# Vorlesungen „Kryptographie I und II”
# Bei Prof. Dr. Schnorr
# WS1999/2000 - SS2000

Michael Jaeger (michael.jaeger@in-flux.de)

17. Oktober 2002 ($Revision: 1.5 $)

Dies Mitschrift entstand während der Vorbereitung für meine Diplomprüfung in Kryptographie I und II. Für Vollständigkeit oder Korrektheit kann ich nicht garantieren, bitte jedoch um aktive Mithilfe bei der Gestaltung dieses Skriptes. Sollten Fehler gefunden werden, so bitte ich darum, mir diese mitzuteilen, damit ich beheben kann.

Der Inhalt dieses Skriptes basiert auf den handschriftlichen Unterlagen der Vorlesungen Kryptographie I und II von Prof. Dr. C. P. Schnorr aus den Jahren 1999/2000.

Mein Dank geht vor allem an das LyX-Team, durch deren Programm dieses Skript erst möglich wurde. Die Software existiert für diverse Plattformen und kann unter der Adresse `http://www.lyx.org` bezogen werden.

Michael Jaeger, Frankfurt am Main, der 17. Oktober 2002.

# Inhaltsverzeichnis

# Teil 1

# Kryptographie I

KAPITEL 1

# Diskreter Logarithmus und Protokolle

In diesem einführenden Kapitel werden zum Einen kurz die mathematischen Grundlagen der Kryptographie (diskreter Logarithmus, verschiedene DL-Algorithmen) und zum Anderen konkrete Krypto-Verfahren (Diffie-Hellmann, ElGamal, verschiedene Signatur-Verfahren) präsentiert.

Während Verschlüsselungs-Algorithmen die Privatheit sicherstellen, also das Abhören der Kommunikation durch Dritte verhindern sollen, dienen Signaturen dazu die Integrität und Authentizität einer Nachricht überprüfbar zu machen, ohne diese unbedingt zu verschlüsseln. Da Signaturen in der Regel zusätzlich zu der eigentlichen Nachricht verschickt werden, bedeuten sie ein zusätzliches Datenaufkommen und sollen deshalb möglichst klein sein. Hier tun sich besonders die Schnorr-Signaturen hervor, weil sie sowohl kurz sind als auch zum Großteil offline berechnet werden können.

Bei den vorgestellten Verschlüsselungs-Verfahren sind besonders Shamir's „Schlüsseltransport mit Rücklauf" und die ElGamal-Verschlüsselungen zu erwähnen. Bei Ersterem können zwei Parteien verschlüsselte Nachrichten mittels einem symmetrischen Verschlüsselungsverfahren austauschen ohne vorher einen gemeinsamen Schlüssel zu vereinbahren (zu diesem Zweck könnte man das Diffie-Hellmann-Verfahren einsetzen) und bei Letzterem sind die Cypher-Texte randomisiert, was bedeutet, dass ein und derselbe Klartext zweimal verschlüsselt zwei unterschiedliche Schlüsseltexte ergibt!

Nach der Lektüre sollten die mathematische Grundlagen, die grundlegendsten Verfahren bei Signaturen (ElGamal, Schnorr, DSA, Nyber-Rueppel) und das ElGamal-Verfahren verstanden worden sein. Ausserdem sollten die verschiedenen Formen von Angriffen auf Verschlüsselungen (CA, CPA, CCA) verinnerlicht sein.

## 1.1. Grundlagen

DEFINITION 1.1.1. Die **Exponentialfunktion**

$$\begin{aligned} \exp_g : \quad & \mathbb{Z}_q \to G \\ & a \mapsto g^\alpha \end{aligned}$$

ist wohldefiniert wegen $g^q \equiv 1 \mod q$, $g^{a+tq} \equiv g^a \mod q$. Die Exponentialfunktion ist bijektiv.

Damit ergibt sich der diskrete Logarithmus:

DEFINITION 1.1.2. Der **diskrete Logarithmus** (**DL**) ist definiert durch

$$\begin{aligned} \log_g : \quad & G \mapsto \mathbb{Z}_q \\ & \log_g = \exp_g^{-1} \end{aligned}$$

damit ist

$$\log_g h \equiv x \mod q \quad \Leftrightarrow \quad h \equiv g^x \mod q.$$

Unter dem **DL-Problem** wird die Aufgabe verstanden bei gegebenem $g,h \in G$ ein $x \in \mathbb{Z}_q$ zu berechnen, so dass gilt $g^x = h$. In der additiven Gruppe ist das Problem trivial, da hier 1 der Generator ist.

DEFINITION 1.1.3. Sei $\mathbb{Z}_N^* \subset \mathbb{Z}_N$ die multiplikative Gruppe der invertierbaren Zahlen modulo $N$:

$$\mathbb{Z}_N^* = \{a \in [0,N[ : \operatorname{ggT}(a,N) = 1\}.$$

LEMMA 1.1.4. *Sei $G = \langle g \rangle$, $|G| = q$, $n = \lceil \ln q \rceil$, $2^{n-1} < q \leq 2^n$. Die Berechnung $a \mapsto g^a$ geht in $2n-3$ Multiplikationen in $G$.*

BEWEIS. (Binäre Methode) Sei $a = \sum_{i=0}^{n-1} a_i 2^i \in [0,q[$, $a_i \in \{0,1\}$ die Binärdarstellung von $a$. Das Produkt

$$g^a = \prod_{a_i=1} g^{2^i}$$

hat $\leq n-2$ Faktoren für $1 \leq i \leq n-1$. Man erhält $g, g^2, g^{2^2}, \ldots, g^{2^{n-1}}$ durch $n-1$ Quadrierungen. □

PROPOSITION 1.1.5. *Sei $G = \langle g \rangle$, $|G| = q$. Dann geht $h \mapsto \log_g h$ in $2\sqrt{q}$ Multiplikationen in $G$.*

BEWEIS. (Baby step-Giant step Algorithmus von Shanks) Sei

$$\begin{aligned} k := \lceil \sqrt{q} \rceil, \qquad & l := \left\lceil \tfrac{q}{k} \right\rceil, \qquad && kl - k < q \leq kl \\ L_1 := \quad & \left\{ g^i \mid 0 \leq i < k \right\} && \text{in } k-2 \text{ Mult.} \\ L_2 := \quad & \left\{ hg^{kj} \mid 0 \leq j < l \right\} && \text{in } l \text{ Mult.} \end{aligned}$$

Gilt $g^{ik} = h \cdot g^j$ für Elemente aus $L_1$ und $L_2$, so ist

$$\begin{aligned} g^{ik} \quad & \equiv \quad hg^j \mod q \\ \Leftrightarrow \quad h \quad & \equiv \quad g^{ik-j} \mod q \\ \Leftrightarrow \quad \log_g h \quad & \equiv \quad ik - j \mod q. \end{aligned}$$

. □

**Algorithm 1** Algorithmus von Shanks

**Eingabe:** $g \in G$ mit $\langle g \rangle = G$ und $h \in G$.
**Ausgabe:** $\log_g h$.

(1) Berechne $k := \left\lceil \sqrt{|G|} \right\rceil$.
(2) Setze $l := \left\lceil \frac{q}{k} \right\rceil$.
(3) **for** $i = 1, \ldots, k-1$ **do**
  (a) Berechne $g^{ik}$.
  (b) Füge $\left(i, g^{ik}\right)$ in Liste $L_1$ ein.
(4) **for** $i = 0, \ldots, l-1$ **do**
  (a) Berechne $hg^i$.
  (b) Füge $\left(i, hg^i\right)$ in Liste $L_2$ ein.
(5) Suche Elemente $\left(i, g^{ik}\right) \in L_1$ und $\left(j, hg^j\right) \in L_2$ mit identischer zweiter Koordinate.
(6) **return(**$\log_g h = ik - j$**)** $0 \leq i, j \leq k-1$**.**

**Laufzeit.** Das Sortieren der Zahlen in Schritt 5 wird durch vorheriges Sortieren erleichtert. Das Sortieren benötigt $k \lg k$ Vergleiche. Nun füge Elemente aus $L_1$ durch binary insertion in $L_2$ ein, solange bis ein Element aus $L_1 \cap L_2$ gefunden wird. Der Aufwand dafür beträgt $\leq l \lg k$ Vergleiche.
Damit ergeben sich Gesamtkosten von $\leq 2\sqrt{q} \lg \sqrt{q}$ Vergleichen.

DEFINITION 1.1.6. Ein DL-Algorithmus ist **generisch**, wenn er auf Gruppenelementen in $G$ nur Gruppenoperationen wie $\cdot$, $/$ usw. und Gleichheitstests ausführt.

Generische Algorithmen können auf die Bits der Binärkodierung der Gruppenelemente nicht zugreifen.

