithmus Lampat-Urbild erfolg hat.
Zu zeigen ist $p = \frac{1}{|V|} \sum_{v \in V} p_v \geq \frac{1}{2}$, wobei p_v die Erfolgswahrscheinlichkeit von Lampat-Urbild(v) ist. Mit $t_v = |T_v|$ und $s_v = |S_v|$ gilt $p_v = \frac{|T_v|}{|S_v|}$.

1. Es gilt $\sum_{v \in V} t_v = ns$.

   Für jedes der s möglichen Verifikationsschlüssel findet Lampat-Fälschung(k) genau n Urbilder, was der Gesamtzahl $\sum_{v \in V} t_v$ entspricht.

2. Für jedes $v \in V$ gilt $s_v|V| = 2ns$.

   Aus Symmetriegründen gilt $s' = s_v = s_{v'}$ für alle $v, v' \in V$. Zudem enthält jeder der s Verifikationsschlüssel $k \in S$ genau $2n$ Elemente von V.

   Daher folgt $2ns = \sum_{v \in V} s_v = s_v \sum_{v \in V} 1 = s_v|V|$.

Damit folgt $p = \frac{1}{|V|} \sum_{v \in V} p_v = \frac{1}{|V|} \sum_{v \in V} t_v s_v = \frac{1}{|V|s'} \sum_{v \in V} t_v = \frac{1}{2ns} \cdot ns = \frac{1}{2}$. □

Vorlesung am 03.07.2008

2.9 Verbindliche Signaturen (undeniable signatures)

Definition (Chaum van Antwerpe Signaturverfahren)
Sei $p = 2q + 1$ mit $p, q \in \mathbb{P}$ sodass das DLP in $\mathbb{Z}_p^*$ hinreichend schwer ist. Weiterhin seien $\alpha \in \mathbb{Z}_p^*$ ein Element mit $\text{ord}_{\mathbb{Z}_p^*}(\alpha) = q$, d.h. $\alpha = g^2, \langle g \rangle = \mathbb{Z}_p^*$ bzw. $G = \langle \alpha \rangle = QR_p$.
Wir setzen nun $X = Y = G$.
Signierschlüssel: $\hat{k} = (p, \alpha, a)$ mit $a \in \mathbb{Z}_q^*$

Verifikationsschlüssel: $k = (p, \alpha, \beta)$ mit $\beta = \alpha^a \mod p$
Signaturerstellung: $y = sig(\hat{k}, x) = x^a \mod p$
Verifikationsprotokoll: zwischen Alice (legaler Unterzeichner) und Bob (Verifizierer)

1. Bob wählt zufällig $e_1, e_2 \in \mathbb{Z}_q$ und sendet $c = y^{e_1}\beta^{e_2}$ an Alice.
2. Alice sendet $d = c^{a^{-1} \mod q} \mod p$ an Bob zurück.
3. Bob akzeptiert genau dann die Signatur, wenn $d \equiv_p x^{e_1}\alpha^{e_2}$.

Lemma 2.3. (Korrektheit) *Falls $y = x^a \mod p$ eine gültige Signatur ist und sich Alice und Bob an das Verifikationsprotokoll halten, dann akzeptiert Bob y als gültige Signatur für x.*

Beweis:

1. Wegen $\beta \equiv_p \alpha^a$ folgt $\beta^{a^{-1} \mod q} \equiv_p \alpha^{aa^{-1} \mod q} \equiv_p \alpha$.
2. Wegen $y \equiv_p x^a$ folgt $y^{a^{-1} \mod q} \equiv_p x^{aa^{-1} \mod q} \equiv_p x$.