PROPOSITION 1.1.7. *Sei AL ein beliebiger generischer Algorithmus mit t Gruppenoperationen und $|G| = q$ prim. Dann gilt für ein zufälliges $h \in_R G$*

$$Ws_h\left[AL(h) = \log_g h\right] \leq \frac{1}{q} + \binom{t}{2}/q.$$

BEWEIS. Siehe Beweis zu Satz 4.3.1 auf Seite 54. □

Beispiele für **nicht-generische DL-Algorithmen** sind die Index-Calculus- und die Siebmethode für $G = \mathbb{Z}_p^*$: Man konstruiert über einer Primzahlbasis $p_1, \ldots, p_t$ Relationen der Form

$$h^{e_0} \cdot \prod_i p_i^{e_i} = \prod_i p_i^{e_i'} \mod p$$

und löst die zugehörigen Gleichungen

$$e_0 \log_g h + \sum_i e_i \log_g p_i = \sum_i e_i' \log_g p_i$$

auf nach $\log_g h, \log_g p_1, \ldots, \log_g p_t$.

## 1.2. Laufzeit von DL-Algorithmen zu $G \subset \mathbb{Z}_p^*$

**Quadratisches Sieb:** $e^{(1+o(1))(\ln p \ln\ln p)^{1/2}}$
**Zahlkörpersieb:** $e^{(1+o(1))(\ln p)^{1/3}(\ln\ln p)^{2/3}}$

Für generische Algorithmen ergibt sich folgende Komplexitätsschranke:

PROPOSITION 1.2.1. *Jeder generische DL-Alg. zu G mit $|G| = q$ prim benötigt $\sqrt{2q}$ Gruppenoperationen.*

BEWEIS. Terminiert der Algorithmus mit Wahrscheinlichkeit 1, dann gilt nach Satz 1.1.7 auf der vorherigen Seite $\binom{t}{2}/q \geq 1$ und somit $t^2 \geq 2q$ und $t \geq \sqrt{2q}$. □

REMARK 1.2.2. Für kryptographische Anwendungen wählt man Gruppen, für die der schnellste bekannte DL-Algorithmus generisch ist.

**Kryptographisch starke Gruppen** sind z.B. zyklische Gruppen $G$ mit $|G| = q$ prim. Z.B. $p$ prim und $q \mid p-1$:

$$G = \mathbb{Z}_p^{*\,\frac{p-1}{q}} \subset \mathbb{Z}_p^*,$$

für zufällige Primzahlen $q$ und zufällige $\frac{p-1}{2q}$ der Größe $\lg q \geq 160$, $\lg p \geq 768$.

Ebenso zufällige elliptische Kurven $E(\mathbb{F}_p)$, $E\left(\mathbb{F}_{2^t}\right)$ mit $\lg p \geq 160$, $t \geq 160$ und $p$ prim.

### 1.2.1. Der DL nach Pohlig-Hellman.

PROPOSITION 1.2.3. *Sei $|G| = q$, $q = \prod_i p_i^{e_i}$ Primfaktorzerlegung. Dann geht $h \mapsto \log_g h$ mit $\sum_{i=1}^t e_i \left(4\lg p_i + 2\sqrt{p_i}\right)$ Mult. in G durch Anwendung des Pohlig-Hellman Algorithmus.*

BEWEIS. Algorithmus 2 von Pohlig und Hellman kann auch bei jeder zyklischen Gruppe $G$ angewandt werden. Dazu benötigt er allerdings eine vollständige Primfaktorzerlegung von $|G|$. Die Idee ist, den $\log_g a$ modulo der Primzahlpotenzen von $|G|$ zu berechnen, um dann den diskreten Logarithmus mit Hilfe des CRT bestimmen zu können. □

**Korrektheit:** $G^{q/p_i}$ ist eine Gruppe der Ordnung $p_i$. $h \mapsto h^{q/p_i}$ ist Gruppenhomo. $G \to G^{q/p_i}$. Es gilt:

$$\log_g h \equiv \log_{g^{q/p_i}} h^{q/p_i} \mod p_i,$$

denn $h = q^a \Rightarrow h^{q/p_i} = g^{aq/p_i}$. Um $\log_g h \mod p_i$ zu berechnen, bildet man $h^{q/p_i}$ und berechnet in $G^{q/p_i}$ das Element $g^{q/p_i}$ und berechnet in $G^{q/p_i}$ $\log_{g^{q/p_i}} h^{q/p_i}$ mod $p_i$ nach der Baby step-Giant step Methode.

**Laufzeit:** Für die Binäre Methode werden $4\lg q$ und für die Baby-Giant-Methode $2\sqrt{p_i}$ Multiplikationen in $G$ benötigt. Das macht

$$4\lg q + 2\sqrt{p_i} \text{ Multiplikationen in } G.$$

Die CRT Zusammensetzung

$$\left(a \mod p_1^{e_1}, \ldots, a \mod p_t^{e_t}\right) \mapsto a \mod q$$

erfordert keine Multiplikationen in $G$.

### 1.2.2. Forderungen an die Berechenbarkeit.

**Algorithm 2** Pohlig-Hellman

**Eingabe:** $g \in G$ mit $\langle g \rangle = G$, $|G| = \prod_{i=1}^{k} p_i^{e_i}$ mit $p_i$ prim und $a \in G$.
**Ausgabe:** $\log_g a$.

(1) **for** $i = 1, \ldots, k$ **do**
  (a) **for** $l = 1, \ldots, p_i - 1$ **do**
    (i) Berechne $r_{il} = g^{l|G|/p_i}$.
(2) **for** $i = 1, \ldots, k$ **do**
  (a) Setze $b := a$.
  (b) **for** $j = 0, \ldots, e_i - 1$ **do**
    (i) Berechne $c = b^{|G|/p_i^{j+1}}$.
    (ii) Bestimme $i, l$ mit $r_{il} := c$.
    (iii) Setze $c_{ij} := l$.
    (iv) Setze $b := bg^{-c_{ij} p_i^j}$.
(3) **for** $i = 1, \ldots, k$ **do**
  (a) Setze $d_i := \sum_{j=0}^{e_i - 1} c_{ij} p_i^j$.
(4) Bestimme mit CRT eine Lösung $x$ des Systems $x \equiv d_i \mod p_i^{e_i}$ mit $i = 1, \ldots, k$ ($\rightarrow$Gauss).
(5) **return($x$)**

**Kryptographische Forderungen an** $\log_g, \exp_g$**.**

(1) $\exp_g$ muss in einem Bruchteil einer Sekunde gehen, z.B. in $2^{20} \approx 10^6$ Maschinen-Zykeln.
(2) $\log_g h$ muss für fast alle $h$ etwa $2^{80}$ arithmetische Schritte erfordern. Etwa $2^{50}$ Schritte sind technisch durchführbar.

DEFINITION 1.2.4. $f$ ist eine **Einweg-Funktion**, wenn $f$ „schnell" berechenbar aber „schwer" invertierbar ist.

Wir betrachten probabilistische Algorithmen $AL$ für $f^{-1}$ mit Laufzeit $|AL|$ und interne Zufallsbits $w$. Sei $h \in_R G$ zufällige Eingabe zu $AL$.

DEFINITION 1.2.5. $f^{-1}$ ist $(2^s, 2^t)$**-schwer**, wenn $AL$ mit $\mathbb{E}_{k,w}|AL| \leq 2^s$ :

$$Ws_{h,w}[f(AL(x)) = x] \leq 2^{-t}.$$

Damit ergeben sich folgende Bezeichnungen:

$$\begin{array}{lcl} (2^{80}, 2^0)\text{-schwer} & \sim & \text{worst-case schwer} \\ (2^{80}, 2^1)\text{-schwer} & \sim & \text{im Mittel schwer} \\ (2^{80}, 2^{30})\text{-schwer} & \sim & \text{fast überall schwer} \end{array}$$

**Algorithm 3** Diffie-Hellman Verfahren

**Anwendung:** Sichere Schlüsselerzeugung über einen unsicheren Kanal.
**Öffentlich:** $p$, $\langle g\rangle = \mathbb{Z}_p^*$
**Geheim:** $g^{r_A r_B}$
**Verschlüsselung:** Symmetrisch mit dem geheimen Schlüssel.
**Entschlüsselung:** Symmetrisch mit dem geheimen Schlüssel.

(1) $A$ und $B$ wählen eine Primzahl $p$ und ein erzeugendes Element $g$ der Gruppe $\mathbb{Z}_p^*$. **[öffentlich]**
(2) $A$ wählt $r_A \in_R [1, p-2]$ **[geheim]** und sendet $g^{r_A}$ an $B$ **[öffentlich]**
(3) $B$ wählt $r_B \in_R [1, p-2]$ **[geheim]** und sendet $g^{r_B}$ an $A$ **[öffentlich]**
(4) Der geheime Schlüssel ist nun $g^{r_A r_B}$.

**1.2.3. Diffie-Hellman Schlüsselerzeugung.** Analog ist das **DH-Problem** zu $G = \langle g\rangle$ bei gegebenem $g$, $g^x$, $g^y$ die Berechnung von $g^{xy}$. Diese Schwierigkeit liegt der Diffie-Hellman Schlüsselvereinbahrung (siehe Algorithmue 3) zugrunde.

LEMMA 1.2.6. *Das DH-Problem ist polynomiell äquivalent zum DL-Problem:*

$$\Leftrightarrow \quad DH \leq_{pol} DL.$$

BEWEIS. Jeder DL-Algorithmus löst das DH-Problem:

$$x := \log_g(g^x), y := \log_g(g^y) \quad \Rightarrow \quad g^{xy} := (g^y)^x.$$

□

PROPOSITION 1.2.7. *Sei $G = \langle g\rangle$, $|G| = q$ prim. Dann benötigt jeder generische Algorithmus $(g, g^a, g^b) \mapsto g^{ab}$ mindestens $\sqrt{2q}$ Gruppenoperationen.*

BEWEIS. Siehe Beweis zu Satz 1.2.1 auf Seite 10. □

PROBLEM 1.2.8. $\mathrm{DL} \leq_{\mathrm{pol}}$ DH-Problem.