$\Rightarrow d \equiv_p c^{a^{-1} \mod q} \equiv_p (y^{e_1}\beta^{e_2})^{a^{-1} \mod q} \equiv_p y^{a^{-1}e_1 \mod q}\beta^{a^{-1}e_2 \mod q} \equiv_p x^{e_1}\alpha^{e_2}$ □

Beispiel
Seien $p = 2 \cdot \underbrace{233}_{=q} + 1 = 467$ und $g = 2$, d.h. $\alpha = g^2 = 4$.
$\Rightarrow \hat{k} = (p, \alpha, a)$ mit $a = 101$
$\Rightarrow k = (p, \alpha, \beta)$ mit $\beta = \alpha^a \mod p = 4^{101} \mod 467 = 449$
$\Rightarrow$ Die Signatur von $x = 119 \in G = QR_p$ ist $y = sig(\hat{k}, x) = x^a \mod p = 119^{101} \mod 467 = 129$.
Verifikationsprotokoll für $x = 119$ und $y = 129$

1. Seien $e_1 = 38$ und $e_2 = 164$.
   $\Rightarrow c = y^{e_1}\beta^{e_2} \mod p = 129^{38} \cdot 449^{164} \mod 467 = 13$
2. $d = c^{a^{-1} \mod q} \mod p = 13^{101^{-1} \mod 233} \mod 467 = 9$
3. $x^{e_1}\alpha^{e_2} \mod p = 119^{38} \cdot 4^{164} \mod 467 = 9$

Satz 2.6. *Im Fall $y \not\equiv_p x^a$ akzeptiert Bob mit Wahrscheinlichkeit $\leq \frac{1}{q}$, selbst dann, wenn sich Alice nicht an das Verifikationsprotokoll hält.*

Beweis:
Da zu $y, \beta, c \in G$ und zu $e_1 \in \mathbb{Z}_q$ genau ein $e_2 \in \mathbb{Z}_q$ mit $c \equiv_p y^{e_1}\beta^{e_2}$ existiert, führen je q Paare $(e_1, e_2) \in \mathbb{Z}_q^2$ auf das selbe c.
Behauptung: Im Fall $y \not\equiv_p x^a$ erfüllt für jedes $d \in G$ genau ein Paar $(e_1, e_2) \in \mathbb{Z}_q^2$ die Kongruenzen $c \equiv_p y^{e_1}\beta^{e_2}$ und $d \equiv_p x^{e_1}\alpha^{e_2}$.
Beweis der Behauptung: Seien $c = \alpha^i$, $d = \alpha^j$, $x = \alpha^k$ und $y = \alpha^l$. Dann sind die beiden Äquivalenzen aus der Behauptung äquivalent zu $i \equiv_q le_1 + ae_2$ und $j \equiv_q ke_1 + e_2$.
Wegen $y \not\equiv_p x^a \Leftrightarrow l \not\equiv_q ka \Leftrightarrow \det \begin{pmatrix} l & a \\ k & 1 \end{pmatrix} \neq 0$ hat dieses Gleichungssystem eine eindeutige Lösung in $\mathbb{Z}_q$.
Damit ist die Behauptung bewiesen. Aus der Behauptung folgt dann auch der Satz. □

Vorlesung am 08.07.2008

Problem
Alice könnte nun behaupten, dass eine (möglicherweise echte) Signatur falsch ist und nicht von ihr stammt. Um dies nachweisen zu können, gibt es das folgende Ableugnungsprotokoll.

Definition
Ableugnungsprotokoll

1. Bob wählt zufällig $e_1, e_2 \in \mathbb{Z}_q$ und sendet $c = y^{e_1}\beta^{e_2} \mod q$ an Alice.
2. Alice sendet $d = c^{a^{-1}} \mod p$ an Bob zurück.
3. Bob testet, ob $d \not\equiv_p x^{e_1}\alpha^{e_2}$ gilt.
4. Bob wählt zufällig $f_1, f_2 \in \mathbb{Z}_q$ und sendet $C = y^{f_1}\beta^{f_2} \mod q$ an Alice.
5. Alice sendet $D = C^{a^{-1}} \mod p$ an Bob zurück.
6. Bob testet, ob $D \not\equiv_p x^{f_1}\alpha^{f_2}$ gilt.
7. Bob erkennt y als falsche Signatur an, wenn einer der beiden Tests in Zeilen 3 und 6 positiv ausfällt und zusätzlich folgende Konsistenzbedingung erfüllt ist:
   $(d\alpha^{-e_2})^{f_1} \equiv_p (D\alpha^{-f_2})^{e_1}$
   (Der dritte und sechste Schritt allein reichen noch nicht aus, um nachzuweisen, dass die Signatur falsch ist, da Alice ja absichtlich falsche d und D an Bob gesendet haben könnte.)