Kann man also $\log_g$ mit polynomiell vielen DH-Aufrufen berechnen?

DH: $(g, g^a, g^b) \mapsto g^{ab}$ entspricht Multiplikationen in $\mathbb{Z}_q^*$: $(a,b) \mapsto ab$.

Ist eine Zerlegung von $q-1$ bekannt, dann gilt $\mathrm{DL} \leq_{\mathrm{pol.}}$ DH:

PROPOSITION 1.2.9. *Sei $G = \langle g\rangle$, $|G| = q$ prim, mit gegebener Zerlegung $q-1 = q_1 \cdots q_t$ in teilerfremde $q_1, \ldots, q_t$. Dann geht $h \mapsto \log_g(h)$ mit $4t \lg q + 2\sum_{i=1}^{t} \sqrt{q}$ DH-Aufrufen und $2t \lg q$ Multiplikationen in $G$.*

BEWEIS. Sei $x = \log_g(h)$, $x \neq 0$, $x \in \mathbb{Z}_q^*$. Bestimme nun $x$, durch Finden eines Generators $c$ mit $\langle c\rangle = \mathbb{Z}_q^*$ und bestimme die $w_i$ mit $x = c^{w_i} \mod q_i$ für alle $1 \leq i \leq t$. Bilde zuletzt mit dem CRT aus den gefundenen $w_i = w \mod q_i$ ein $w \in \mathbb{Z}_{q-1}$ mit $x := c^w \mod q$:

| | | | | |
|---|---|---|---|---|
| 1. | $h = g^x$ | $\mapsto$ | $g^{\left(x^{2^i}\right)}$, $i = 0, \ldots, \lg q$ | $\lg q$ DH |
| 2. | | $\mapsto$ | $g^{\bar{x}} = g^{\left(x^{(q-1)/q_j}\right)}$ | $\lg q$ DH |
| 3. | $g$ | $\mapsto$ | $g^{\bar{c}}$ | $2\lg q$ Mult. in $G$ |
| 4. | $g^c$ | $\mapsto$ | $g^{\bar{c}}$ | $2\lg q$ DH |
| 5. | Löse $\vec{c} = \bar{x}$ | für | $\bar{e}, \bar{x} \in \mathbb{Z}_q^{*\frac{q-1}{q_j}}$ | $2\sqrt{q_j}$ DH |

Eine Alternative wäre folgende Vorgehensweise:

| | | | | |
|---|---|---|---|---|
| 1. | $x$ | $\mapsto$ | $g^{\bar{x}}$ | $2\lg q$ DH |
| 2. | $g^{(\vec{c}^t)}$ | für | $t = 0, \ldots, q_j - 1$ | $2q_i \lg q$ Mult. in $G$ |
| 3. | Löse | | $g^{\bar{x}} = g^{(\vec{c}^t)}$ durch Vergleiche. | |

□

REMARK 1.2.10. Bei Algorithmus 3 handelt es sich nicht nicht um einen Schlüsselaustausch, sondern um eine Schlüsselvereinbahrung, da beide Parteien an der Erzeugung des Schlüssels beteiligt sind.

**Algorithm 4** ElGamal-Verfahren

**Anwendung:** Asymmetrische Verschlüsselung.
**Öffentlich:** $p$, $\langle g \rangle = \mathbb{Z}_p^*$, $h = g^e$
**Geheim:** $e$
**Verschlüsselung:** $e_k(x,r) = (g^r \mod p, xh^r \mod p) = (y_1, y_2)$
**Entschlüsselung:** $d_k(y_1, y_2) \equiv y_2 (y_1^e)^{-1} \mod p$

Ein Schlüssel wird durch Wahl einer Primzahl $p$ und eines erzeugenden Elements $g$ für $\mathbb{Z}_p^*$ erzeugt. Dazu wird aus einem $e \in \mathbb{N}$ die Zahl $h \equiv g^e \mod p$ berechnet. Die Zahlen $p, g, h$ sind öffentlich, der Exponent $e$ bleibt geheim. Es gilt

$$P_k = \mathbb{Z}_p^*, C_k = \mathbb{Z}_p^* \times \mathbb{Z}_p^*.$$

Zum Verschlüsseln wird ein $r \in_R \mathbb{Z}_{p-1}$ gewählt und

$$e_k(x,r) = (g^r \mod p, xh^r \mod p) = (y_1, y_2)$$

berechnet.
Die Entschlüsselungsfunktion ist dann

$$d_k(y_1, y_2) \equiv y_2 (y_1^e)^{-1} \mod p.$$

**1.2.4. ElGamal Verschlüsselung.** Durch die Wahl von $r \in_R \mathbb{Z}_{p-1}$ ist die Verschlüsselung randomisiert. Zu jedem $x \in G$ gibt es $q$ Cipher-Texte $(g^r, xh^r)$ – die Expansionsrate ist 2.

Bzgl. des **Nachrichtenraums** $M$ bei der ElGamal-Verschlüsselung gilt

(1) $M \subset G$ geht für $G = \mathbb{Z}_p^*$, $E_{A,B}(\mathbb{F}_p)$, $E_{A,B}(\mathbb{F}_{2^t})$
(2) $(M, +)$ ist Gruppe mit öffentlicher Zufallsfunktion $H : G \to M$.

Das Verschlüsseln erfolgt durch

$$x \mapsto (g^r, m + H(h^r))$$

das Entschlüsseln durch

$$m := m + H(h^r) - H(g^{rx}).$$

## 1.3. Angriffe auf Verschlüsselungen

Wir unterscheiden im Folgenden drei Arten von Angriffen:

**Ciphertext Only Attack (CA):** Der Angreifer hat den Ciphertext und den öffentlichen Schlüssel und will entschlüsseln.

**Chosen Plaintext Attack (CPA):** Der Angreifer erhält zusätzlich Ciphertexte zu Nachrichten seiner Wahl.

**Chosen Ciphertext Attack (CCA):** Der Angreifer hat ein Entschlüsselungsorakel zum Entschlüsseln von Ciphertexten, die verschieden sind von der Herausforderung.

PROPOSITION 1.3.1. *Das Brechen der ElGamal-Verschlüsselung durch CA ist äquivalent zum Lösen des DH-Problems.*

LEMMA 1.3.2. *Für dieElGamal-Verschlüsselung ist CPA äquivalent zu CA.*

BEWEIS. Der Angreifer kann beliebige Nachrichten selbst verschlüsseln. □

Bzgl. der CCA stellt sich die ElGamal-Verschlüsselung als unsicher dar:

LEMMA 1.3.3. *Die ElGamal-Verschlüsselung wird durch CCA gebrochen.*

BEWEIS. Gegeben ist $g, h, cip = (g^r, mh^r)$. Der Angreifer sucht sich $a \in_R \mathbb{Z}_q$ und definiert $cip' := (g^r, mh^r h^a)$, indem er einfach die zweite Komponente mit $h^a$ multipliziert und lässt $cip'$ entschlüsseln. Damit ist $cip'$ der Schlüsseltext zu $mh^a$. Durch einfaches Dividieren von $h^a$ erhält der Angreifer somit den Klartext:

$$\begin{aligned} & cip &=& (g^r, mh^r) \\ & cip' &=& (g^r, mh^r h^a) = (g^r, mh^{r+a}) \\ \Rightarrow\quad & d_k(cip') &=& mh^a \\ \Rightarrow\quad & cip &=& d_k(cip')/h^a = m. \end{aligned}$$

□

REMARK 1.3.4. Um sich bei der ElGamal-Verschlüsselung gegen CCA zu schützen kann man dem Ciphertext einen Nachweis über die Kenntnis von $r$ (z.B. durch eine digitale Unterschrift zum Schlüsselpaar $(r, g^r)$) anhängen.

## 1.4. Elliptische Kurven

Elliptische Kurven kann man über beliebigen Körpern definieren. Für die Kryptographie interessant sind elliptische Kurven über endlichen Körpern, speziell über Primkörpern. Die Verwendung elliptische Kurven für kryptographische Anwendungen kann mehrere Gründe haben:

(1) Public-Key-Kryptographie mit elliptischen Kurven ist die wichtigste bis jetzt bekannte Alternative zu RSA-basierten Verfahren. Solche Alternativen sind dringend nötig, da niemand die Sicherheit von RSA garantieren kann.

(2) Das EC-Kryptosystem bietet Effizienzvorteile gegenüber dem RSA-Verfahren: Während RSA modulare Arithmetik mit 1024-Bit-Zahlen verwendet, genügen für EC-Verfahren 163-Bit-Zahlen. Zwar ist die Arithmetik auf elitischen Kurven aufwendiger als die in primen Restklassengruppen, doch das wird durch die geringere Länge der verwendeten Zahlen kompensiert. Dadurch ist es z.B. möglich, EC-Kryptographie auf Smartcards ohne Co-Prozessor zu implementieren, was billiger als eine Lösung mit Co-Prozessor ist. [**Beutelspacher et al. 2001**, Kapitel 12.2]

Sei $\mathbb{F} \subset \mathbb{K}$ Körper und $A, B \in \mathbb{F}$. Die Gleichung

$$y^2 z = x^3 + Axz^2 + Bz^3 \quad (*)$$

hat die **Diskriminante** $\Delta = -16\left(4A^3 + 27B^2\right)$.