Satz 2.7. *Im Fall $y \not\equiv_p x^a$ erkennt Bob y mit einer Wahrscheinlichkeit von mindestens $1 - \frac{1}{q^2}$ als falsch an, falls sich beide an das Ableugnungsprotokoll halten.*

Beweis:
Mit Hilfe des vorhergehenden Satzes erhalten wir als Wahrscheinlichkeit für das zweimalige Gelingen der Verifikation (d.h. die Schritte 1-6) wegen der unabhängigen Wahl von e_1, e_2 und f_1, f_2 höchstens $\frac{1}{q^2}$. D.h. die Wahrscheinlichkeit, dass die Tests der Zeilen 3 und 6 fehlschlagen, ist höchstens $\frac{1}{q^2}$.
Wenn sich beide an das Protokoll halten, dann ist die Konsistenzbedingung mit Wahrscheinlichkeit 1 (hier also immer) erfüllt:

$$(d\alpha^{e_2})^{f_1} \equiv_p \left((y^{e_1}\beta^{e_2})^{a^{-1}}\alpha^{-e_2}\right)^{f_1} \equiv_p y^{e_1 a^{-1} f_1}\alpha^{(e_2-e_2)f_1} \equiv_p y^{e_1 f_1 a^{-1}}$$
$$\equiv_p y^{f_1 a^{-1} e_1}\alpha^{(f_2-f_2)e_1} \equiv_p \left((y^{f_1}\beta^{f_2})^{a^{-1}}\alpha^{-f_2}\right)^{e_1} \equiv_p \left(D\alpha^{-f_2}\right)^{e_1}$$

□

Satz 2.8. *Im Fall $y \equiv_p x^a$ erkennt Bob y mit einer Wahrscheinlichkeit von höchstens $\frac{1}{q}$ als falsch an, selbst wenn Alice sich nicht an das Ableugnungsprotokoll hält.*

Beweis:
Bob erkennt y nur dann als falsch an, wenn die beiden folgenden Bedingungen erfüllt sind:

1. $d \not\equiv_p x^{e_1}\alpha^{e_2}$ oder $D \not\equiv_p x^{f_1}\alpha^{f_2}$
2. $(d\alpha^{-e_2})^{f_1} \equiv_p (D\alpha^{-f_2})^{e_1}$

Annahme: Bob erkennt die Signatur als falsch an.
Wir betrachten den Fall $d \not\equiv_p x^{e_1}\alpha^{e_2}$. (Der Fall $D \not\equiv_p x^{f_1}\alpha^{f_2}$ kann analog behandelt werden.)
Seien e_1, e_2 und d so gewählt, dass $d \not\equiv_p x^{e_1}\alpha^{e_2}$. Alice muss nun zu gegebenem C ein D finden, sodass die Konsistenzbedingung erfüllt ist.
Durch C wird jedem f_1 genau ein f_2 mit der Eigenschaft $C = y^{f_1}\beta^{f_2} \mod p$ zugeordnet.
Behauptung:
Für jedes $D \in G$ existiert genau ein Paar (f_1, f_2) mit $C \equiv_p y^{f_1}\beta^{f_2}$ und $(d\alpha^{-e_2})^{f_1} \equiv_p (D\alpha^{-f_2})^{e_1}$.
Offensichtlich folgt aus der Behauptung auch sofort der Satz.
Beweis der Behauptung:

Z	β	x	y	C	D	$u = d\alpha^{-e_2}$
$\log_\alpha Z$	a	k	ak	i	j	l

Dabei gilt $l + e_2 \not\equiv_q ke_1 + e_2$, wegen $\alpha^{l+e_2} \equiv_p d \not\equiv_p x^{e_1}\alpha^{e_2}$.