DEFINITION 1.4.1. Zwei Lösungen $(x,y,z)$ und $(x',y',z')$ aus $\mathbb{K}^3$ heissen **äquivalent**, wenn gilt

$$\exists c \in \mathbb{K}^* : c\,(x,y,z) = \left(x',y',z'\right).$$

Wir betrachten nur Lösungen $(x,y,z) \neq (0,0,0)$. Dabei ist $(x:y:z)$ eine **Äquivalenzklasse** von $(x,y,z)$.

DEFINITION 1.4.2. Die **elliptische Kurve** $E_{A,B}(\mathbb{K})$ besteht aus allen Punkten $P = (x:y:z)$ mit $x,y,z \in \mathbb{K}$, die die Gleichung $(*)$ oben lösen.

Dabei gilt:

(1) Es gibt genau ein $(x:y:z) \in E_{A,B}(\mathbb{K})$ mit $z = 0$, nämlich $(0:1:0)$. ($z = 0 \Rightarrow x = 0$)
(2) Jeder Punkt $(x:y:z)$ mit $z \neq 0$ ist von der Form $(x',y',1)$ mit $x' = x/z$ und $y' = y/z$. Dabei sind $x',y'$ **normierte Koordinaten** und $x,y,z$ **homogene Koordinaten**.
(3) Die Punkte $(x:y:1) \in E_{A,B}(\mathbb{K})$ lösen die Gleichung $y^2 = x^3 + Ax + B$.

Die Gruppenstruktur einer elliptischen Kurve $E_{A,B}(\mathbb{K})$ ist wie folgt definiert:

(1) Das neutrale Element ist $(0:1:0) =: O$ mit $P + O = O + P = P$.
(2) $(x:y:z) + (x:-y:z) = O$.
(3) Seien $P_i = (x_i : y_i : 1)$ für $i = 1,2$ mit $y_1 \neq y_2$, dann gilt

$$P_1 + P_2 = \left(x_3 : y_3 : 1\right)$$

mit $x_3 = \lambda^2 - x_1 - x_2$ und $y_3 = \lambda\left(x_1 - x_3\right) - y_1$. Also ist

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{, falls } P_1 \neq P_2 \\ \frac{3x_1^2 + A}{2y_1} & \text{, falls } P_1 = P_2 \end{cases}.$$

PROPOSITION 1.4.3. *$E_{A,B}(\mathbb{K})$ mit $+$ ist eine abelsche Gruppe.*

COROLLARY 1.4.4. *Die Addition in $E_{A,B}(\mathbb{K})$ geht mit $O(1)$ Additionen/Multiplikationen im Koeffizientenkörper $\mathbb{F} \subset \mathbb{K}$. Die Berechnung von $P_1 + P_2$ in homogenen Koordinaten geht ohne Division.*

## 1.5. Schlüsseltransport mit Rücklauf

(Shamir's „No Key Protocol")

Zwei Teilnehmer möchten sich eine vertrauliche Nachricht zukommen lassen, besitzen jedoch keinen gemeinsamen Schlüssel. Shamir schlägt vor, dass $A$ die Nachricht mit seinem Schlüssel verschlüsselt und an $B$ schickt. Dieser verschlüsselt die Nachricht wieder mit seinem Schlüssel und schickt sie an $A$ zurück. $A$ entschlüsselt die Nachricht mit seinem Schlüssel und schickt sie wieder an $B$. Dieser kann die Nachricht jetzt mit seinem Schlüssel entschlüsseln und lesen.

Eine mathematische Realisierung basiert auf der diskreten Exponentialfunktion, die die wesentliche Eigenschaft

$$f_a\left(f_b(x)\right) = f_b\left(f_a(x)\right)$$

besitzt. Beide Teilnehmer einigen sich auf eine große Primzahl $p$. $A$ erzeugt ein Paar $(a,a')$ mit $a \cdot a' \equiv 1 \mod (p-1)$ und $B$ erzeugt genauso $(b,b')$. Soll eine Nachricht $m$ an $B$ übertragen werden, so wird wie folgt vorgegangen:

| | $A$ | | $B$ |
|---|---|---|---|
| 1. | Berechnet $x \equiv m^a \mod p$ | $x \rightarrow$ | |
| 2. | | $\leftarrow y$ | Berechnet $y \equiv x^b \mod p$ |
| 3. | Berechnet $z \equiv y^{a'} \mod p$ | $z \rightarrow$ | Berechnet $z^{b'} \equiv m^{aba'b'} \equiv m \mod p$ |

**Vorteil.** Kein öffentliches Verzeichnis erforderlich, Authentikation gesichert.

**Nachteil.** Eignet sich weniger zum Versenden verschlüsselter Nachrichten, weil die Nachricht in $g$ kodiert werden muss.

**Angriff.** $C$ schickt in Schritt 2. $m^{ac}$ und erhält von $A$ $m^c$ zurück, womit die Nachricht entschlüsselt werden kann.

Damit Authentikation (Ursprungsnachweis) gesichert ist, muss der Übertragungskanal gegen aktive Störer geschützt weden, z.B. durch eine digitale Unterschrift. In diesem Fall wird in Schritt 1. und 3. zusätzlich noch $\mathrm{sig}_A(x)$ bzw. $\mathrm{sig}_A(z)$ und in Schritt 2. $\mathrm{sig}_B(y)$ mit übertragen.

Wieder ist das Verfahren so sicher wie das DL-Problem: Gibt es eine Lösung für das DL-Problem, so kann ein Angreifer aus $x = m^a$ und $y = m^{ab} = \left(m^b\right)^a$ den Schlüssel $a$ berechnen und dann ganz einfach $a'$ erhalten.

## 1.6. ElGamal Signaturen

**Öffentlich:** $g, G = \langle g \rangle, |G| = q, H()$.
**Secret-Key:** $x \in_R \mathbb{Z}_q$.
**Public-Key:** $h = g^x$.
**Signatur:** $(r,s) \in G \times \mathbb{Z}_q$ mit $r \equiv g^k$ und $s \equiv k^{-1}\left(H(m) - x \cdot r\right)$ mit $k \in_R \mathbb{Z}_{q-1}^*$ (geheim).
**Test:** $g^{H(m)} \stackrel{?}{\equiv} h^{H(r)} \cdot r^s$.

Der Absender sucht sich eine kollisionsresistente Hashfunktion

$$H : \{0,1\}^* \to \{1,2,\ldots,q-1\}$$

und errechnet den Hashwert $H(m)$ der Nachricht $m \in \{0,1\}^*$ und den öffentlichen Schlüssel

$$h \equiv g^x \mod q.$$

Jetzt wählt er eine Zufallszahl $k \in_R \mathbb{Z}_q^*$ und berechnet

$$r :\equiv g^k \mod q, \quad s :\equiv k^{-1}(H(m) - xH(r)) \mod (q-1).$$

Die digitale Unterschrift besteht nun aus dem Paar $(r,s) \in G \times \mathbb{Z}_q$.

Das Prüfen der Unterschrift geschieht durch ein Vergleichen von $g^{H(m)}$ mit $h^{H(r)} \cdot r^s \mod q$. Bei Gleichheit ist die Nachricht korrekt, da nach Fermat gilt:

$$h^{H(r)} \cdot r^s \equiv (g^x)^{H(r)} \cdot g^{ks} \equiv g^{xH(r)+ks} \equiv g^{H(m)} \mod q,$$

da $ks \equiv H(m) - xH(r) \mod (q-1)$ ist.

**Sicherheit.** Um Unterschriften fälschen zu können muss die Gleichung $g^{H(m)} = h^{H(r)} r^s$ oder der $\log_g h$ gelöst werden.
Es muss allerdings darauf geachtet werden, dass keine zwei Nachrichten mit der gleichen Zufallszahl $k$ signiert werden. In diesem Fall kann ein Angreifer den geheimen Schlüssel finden:

$$\begin{aligned}
r &:= g^k \mod q \\
h^{H(r)} \cdot r^{s_1} &\equiv g^{H(m_1)} \mod q \\
h^{H(r)} \cdot r^{s_2} &\equiv g^{H(m_2)} \mod q \\
\Rightarrow \quad g^{H(m_1)-H(m_2)} &\equiv r^{s_1-s_2} \mod q \\
\Leftrightarrow \quad g^{H(m_1)-H(m_2)} &\equiv g^{k(s_1-s_2)} \mod q \\
\Rightarrow \quad H(m_1)-H(m_2) &\equiv k(s_1-s_2) \mod (q-1) \\
\text{Sei } d &:= \mathrm{ggT}(s_1-s_2, q-1) \\
\Rightarrow \quad H(m') &= \frac{H(m_1)-H(m_2)}{d} \\
s' &:= \frac{s_1-s_2}{d} \\
q' &:= \frac{q-1}{d} \\
\Rightarrow \quad H(m') &= k \cdot \frac{s_1-s_2}{d} \mod q' \\
\overset{\mathrm{ggT}(s',q')=1}{\Rightarrow} \quad \varepsilon &\equiv (s')^{-1} \mod q' \\
x &\equiv H(m')\varepsilon \mod q' \\
\Rightarrow \quad x_i &\equiv H(m')\varepsilon + iq' \mod q \quad i = 0,\ldots,d-1
\end{aligned}$$

Für eines der $i$ gilt dann $h \equiv g^{x_i} \mod q$.

**Nachteil.** Fälschen ist nicht äquivalent zum DL-Problem. Siehe dazu auch [**PointchevalStern1996**].

**Länge der Unterschrift.** $2\lg q$.

DEFINITION 1.6.1. Unter einer **existential forgery**, oder auch **existentielle Fälschung**, werden nach [**Menezes1997**, Kapitel 9] Attacken verstanden, bei denen es für einen Angreifer möglich ist, Signaturen von Nachrichten zu erzeugen ohne den geheimen Schlüssel zu kennen, wobei er allerdings keine Kontrolle über den zu signierenden Text hat.

Eine existential forgery wird ermöglicht, wenn anstelle des Hashwertes der Nachricht die Nachricht selbst signiert wird.

Sei also $G = \mathbb{Z}_p^*$ und $H : G \subset [0,p[$ die Hashfunktion mit $H(m) = m$ und $H(r) = r$. Wähle nun $u \in_R \mathbb{Z}$ und $v \in \mathbb{Z}$ mit $\mathrm{ggT}(v, p-1) = 1$ und setze

$$r := g^u h^v \equiv g^{u+xv} \mod p, \quad s := -rv^{-1} \mod (p-1).$$

Dann ist $(r,s)$ Signatur zu $m := su \mod (p-1)$:

$$\begin{aligned} h^{H(r)} r^s &\equiv (g^x)^{H(r)} \left(g^{u+xv}\right)^s \equiv g^{xr} g^{su+svx} \\ &\equiv g^{xr+svx} g^{su} \equiv g^{su} \equiv g^m \mod p, \end{aligned}$$

da gilt

$$\begin{aligned} xr + svx &\equiv xr + \left(-rv^{-1}\right) vx \\ &\equiv xr - rx \equiv 0 \mod (p-1). \end{aligned}$$

## 1.7. Schnorr Signaturen

**Öffentlich:** $g, G = \langle g \rangle, |G| = q \text{prim}, H : G \times \{0,1\}^* \to [0, 2^t[ \subset \mathbb{Z}_p, t \geq 80$.
**Secret-Key:** $x \in_R \mathbb{Z}_q$.
**Public-Key:** $h = g^x$.
**Signatur:** $(e,y) \in [0,2^t[ \times \mathbb{Z}_q$ mit $e \equiv H(g^r, m)$ und $y \equiv r + ex$ mit $r \in_R \mathbb{Z}_{q-1}^*$
**Test:** $H(g^y h^{-e}, m) \stackrel{?}{\equiv} e$.

Um eine Signatur von $m$ zu erzeugen wählt $A$ ein $r \in_R \mathbb{Z}_q$ und berechnet

$$e := H(g^r, m), \quad y := r + xe \mod q.$$

Die Signatur ist dann $(e,y) \in [0,2^t[ \times \mathbb{Z}_q$. Zur Überprüfung wird folgender Vergleich angestellt:

$$H\left(g^y h^{-e}, m\right) \stackrel{?}{=} e.$$

Es gilt

$$g^y h^{-e} = g^{y-ex} = g^r \quad \Leftrightarrow \quad r = y - ex.$$

Die Länge der Signatur ist dann $\lg q + t \leq 160 + 80 = 240$. Ursprünglich war $G \subset \mathbb{Z}_p^*$ mit $p$ prim und $G$ elliptische Kurve usw.

Wählt man $p$ als Primteiler von $q-1$, so kann auch kurze Unterschriften erzeugen, dieu z.B. zum Unterschreiben der öffentlichen Schlüssel eingesetzt werden.

**Sicherheit:** Die Schnorr-Signaturen sind sicher im Random Oracle und im generischen Modell.

## 1.8. Digital Signature Algorithm, DSA

**Öffentlich:** $g, G = \langle g \rangle, |G| = q \text{prim}, \text{G} \subset \mathbb{Z}_p^*$ oder $G = E(\mathbb{F}_p), H : \{0,1\}^* \to \mathbb{Z}_q$.
**Secret-Key:** $x \in_R \mathbb{Z}_q$.
**Public-Key:** $h = g^x$.
**Signatur:** $(r,s) \in G \times \mathbb{Z}_q$ mit $r := (g^k \mod p) \mod q$ und $s := k^{-1}(H(m) + xr) \mod q$.
**Test:** $0 < r, s < q$ und $r = (g^{H(m)/s} h^{r/s} \mod p) \mod q$.

Eine Signatur zu einer Nachricht $m$ wird durch Wahl von $k \in_R \mathbb{Z}_q^*$ und

$$r := \left(g^k \mod p\right) \mod q, \quad s := k^{-1}(H(m) + xr) \mod q$$

erzeugt. Die Signatur ist dann $(r,s) \in G \times \mathbb{Z}_q$.

Zur Überprüfung der Korrektheit wird getestet, ob gilt

$$\begin{aligned} & r \equiv (g^{H(m)/s} h^{r/s} \mod p) \mod q \\ \Leftrightarrow \quad & g^k \equiv (g^{H(m)/s} g^{xr/s} \mod p) \mod q \\ \Leftrightarrow \quad & k \equiv H(m)/s + xr/s \mod (q-1) \\ \Leftrightarrow \quad & k \equiv (H(m) + xr)k/(H(m) + xr) \mod (q-1) \\ \Leftrightarrow \quad & k \equiv k \mod (q-1). \end{aligned}$$

Im Fall $G = E(\mathbb{F}_p)$ ersetze

$$\begin{aligned} g \quad & \text{durch} \quad P \in E(\mathbb{F}_p) \text{ mit } \langle P \rangle = E(\mathbb{F}_p) \\ r \quad & \text{durch} \quad x \text{ mit } (x : y : 1) = k \cdot P. \end{aligned}$$

Siehe auch [**CoppersmithOdlukoSchroeppel1985**].

## 1.9. Nyberg-Rueppel Signaturen, Message Recovery

[**NybergRueppel1993**]

Ziel dieser Signatur war die Minimierung der Länge von (Nachricht, Signatur).

**Secret-Key:** $x \in_R \mathbb{Z}_q$.
**Public-Key:** $h = g^x$
**Signatur:** $(e,y) \in \mathbb{Z}_q^2$ mit $e \equiv (m \cdot g^r)$ und $y \equiv r - xe$.
**Test:** $m \stackrel{?}{=} e/(g^y h^e) \mod q$.

Alle Gruppenelemente sind als ganze Zahlen kodiert, $G \subset \mathbb{Z}$, z.B. $G \subset \mathbb{Z}_p^* \cong [1, p[$. Seien nun $\cdot, /$ Mult. und Div. in $\mathbb{Z}$ – im Unterschied zur Mult. und Div. in $G$.

Eine Signatur wird erzeugt durch Wahl von $r \in_R \mathbb{Z}_q$ und

$$e := (m \cdot g^r) \mod q \qquad y := r - xe$$

Die NR-Signatur ist dann $(e,y) \in \mathbb{Z}_q^2$ mit der Bitlänge ohne $m$ von $\lg q \approx 160$.

Die Überprüfung bzw. das Recovery funktioniert durch

$$m \stackrel{?}{=} e/(g^y h^e) \mod q.$$

Das gilt wegen

$$g^y h^e = g^{y+ex} = g^{r-ex+ex} = g^r = e/m.$$

**Sicherheit.** Auf NR-Signaturen kann ein CMA-Angriff vollzogen werden:
$(e,y)$ ist eine Signatur von $m$ gdw. $(e,y+r')$ eine Signatur von $\left(m/g^{r'}\right) \mod q$ ist. Um $m$ zu signieren fordert man eine Signatur $(e,y')$ von $m/g^{r'}$ an und erhält eine Signatur $(e,y'-r')$ von $m$.

In der Standard Version nach IEEE ist $m \in \mathbb{Z}_q$. Die Signatur wird dann aus $r \in_R \mathbb{Z}_q$ erzeugt mit

$$e := (g^r + m) \mod q.$$

Wenn $e \neq 0$, dann wird $y := r - xe$ gesetzt. Die Signatur ist jetzt $(e,y) \in \mathbb{Z}_q^2$. Die Unterscheidung nach $e$ muss gemacht werden, da Signaturen $(0,y)$ von $m \equiv g^y \mod q$ ohne $x$ gehen und deshalb verboten sind.