Wir erhalten also das Gleichungssystem $ak \cdot f_1 + a\dot{f}_2 \equiv_q i$ und $l \cdot f_1 + e_1 \cdot f_2 \equiv_q e_1 j$, d.h.

$$\begin{pmatrix} ak & a \\ l & e_1 \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \equiv_q \begin{pmatrix} i \\ e_1 j \end{pmatrix}.$$

Wegen $\det \begin{pmatrix} ak & a \\ l & e_1 \end{pmatrix} = ake_1 - la = a(ke_1 - l)$ und $ke_1 \not\equiv_q l$ und $a \in \mathbb{Z}_q^*$ gilt $\det \begin{pmatrix} ak & a \\ l & e_1 \end{pmatrix} \neq 0$, d.h. das Gleichungssystem hat genau eine Lösung. □

2.10 Fail-Stop-Signaturen

Diese Signaturen erlauben es Alice (d.h. dem legalen Signaturerzeuger) für den Fall, dass ihr Signierschlüssel $\hat{k}$ geknackt wird (fail), dies zu beweisen und damit alle von ihr mit $\hat{k}$ geleisteten Unterschriften zu widerrufen (stop).
Genauer: Alice kann mit hoher Wahrscheinlichkeit beweisen, dass eine von einem Gegner erzeugte gültige Signatur nicht von ihr stammt. Dies setzt natürlich voraus, dass es für ein Dokument mehrere gültige Signaturen gibt.
Wir verwenden ein Verfahren, dass eine Information in der Signatur kodiert, die selbst bei Kenntnis des Signierschlüssels nicht bekannt ist.

Definition (van Heyst-Pederson-Signaturverfahren)
Seien $q = 2p + 1$ mit $p, q \in \mathbb{P}$ und $\alpha \in \mathbb{Z}_p^*$ mit $\text{ord}_p(\alpha) = q$. Weiterhin seien $\beta = \alpha^{a_0} \mod p$ und $G = \langle \alpha \rangle = QR_p$.
p, q, α, β werden von einer vertrauenswürdigen Instanz generiert und bekanntgegeben. Das a_0 wird jedoch geheim gehalten.
Wir verwenden $X = \mathbb{Z}_q$ und $Y = \mathbb{Z}_q^2$.
Der Signierschlüssel $\hat{k} = (a_1, a_2, b_1, b_2) \in \mathbb{Z}_p^4$ kann von dem Teilnehmer (ohne Mitwirkung der vertrauenswürdigen Instanz) gewählt werden.
Der Verifikationsschlüssel ist $\hat{k} = (\gamma_1, \gamma_2) = (\alpha^{a_1}\beta^{a_2}, \alpha^{b_1}\beta^{b_2}) \in G^2$.
Signaturerstellung: $sig(\hat{k}, x) = (y_1, y_2) = (a_1 + xb_1 \mod q, a_2 + x_b \mod q)$.
Verifikation: $ver(k, x, y_1, y_2) = (\gamma_1\gamma_2^x \equiv_p \alpha^{y_1}\beta^{y_2})$

Vorlesung am 10.07.2008

Lemma 2.4. (Korrektheit) $(y_1, y_2) = (a_1 + xb_1 \mod q, a_2 + xb_2 \mod q) \Rightarrow ver(k, x, y_1, y_2) = 1$

Beweis:
$\gamma_1\gamma_2^x = \alpha^{a_1}\beta^{a_2}\alpha^{xb_1}\beta^{xb_2} \equiv_p \alpha^{a_1+xb_1}\beta^{a_2+xb_2} \equiv_p \alpha^{y_1}\beta^{y_2}$ □

Lemma 2.5. *Sei S die Menge aller Paare $(\hat{k}, k) \in \mathbb{Z}_p^4 \times G^2$ mit $\hat{k} = (a_1, a_2, b_1, b_2) \in \mathbb{Z}_p^4$ und $k = (\alpha^{a_1}\beta^{a_2}, \alpha^{b_1}\beta^{b_2}) \in G^2$. Weiter sei $S(k) = \{\hat{k} | (\hat{k}, k) \in S\}$.*
Dann gilt:

1. $|S(k)| = p^2$
2. $\hat{k}, \hat{k}' \in S(k), (y_1, y_2) = sig(\hat{k}, x), y' = (y_1', y_2') = sig(\hat{k}', x) \Rightarrow ver(k, x, y) = ver(k, x, y') = 1$

Beweis:

1. Es muss $\alpha^{a_1}\beta^{a_2} \equiv_q \gamma_1$ und $\alpha^{b_1}\beta^{b_2} \equiv_q \gamma_2$ gelten.

   $\Rightarrow a_1 + a_0 a_2 \equiv_p c_1$ und $b_1 + a_0 b_2 \equiv_p c_2$

   $\Rightarrow$ Zu jedem Paar (a_2, b_2) gibt es jeweils mindestens p Möglichkeiten für a_1 und b_1 um die Gleichungen zu erfüllen.

2. Trivial

□

Lemma 2.6. *Seien* $y=(y_1,y_2)=sig(\hat{k},x)$ *und* $\hat{k}\in S(k)$*. Dann gilt* $|\underbrace{\{\hat{k}' \in S(k)|sig(\hat{k}',x)=y\}}_{=S(k,x,y)}| = p.$

Beweis:
Sei $k=(\gamma_1,\gamma_2)$.
Dann ist $\hat{k}'=(a_1,a_2,b_1,b_2)\in S(k,x,y)$ äquivalent zu $\alpha^{a_1}\beta^{a_2}\equiv_q \gamma_1, \alpha^{b_1}\beta^{b_2}\equiv_q \gamma_2$, $a_1+xb_1\equiv_p y_1$ und $a_2+xb_2\equiv_p y_2$.
Um die Anzahl der Lösungen zu bestimmen, bestimmen wir den Rang der Matrix $\begin{pmatrix}1&a_0&0&0\\0&0&1&a_0\\1&0&x&0\\0&1&0&x\end{pmatrix}$.
Seien $r_1,...,r_4$ die 4 Zeilen der Matrix. Dann gilt $r_1=r_3+a_0r_4-xr_2$. Weiterhin sind die Zeilen 2 bis 4 linear unabhängig, womit wir den Rang 3 erhalten. Demnach folgt $|S(k,x,y)|=p$. □

Lemma 2.7. *Sei* $\hat{k}\in S(k)$ *und es gelte* $sig(\hat{k},x)=y$*. Weiterhin sei* (x',y') *ein Paar mit* $x'\neq x$ *und* $ver(k,x',y')=1$.
Dann gilt $|\underbrace{\{\hat{k}' \in S(k)|sig(\hat{k}',x)=y \wedge sig(\hat{k}',x')=y'\}}_{=S(k,x,y,x',y')}| \leq 1.$

Beweis:
Es gilt $\hat{k}'\in S(k,x,y,x',y')$ genau dann, wenn:
$a_1+a_0a_2\equiv_p c_1$
$b_1+a_0b_2\equiv_p c_2$
$a_1+xb_1\equiv_p y_1$
$a_2+xb_2\equiv_p y_2$
$a_1+x'b_1=y_1'$
$a_2+x'b_2=y_2'$
Dieses Gleichungssystem ist äquivalent zu $\begin{pmatrix}1&a_0&0&0\\0&0&1&a_0\\1&0&x&0\\0&1&0&x\\1&0&x'&0\\0&1&0&x'\end{pmatrix}\cdot\begin{pmatrix}a_1\\a_2\\b_1\\b_2\end{pmatrix}=\begin{pmatrix}c_1\\c_2\\y_1\\y_2\\y_1'\\y_2'\end{pmatrix}$.
Weil die 2., 3., 4. und 6. Zeile der Matrix linear unabhängig sind (es gilt $x\neq x'$), ist der Rang der Matrix 4. □

Vorlesung am 15.07.2008

Lemma 2.8. *Sei* $\hat{k}\in S(k)$ *und es gelte* $sig(\hat{k},x)=y$*. Dann gilt* $|\{\hat{k}'\in S(k)|sig(\hat{k}',x)=y\}|=q$.