Das Prüfen, Recovery geht mittels

$$m \stackrel{?}{=} e - g^y h^e \mod q.$$

**Schutz gegen CMA.** Um sich gegen CMA-Attacken zu schützen muss die Signatur etwas abgeändert werden:
Der Nachrichtenraum sei $\mathbb{Z}_q^k$, $k \geq 1$ mit $H : G \to \mathbb{Z}_q^k$ und $H' : \mathbb{Z}_q^k \to \mathbb{Z}_q$ öffentliche Zufallsfunktionen (es sei keine Kodierung von $m \mapsto cod(m) \in G$ erforderlich).
Die Signatur wird nun mittels $r \in_R \mathbb{Z}_q$,

$$e := \left(e_1, \ldots, e_k\right) = H\left(g^r + m\right), \quad \bar{e} := e_1 + \ldots + e_k, \quad y := r - \bar{e}x.$$

erzeugt und ist $(e,y) \in \mathbb{Z}_q^{k+1}$. Das Prüfen/Recovery geht mit

$$m \stackrel{?}{=} e - H\left(g^y h^{\bar{e}}\right).$$

Ein existential forgery im ROM geht mittels

$$(e,y) \mapsto m := e - H\left(g^y h^{\bar{e}}\right)$$

und liefert eine Signatur einer Zufallsnachricht $m \in_R \mathbb{Z}_q^k$, die stat. unabh. ist von $e,y$.

KAPITEL 2

# Diskrete Logarithmus Identifikation

Dieses Kapitel widmet sich komplett der DL-Identifikation, also diverser Verfahren, in denen ein Prover $P$ einem Verifier $V$ seine Identität beweist. Der Clou hierbei ist, dass $V$ den öffentlichen Schlüssel von $P$ besitzt (in der Regel $v = g^x$) und $P$ beweist, dass er $x$ (seinen geheimen Schlüssel) kennt. Dabei sollten folgende Bedingungen erfüllt sein:

**Completeness:** Handelt es sich bei $P$ nicht um einen Betrüger, so soll $P$ das Protokoll mit nur vernachlässigbarer Wahrscheinlichkeit nicht bestehen.

**Soundness:** Handelt es sich bei $P$ um einen Betrüger (im Folgenden $\tilde{P}$) und besteht er das Protokoll mit nicht vernachlässigbarer Wahrscheinlichkeit, so kennt er auch den geheimen Schlüssel.

Desweiteren sollte gelten:

(1) Ein Angreifer, der die Kommunikation von $P$ und $V$ belauscht darf daraus keine Informationen über das Geheimnis $x$ erhalten.
(2) Einem betrügerischen Verifier $\tilde{V}$ darf es nicht möglich sein, $P$ nach einer Identifikation zu impersonifizieren.

Zunächst wird die Schnorr-Identifikation vorgestellt und bewiesen, dass diese „sicher" gegen passive Angreifer ist – also Angreifer, die höchstens Kommunikation belauschen können, jedoch nicht aktiv mit dem ehrlichen Prover $P$ kommunizieren können. Ein Beweis der Sicherheit gegen aktive Angreifer im Turingmaschinen-Modell, die vorher mit dem ehrlichen Prover kommunizieren und Informationen auswerten können, wurde bis heute nicht gefunden, im generischen Modell und im ROM erweist sich das Verfahren aber als sicher (siehe Kapitel 4). In Abschnitt 2.5 wird die Sicherheit der Schnorr-Identifikation gegen aktive Angreifer im Turingmaschinen-Modell mit einer Abwandlung nach Okamoto bewiesen.

Desweiteren wird eine Abwehrmöglichkeit gegen Man-In-The-Middle-Attacken erklärt und ein Verfahren nach Brickell-McCurley präsentiert, das sich sowohl das DL-Problem als auch das Faktorisierungs-Problem stützt.

## 2.1. Schnorr Identifikation

**[ChaumEvertseGraaf1998, Schnorr1991]**

**Öffentlich:** $g, G = \langle g \rangle, |G| = q\text{prim}, h = g^x$.
**Privat:** $x \in_R \mathbb{Z}_q$.

Damit erfolgt eine Identifikation wie folgt:

| $(P,V)$ | $P$ | | $V$ |
|---|---|---|---|
| 1. | Wählt $r \in_R \mathbb{Z}_q$ und berechnet $\bar{g} := g^r$ | $\bar{g} \rightarrow$ | |
| 2. | | $\leftarrow e$ | wählt $e \in_R [0, 2^t[$ |
| 3. | Berechnet $y := r + xe$ | $y \rightarrow$ | Berechnet $\bar{g} \stackrel{?}{=} g^y h^{-e}$ |

Die Korrektheit ergibt sich aus

$$g^y h^{-e} = g^{r+xe} g^{-xe} = g^r = \bar{g}.$$

Im Folgenden sei der Erfolg eines Betrugs definiert durch

$$\varepsilon := \text{ERF}_w\left(\tilde{P}, h\right) := Ws_w\left[\left(\tilde{P}, V\right) \text{ akzeptiert mit } h\right]$$

und

$$w := \text{Zufallsbits von } \left(\tilde{P}, v\right).$$

LEMMA 2.1.1. *Es gibt ein prob. pol. Zeit $\tilde{P}$ mit $ERF_w\left(\tilde{P}, h\right) = 2^{-t}$.*

BEWEIS. $\tilde{P}$ wählt $r \in_R \mathbb{Z}_q$ und $\tilde{e} \in_R [0, 2^t[$ und sendet $\bar{g} := g^r h^{-\tilde{e}}$ und $y := r$. Offenbar gilt

$$\bar{g} = g^r h^{-e} \quad \Leftrightarrow \quad e = \tilde{e}$$

und

$$Ws[e = \tilde{e}] = 2^{-t}.$$

□

Sei $|\tilde{P}|$ die Schrittzahl von $(\tilde{P}, V)$, diese sei o.B.d.A. unabhängig von $h, r$. Ebenso sei $\varepsilon = \text{ERF}_w(\tilde{P}, h)$ unabh. von $h$. Ziel ist ein „Angriff aus dem Stand" (also ein passiver Angriff). Wir zeigen jetzt, dass ein Angreifer mit nicht vernachlässigbarer Erfolgsws. in der Lage ist, den DL zu berechnen:

$$\text{DL} \leq_{\text{pol}} \text{ erfolgreicher } \tilde{\text{P}}.$$

PROPOSITION 2.1.2. *Es gibt einen prob. Extraktor*

$$AL : (\tilde{P}, h) \mapsto \log_g h$$

*mit $E_w |AL| = O(|\tilde{P}|/\varepsilon)$, sofern $\varepsilon \geq 2^{-t+1}$.*

**Korrektheit:** Die Korrektheit von Algorithmus 5 lässt sich zeigen durch

$$\text{ERF}_{w_{\tilde{P}}, e} = 1 \quad \Leftrightarrow \quad \bar{g} = g^y h^{-e}$$

mit

$$\bar{g} = \bar{g}\left(\tilde{P}, w_{\tilde{P}}\right) \quad , \quad y = y\left(\tilde{P}, w_{\tilde{P}}, e\right).$$

**Algorithm 5** AL$(\tilde{P},h)$ „Probabilistischer Extraktor"

AL$(\tilde{P},h)$ mit $w_{\mathrm{AL}}$ enthält $w_{\tilde{P}} \in \{0,1\}^{|\tilde{P}|}$ für alle Aufrufe von $\tilde{P}$:

(1) $u := 0$;
*** Stufe 1: ***
(2) **do**
(a) $\bar{g} := \bar{g}(\tilde{P}, w_{\tilde{P}})$ – wie $\tilde{P}$ mit $w_{\tilde{P}} - e \in_R [0,2^t[$;
(b) $y := y(\tilde{P}, w_{\tilde{P}}, e)$ – wie $(\tilde{P}, V)$;
(c) $u := u+1$;
(3) **until** $(\mathrm{ERF}_{w_{\tilde{P}},e} = 1)$;
*** Stufe 2: ***
*** In $u$ stehen #Versuche beim Erzeugen von $w_{\tilde{P}}, \bar{g}, y, e$ ***
(4) **for** maximal $u$ zufällige $\bar{e} \in_R [0,2^t[$ **do**
(a) Teste für $w_{\tilde{P}}, \bar{g}$ ob $\mathrm{ERF}_{w_{\tilde{P}},\bar{e}} = 1$;
(5) **if** $\mathrm{ERF}_{w_{\tilde{P}},\bar{e}} = 1$, $e \neq \bar{e}$ **then**
(a) $\log_g h = \frac{y-\bar{y}}{e-\bar{e}} \mod q$.
(6) **else**
(a) **goto** 1 und starte mit unabhängigen $w_{\tilde{P}}$ neu.

Dann gilt:

$$g^y h^{-e} = \bar{g} = g^{\bar{y}} h^{-\bar{e}} \Rightarrow \log_g h = \frac{y-\bar{y}}{e-\bar{e}} \mod q.$$

Beachte, dass $\bar{g}$ nur von $w_{\tilde{P}}$ und nicht von $e, \bar{e}$ abhängt!

**Die Erfolgsmatrix.** Mit der Erfolgsmatrix soll zu jedem von $V$ zufällig gewählten $e$ bestimmt werden, für welche Zufallsbits $w \in \{0,1\}^{|\tilde{P}|}$ der Angreifer $\tilde{P}$ eine korrekte Identifikation erschleichen kann.

Es ist $\mathrm{ERF}_{w,e} \in \{0,1\}$, wobei 1 für einen Erfolg und 0 für einen Misserfolg steht (siehe Abbildung 2.1.1).

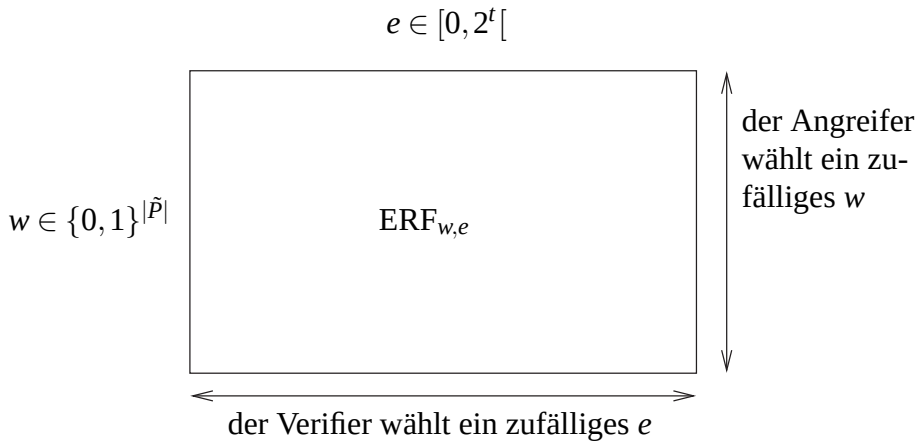



ABBILDUNG 2.1.1. Die Erfolgsmatrix

Ziel des AL ist es nun zwei Einsen in einer Zeile $w \in \{0,1\}^{|\tilde{P}|}$ zu finden. Eine Zeile mit zwei Einsen verdoppelt die Chancen, die Identifikation zu fälschen auf $\geq 2^{-t+1}$.

DEFINITION 2.1.3. Eine Zeile $w$ heisst **schwer**, wenn sie mehr als eine Eins enthält.

FACT 2.1.4. *Wegen $\varepsilon \geq 2^{-t+1}$ liegt mindestens die Hälfte aller Einsen in schweren Zeilen.*

BEWEIS. Einerseits gilt in dem Fall, dass es keine schwere Zeile gibt, also in jeder Zeile maximal eine Eins stehen kann

$$\sum_{w \text{ leicht}} \sum_{e} \mathrm{ERF}_{w,e} \leq 2^{|\tilde{P}|}$$

und andererseits folgt aus $\varepsilon \geq 2^{-t+1}$ (siehe Lemma 2.1.1 auf Seite 22), dass gilt

$$\sum_{w} \sum_{e} \mathrm{ERF}_{w,e} \geq 2^t 2^{|\tilde{P}|} 2^{-t+1} = 2 \cdot 2^{|\tilde{P}|}.$$

Hier wird das Produkt aus der Anzahl der Spalten ($2^t$), der Anzahl der Zeilen ($2^{|\tilde{P}|}$) und der Wahrscheinlichkeit, dass in einer Zelle der Erfolgsmatrix eine Eins steht ($2^{-t+1}$) berechnet.

Von mindestens $2^{|\tilde{P}|+1}$-vielen Einsen sind also nur maximal $2^{|\tilde{P}|}$-viele in nicht-schweren Zeilen – der Rest der Einsen (also mindestens die Hälfte alle Einsen) liegt also in schweren Zeilen. □

**Laufzeit von AL.** Es gilt für die erwartete Anzahl von Wiederholungen von Stufe 1:

$$\begin{aligned} E_w[u] &= \sum_{j=0}^{\infty} (j+1)(1-\varepsilon)^j \varepsilon = \sum_{j=0}^{\infty} j(1-\varepsilon)^{j-1} \varepsilon \\ &= \varepsilon \cdot \sum_{j=0}^{\infty} j(1-\varepsilon)^{j-1} = \frac{\varepsilon}{(1-(1-\varepsilon))^2} = \varepsilon^{-1}, \end{aligned}$$

wegen

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x} \quad , \quad \sum_{i=0}^{\infty} i \cdot x^{i-1} = \frac{1}{(1-x)^2}.$$

Im Mittel wurden also in Stufe 1 $\varepsilon^{-1}$-viele Schritte benötigt. Damit ist die Erwartete Laufzeit von Stufe 1 gleich $\varepsilon^{-1}|\tilde{P}| \leq 2^{t-1}|\tilde{P}|$. Nach $\varepsilon^{-1}$-vielen Schritten (also Aufrufen von $\tilde{P}$) wird also sehr Wahrscheinlich in Stufe 1 eine 1 in der Erfolgsmatrix gefunden. Wie groß die Wahrscheinlichkeit ist, dass der Algorithmus wesentlich weniger oder mehr Schritt benötigt, wollen wir nun mit der Chernov-Ungleichung berechnen:

$$\begin{aligned} \mathrm{Ws}\left[u < \varepsilon^{-1}\right] &\overset{\varepsilon^{-1} \in \mathbb{Z}}{=} \varepsilon \sum_{i=0}^{\varepsilon^{-1}-1} (1-\varepsilon)^i = 1-(1-\varepsilon)^{\varepsilon^{-1}} \\ &\approx 1-e^{-1} < \frac{1}{2}. \\ \mathrm{Ws}\left[k\varepsilon^{-1} \leq u < (k+1)\varepsilon^{-1}\right] &\approx e^{-k} - e^{-k-1}. \\ \mathrm{Ws}\left[\frac{1}{k+1}\varepsilon^{-1} \leq u < \frac{1}{k}\varepsilon^{-1}\right] &\approx e^{-1/(k+1)} - e^{-1/k}. \end{aligned}$$

Dann ist die Wahrscheinlichkeit, dass Stufe 2 von AL ein $\bar{e}$ findet mit $\mathrm{ERF}_{w,\bar{e}} = 1$ und $w$ schwer

$$\begin{aligned} & \mathrm{Ws}\left[\text{Stufe 2 findet } \bar{e} \text{ mit } \mathrm{ERF}_{w,\bar{e}} = 1 \mid w \text{ schwer}\right] \\ (*) \quad & \geq \sum_{k=1}^{\infty} \left(e^{-k} - e^{-k-1}\right)\left(1 - e^{-k}\right) + \sum_{k=1}^{\infty} \left(e^{-1/(k+1)} - e^{-1/k}\right)\left(1 - e^{-1/k}\right) \\ & > \frac{1}{2}. \end{aligned}$$

Damit ist die Wahrscheinlichkeit, dass Stufe 2 ein $\bar{e} \neq e$ findet mit $\mathrm{ERF}_{w,\bar{e}} = 1$

$$\begin{aligned} & \mathrm{Ws}\left[\text{Stufe 2 findet } \bar{e} \neq e \text{ mit } \mathrm{ERF}_{w,\bar{e}} = 1\right] \\ & \geq \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}, \end{aligned}$$

also dem Produkt aus den Wahrscheinlichkeiten, dass $w$ schwer, $e \neq \bar{e}$ ist und der Wahrscheinlichkeit in $(*)$.

**Mittlere Anzahl der Proben von AL:** Damit ist die mittlere Anzahl der Proben in Stufe 1 von AL gleich

$$E_{w_{AL}}\left[\sum u\right] \quad < \quad 8 \cdot \varepsilon^{-1}.$$

Die Stufen 1 und 2 werden also im Mittel $\leq 8$-mal iteriert. Somit ist die erwartete Anzahl der Proben in AL

$$E_{w_{AL}}\left[\#\text{Proben von } AL\right] < 2 \cdot 8 \cdot \varepsilon^{-1} = 16 \cdot \varepsilon^{-1}.$$

Ist andererseits $\varepsilon$ bekannt und Stufe 2 macht nicht $u$ sondern $\varepsilon^{-1}$-viele Proben, dann hat Stufe 2 mit Ws $\frac{1}{4}\left(1 - e^{-1}\right) > \frac{1}{6{,}5}$ Erfolg. Dann gilt

$$E_{w_{AL}}\left[\#\text{Proben von } AL\right] < 13 \cdot \varepsilon^{-1}.$$

**Fazit:**
Entweder ist $\varepsilon < 2^{-t+1}$ oder $E_w|AL| \leq 16|\tilde{P}|\varepsilon^{-1}$. Ist $\varepsilon \geq 2^{-t+1}$, dann liefert $AL : h \mapsto \log_g h$ den DL mit der erwarteten Schrittzahl die $\tilde{P}$ zum Erfolg führt.
Nach [**Stinson1995**] waren die Hauptkriterien beim Entwurf des Schnorr-Schemas Geschwindigkeit und Effizenz, sowohl vom rechnerischen Standpunkt aus betrachtet als auch bzgl. der Information, die ausgetauscht wird. In [**Schnorr1991**] wird die gute Verwendbarkeit mit Smartcards hervorgehoben. Ein Vergleich mit anderen Schemata wie Fiat-Shamir und RSA zeigt, dass sowohl die Schlüssel kürzer, als auch die Anzahl der Multiplikationen weniger sind. Zudem können viele Operationen bereits im Voraus berechnet werden, so dass nur sehr wenige Berechnungen wirklich „online" durchgeführt werden müssen.

Nach Satz 2.1.2 auf Seite 22 ist die DL-Identifikation sicher gegen Angriffe aus dem Stand, sofern der DL schwer ist: $DL \leq_{pol} \tilde{P}$.