Beweis:
Es reicht zu zeigen, dass im Fall $ver(k,x',y')=1$ das Gleichungssystem aus dem vorhergehenden Lemma lösbar ist: $ver(k,x',\underbrace{y'}_{=(y_1',y_2')})=1 \Rightarrow \gamma_1\gamma_2^{x'}\equiv_p \alpha^{y_1'}\beta^{y_2'} \Rightarrow c_1+x'c_2\equiv_q y_1'+a_0y_2'$
$\Rightarrow y_1'\equiv_q c_1+x'c_2-a_0y_2'$
Wegen $ver(k,x,y)=1$ folgt außerdem $y_1\equiv_q c_1+xc_2-a_0y_2$.
Wegen $r_5=r_1+x'r_2-a_0r_6$ und $r_3=r_1+xr_2-a_0r_4$ (dabei sind $r_1,...,r_6$ die 6 Zeilen der Matrix aus dem vorhergehenden Lemma) folgt, dass jede Lösung von $\begin{pmatrix}r_1\\r_2\\r_4\\r_6\end{pmatrix}\cdot\begin{pmatrix}a_1\\a_2\\b_1\\b_2\end{pmatrix}=\begin{pmatrix}c_1\\c_2\\y_2\\y_2'\end{pmatrix}$ auch

Lösungen des gesamten Gleichungssystems sind. (Hierfür benötigen wir, dass die rechten Seiten des Gleichungssystems die selben linearen Abhängigkeiten erfüllen.) □

Korollar 2.1. *Seien $x \neq x'$ und $ver(k, x', y') = 1$.*
Dann gilt $\mathbf{P}_{\hat{k} \in \mathbb{Z}_q^4}(\underbrace{sig(\hat{k}', x') = y'}_{A} \mid \underbrace{sig(\hat{k}', x) = y, \hat{k}' \in S(k)}_{B}) = \frac{1}{q}$, d.h. für jedes von einem Gegner bei Kenntnis von $k, x, y = sig(\hat{k}, x)$ generierte Paar (x', y') mit $x' \neq x$ und $ver(k, x', y') = 1$ ist die Wahrscheinlichkeit, dass der "richtige" Schlüssel $\hat{k}$ auch $sig(\hat{k}, x') = y'$ liefert, genau $\frac{1}{q}$.

Beweis:
Das Ereignis B tritt genau dann ein, wenn $\hat{k}' \in S(k, x, y)$ gilt.
Es gilt also $\mathbf{P}(A|B) = \frac{\mathbf{P}(A \cap B)}{\mathbf{P}(B)} = \frac{\frac{1}{q^4}}{\frac{q}{q^4}} = \frac{1}{q}$. □

Korollar 2.2. *Sei $S(k, x, y) = \{\hat{k}_1, ..., \hat{k}_q\}$. Dann gilt $|\{sig(\hat{k}_i, x') | i = 1, ..., q\}| = q$, falls $x \neq x'$ gilt.*

Beweis:
Im Fall $sig(\hat{k}_i, x') = y' = sig(\hat{k}_j, x')$ mit $i \neq j$ wären die Signierschlüssel $\hat{k}_i$ und $\hat{k}_j$ in $S(k, x, y, x', y')$ enthalten. Diese Menge enthält jedoch höchstens ein Element. Daher kann dieser Fall nicht eintreten, d.h. es gilt die Behauptung des Korollars. □

Funktionsweise des Fail-Stop-Mechanismus
Wie kann Alice bei Vorlage (x', y'') mit $ver(k, x', y'') = 1$ und $y'' \neq y' = sig(k, x')$ beweisen, dass diese Signatur y'' nicht von ihr erzeugt wurde?

Antwort: Wir gehen davon aus, dass Alice wegen ihrer beschränkten Rechenressourcen nicht in der Lage ist, $a_0 = \log_\alpha \beta$ zu berechnen. Mit der zusätzlichen Information (k, x', y'') kann sie jedoch a_0 effizient bestimmen.
Wegen $ver(k, x', y') = 1 = ver(k, x', y'')$ folgt:
$\alpha^{y_1' + a_0 y_2'} \equiv_p \alpha^{y_1'} \beta^{y_2'} \equiv_p \gamma_1 \gamma_2^{x'} \equiv_p \alpha^{y_1''} \beta^{y_2''} \equiv_p \alpha^{y_1'' + a_0 y_2''}$
$\Leftrightarrow y_1' + a_0 y_2' \equiv_q y_1'' + a_0 y_2''$
$\Rightarrow a_0 \equiv_q (y_1'' - y_1')(y_2' - y_2'')^{-1}$

Beispiel
Wir setzen $p = 2q + 1 = 3467$ mit $q = 1733$ und $\alpha = 4$ mit $\operatorname{ord} \alpha = q$.
Die vertrauenswürdige Instanz (TTP - trusted third party) wählt nun $a_0 \in \mathbb{Z}_q^*$, z.B. $a_0 = 1567$, berechnet $\beta = \alpha^{a_0} \mod p = 4^{1567} = 514$ und gibt p, q, α und β bekannt.
Alice wählt nun den Signierschlüssel $\hat{k} = (a_1, a_2, b_1, b_2) = (888, 1024, 768, 999)$. Der Verifikationsschlüssel ist dann $k = (\gamma_1, \gamma_2) = (\alpha^{a_1} \beta^{a_2}, \alpha^{b_1} \beta^{b_2}) = (3405, 2281)$.
Wird Alice mit einem vom Gegner gegebenen Paar $(x', y'') = (x', y_1'', y_2'') = (3383, 822, 55)$ konfrontiert, das wegen $\gamma_1 \gamma_2^{x'} = 3405 \cdot 2281^{3383} \equiv_p 2282 \equiv_p 4^{822} \cdot 515^{55} = \alpha^{y_1''} \beta^{y_2''}$ die Verifikationsbedingung erfüllt, so berechnet Alice zunächst

$$y' = (y_1', y_2') = sig(\hat{k}, x') = (a_1 + x' b_1 \mod q, a_2 + x' b_2 \mod q) = (1504, 1291) \neq (822, 55).$$

Damit erhält sie $a_0 = (y_1'' - y_1')(y_2' - y_2'')^{-1} \mod q = (822 - 1504)(1291 - 55)^{-1} \mod q = 1567$.

Vorlesung am 17.07.2008

Kapitel 3

Zero Knowledge Beweise

3.1 Interaktive Beweise

Herkömmliche Beweise entsprechen der Klasse NP, d.h.:
$\mathcal{A} \in NP$, falls ein effizienter Verifizierer V existiert, mit $\forall x \in \mathcal{A} : \exists y : V(x, y) = 1$ Vollständigkeit
und $\forall x \notin \mathcal{A} : \forall y : V(x, y) = 0$ Korrektheit
(V, P) heißt (*nicht interaktives*) *Beweissystem* für $\mathcal{A}$, falls $\forall x \in \mathcal{A} : (V, P)(x) = 1$: P erzeugt einen Beweis für x und V verifiziert diesen Beweis. Das Beweissystem ist nicht interaktiv, weil der Informationsfluss nur von P zu V stattfindet.
Weiterhin gilt $\forall x \notin \mathcal{A} P' : (V, P')(x) = 0$

Ergänzung zu Interaktiven Beweisen
Wir erlauben nun, während des Beweisvorgangs Informationen (und Fragen) zwischen V und P auszutauschen. Damit erhalten wir ein *deterministisches interaktives Beweissystem*.

Hinzunahme von Nichtdeterminismus
Lassen sich mit diesem Verfahren Sprachen außerhalb von NP verifizieren? Dabei soll die Anzahl der Runden und die Komplexität des Verifizierers polynomiell in der Eingabelänge x beschränkt sein. P kann jedoch beliebige Komplexität haben.

Antwort: Nein - die Interaktion allein bringt nichts, weil die Komplexität des Beweisers P beliebig sein kann, kann dieser alle Reaktionen des Verifizierers vorausberechnen und die Antworten gleich beim ersten Mal auf das Band schreiben.

Daher erhält jede der Turingmaschinen P und V nun die Möglichkeit Zufallsbits zu verwenden.

Definition
Eine Sprache $\mathcal{A}$ hat ein interaktives Beweissystem (IBS), falls interaktive Turingmaschinen (ITM) V und P existieren, sodass V polynomiell (in der Eingabe x) zeitbeschränkt ist und für alle Eingaben x gilt:
$x \in A \Rightarrow \mathbf{P}((V, P)(x) = 1) = 1$
$x \notin A \Rightarrow \forall P' : P((V, P')(x) = 1) \leq \frac{1}{2}$
(Wir können die zweite Wahrscheinlichkeit durch Wiederholungen beliebig verkleinern.)

Satz 3.1. *Genau die Sprachen in $PSPACE$ haben ein IBS.*

Beweis:
Kein Beweis □

Bemerkung
Wenn man mehr als einen Prover P zulässt, erhält man genau die Sprachen, die in $NEXP$ liegen. Zwei Beweiser sind dabei genau so mächtig wie polynomiell viele Beweiser.

Beispiel (Komplement der Graphenisomorphie)
Eingabe sind G_1 und G_2 mit $v(G_1) = v(G_2) = n$.
Ein IBS für das Komplement der Graphenisomorphie ist:
V wählt zufällig $i \in \{1, 2\}$ sowie eine Permutation $\pi \in S_n$ und berechnet $H = \pi(G_i)$. V sendet diesen Graphen H nun an P und fragt nach der gewählten Zufallszahl i.
P sendet nun 1 im Fall $H \cong G_1$ und sonst 2 an V zurück.
V akzeptiert nun genau dann, wenn $i = j$ gilt.

Wenn $G_1 \ncong G_2$ gilt, so findet der Beweiser immer das gewählte Zufallsbit heraus.
Bei Isomorphie kann er sich nicht entscheiden, da beide Graphen zu H isomorph sind. Durch die zufällige Wahl von i ist die Akzeptanzwahrscheinlichkeit höchstens $\frac{1}{2}$.

Beispiel (Graphenisomorphie)
Um die Isomorphie zweier Graphen nachzuweisen, kann P einfach einen Isomorphismus zwischen G_1 und G_2 angeben.

3.2 Zero-Knowledge

Es soll nun vermieden werden, dass der Tester durch die Informationen, die ihm der Beweiser liefert, selbst eine weitere Instanz überzeugen kann. Daher darf der Beweiser keine Informationen an den Tester weitergeben, die diesem helfen könnten, selbst einen Beweis zu finden.

Definition
Zero-Knowledge IBS für Graphenisomorphie Eingabe: G_1, G_2 mit $v(G_1) = v(G_2) = n$

1. P wählt eine zufällige Permutation $\pi \in S_n$, berechnet $H = \pi(G_1)$ und sendet H an V.
2. V wählt eine zufällige Zahl $i \in \{1, 2\}$ und fordert eine Bijektion zwischen G_i und H an.
3. P berechnet eine Bijektion $\sigma : G_i \to H$ und sendet dieses an V.
4. V akzeptiert genau dann, wenn $\sigma(G_i) = H$ gilt.

Bemerkung
Das dieses Verfahren ein interaktives Beweissystem ist, ist leicht einzusehen.
Die Sicht von V ist G_1, G_2, H, i und σ. Wir bezeichnen diese als $view(V, P, G_1, G_2)$.

Definition (perfekte Zero-Knowledge-Eigenschaft (informell) - PZK)
Für jeden effizienten Verfizierer V^* existiert eine effiziente probabilistische Turingmaschine S (Simulator), sodass für alle Eingaben $x \in A$ gilt:
S produziert bei Eingabe x die Sicht von $(V^*, P)(x)$ (auch als $view(V^*, P, x)$ bezeichnet), wobei nur Ausgaben $y \neq ?$ von $S(x)$ berücksichtigt werden, d.h. die bedingte Verteilung der Zufallsvariablen $S(x)$ unter der Bedingung $S(x) \neq ?$ stimmt mit der Verteilung von $view(V^*, P, x)$ überein.

Lemma 3.1. *Das obige Verfahren zum Nachweis der Graphenisomorphie hat die Zero-Knowledge-Eigenschaft.*

Beweis:
Kein Beweis. □

Index