Wir unterscheiden **passive** und **aktive Angriffe**:

**Passiver Angriff.** Der Angreifer $C$ hört $(P,V)$ ab und versucht sich dann als $\tilde{P}$.

**Aktiver Angriff.** $C$ fungiert erst als $\tilde{V}$ in $(P,\tilde{V})$ und dann versucht sich dann als $\tilde{P}$ in $(\tilde{P},V)$ („man in the middle attack").

Die Frage, ob bei einem aktiven oder passiven Angriff Informationen über $x$ preisgegeben werden bestimmt die Sicherheit des Verfahrens. So ist in $(P,V)$ das Tripel $(g^r,e,y)$ zwar komponentenweise, jedoch nicht als ganzes Tripel zufällig (so hängt $y$ von $r$ und $e$ ab).

DEFINITION 2.1.5. **Zero-knowledge Protokolle** geben keine Informationen preis.

Ein Beweissystem $(P,V)$ ist genau dann zero-knowledge über $L$, wenn ein Simulator $M$ existiert, der in erwartet polynomialer Zeit läuft, so dass für eine beliebige prob. pol. Zeit $V'$ für eine Eingabe $x \in L$ mit Zeugen $w$ und einer Hilfseingabe zu $V'y$ zwei Ensemble[1] $V'_{P(x,w)}(x,y)$ und $M(x,V'(x,y))$ gibt die polynomiell ununterscheidbar sind. $M$ ist es dabei erlaubt $V'$ in einem Unteraufruf zu benutzen.

Nach [**FeigeFiatShamir1988**] ist ein interaktives Beweissystem zur Zugehörigkeit in $L$ zero-knowledge, wenn für alle Eingaben aus $L$, alle $V$ ihre Sicht der Kommunikation in $(P,V)$ von einer pol. Zeit prob. Turing-Maschine $M$ erzeugen lassen können, die eine ausgezeichnete Wahrscheinlichkeitsverteilung besitzt.

NOTE 2.1.6. In [**Salomaa1996**, Kapitel 6] ist ein Protokoll zero-knowledge gdw. gilt

(1) Der Prover kann den Verifier nicht betrügen.
Wenn $P$ das Geheimnis nicht kennt, so sind seine Chancen $V$ zu betrügen vernachlässigbar klein.
(2) Der Verifier kann nicht den Prover betrügen.
Aus den Informationen, die bei dem Protokoll „freigesetzt" werden bekommt sie keinen Hinweis auf das Geheimnis. Insbesondere kann $V$ das Geheimnis keinem Anderen beweisen ohne es selbst zu kennen.
(3) $V$ lernt von dem Protokoll nichts, was sie nicht ohne Hilfe von $P$ lernen könnte.
$V$ kann also das Protokoll selbst simulieren.

Während die ersten beiden Eigenschaften lediglich für das Bit-Commitment notwendig sind, ist die 3. Eigenschaft die entscheidende für ein zero-knowledge Protokoll.

Dazu folgende Definitionen:

DEFINITION 2.1.7. Das Protokoll $(P,V)$ heisst **perfekt zero-knowledge**, wenn es einen prob. pol. Zeit Simulator $\varphi$ gibt mit

$$\varphi : (\tilde{V},P) \mapsto (\bar{g},e,y) \in G \times \left[0,2^t-1\right] \times \mathbb{Z}_q,$$

so dass $\varphi\left(\tilde{V},h\right)$ genau wie in $(P,\tilde{V})$ verteilt ist. Dabei wird ein Aufruf von $\tilde{V}$ als ein Schritt gerechnet.

Die Verteilung von $(P,\tilde{V})$ kann also ohne den geheimen Schlüssel $x$ erzeugt werden.

Ein **honest verifier zero-knowledge** Protokoll liegt vor, wenn der Simulator $\varphi$ nur für $V$ (statt für beliebige $\tilde{V}$) korrekt simuliert.

NOTE 2.1.8. Bei einem perfekt zero-knowledge Protokoll kann $V$ also keine Informationen über das Geheimnis abgesehen von dessen Existenz erhalten, im Gegensatz zu nicht perfekten Protokollen, bei denen $V$ Informationen erhält, die online oder in Polynomialzeit benutzt werden können. Beispiel ist z.B. der Einsatz von RSA beim Verschlüsseln

[1] Unter einem **Ensemble** verstehen wir eine Folge von Wahrscheinlichkeits-Verteilungen.

des Commitments, das nicht perfekt ist, da es keinen Beweis für die nicht-Existenz eines Faktorisierungsalgorithmus in Linearzeit gibt.

LEMMA 2.1.9. *Zu $(P,V)$ gibt es einen **perfekten Simulator***

$$\varphi : \left(\tilde{V},h\right) \mapsto (\bar{g},e,y) \quad mit \quad E_w|\varphi\left(\tilde{V},h\right)| \leq |\tilde{V}|2^t.$$

BEWEIS. $|\tilde{V}|$ schließt $|\tilde{P}|$ mit ein.

(1) $\varphi$ wählt $r \in_R \mathbb{Z}_q$, $\tilde{e} \in_R [0,2^t[$ und setzt
$$\bar{g} := g^r h^{-\tilde{e}} \quad , \quad e := e\left(\tilde{V},\bar{g}\right) \in \left[0,2^t\right[,$$
wie $\tilde{V}$ mit $\bar{g}$.
(2) **if** $e \neq \tilde{e}$ **then** reset $\tilde{V}$ and restart. **else** $y := r$.

Die Verteilung von $(\bar{g},e,y)$ ist wie bei $(P,\tilde{V})$: $(\bar{g},e) \in_R G \times [0,2^t[$, $y$ ist eindeutig bestimmt durch $g,\bar{g},e$ und $\bar{g} = g^y h^{-e}$. □

DEFINITION 2.1.10. $2^t$ ist genau dann **polynomial**, wenn gilt, dass $2^t$ poly(Bitlänge von $h$), $2^t$ also polynomial in der Bitlänge von $h$ ist.

PROPOSITION 2.1.11. *Die DL-Identifikation ist perfekt zero-knowledge wenn $2^t$ polynomial ist.*

**Beachte:** $P$ ist polynomialzeit, d.h. $|P|$=poly(Bitlänge $h$). Im Fall $G \subset \mathbb{Z}_p^*$ gilt:

$$\begin{aligned} & 2^t \text{ polynomial} \\ \Leftrightarrow \quad & t = O(\lg\lg p) \\ \Leftrightarrow \quad & 2^t = (\lg p)^{O(1)}. \end{aligned}$$

**Fazit:**
Die DL-Identifikation hat eine Betrugswahrscheinlichkeit $\leq 2^{-t}$ – also ein Sicherheitsniveau $2^t$. Aber nur dann wenn $2^t$ polynomial ist wird nachweislich keine Information an $\tilde{V}$ preisgegeben.
Bezüglich der Signatur-Erzeugung ist die Schnorr-Signatur der Star mit 0 online-Multiplikationen. Dieser Wert wird durch ein Preprocessing ermöglicht.

## 2.2. $k$-fach sequentielle DL-Identifikation

Das Protokoll $(P^k,V^k)$ wiederholt das Protokoll $(P,V)$ $k$-mal mit unabhängigen $r_1,\ldots,r_k \in_R \mathbb{Z}_q$ und $e_1,\ldots,e_k \in_R [0,2^t[$. Entsprechendes gilt für $(P^k,\tilde{V})$ und $(\tilde{P},V^k)$.

LEMMA 2.2.1. *Zu $(P^k,V^k)$ gibt es ein prob. pol. Zeit $\tilde{P}$ mit $ERF(\tilde{P},h) = 2^{-kt}$, wobei gilt*

$$ERF\left(\tilde{P},h\right) = Ws_w\left[(\tilde{P},V)\, akzeptiert\ mit\ h\right],$$

*wobei w gleich der Anzahl der Zufallsbits von $(\tilde{P},V^k)$ ist.*

BEWEIS. $\tilde{P}$ wählt $\tilde{e}_1,\ldots,\tilde{e}_k \in_R \mathbb{Z}_q$ und agiert wie $\tilde{P}$ von Lemma 2.1.1 auf Seite 22. Falls $(\tilde{e}_1,\ldots,\tilde{e}_k) = (e_1,\ldots,e_k)$ dann hat $\tilde{P}$ Erfolg. □

Lemma 2.2.1 gilt für beliebige $\tilde{V}$ statt $V$, sofern ein Aufruf von $\tilde{V}$ als ein Schritt gezählt wird.

PROPOSITION 2.2.2. *Zu $(P^k, V^k)$ gibt es einen prob. Algorithmus*

$$AL : (\tilde{P}, h) \mapsto \log_g h \quad \textit{mit} \quad E\,|AL| = O(|\tilde{P}|/\varepsilon),$$

*sofern $\tilde{P}$ eine Erfolgswahrscheinlichkeit $\varepsilon \geq 2^{-tk+1}$ für $(\tilde{P}, V^k)$ hat.*

BEWEIS. Der Algorithmus $AL(\tilde{P}, h)$ funktioniert analog zu Algorithmus 5 auf Seite 23 und ist in Algorithmus 6 beschrieben. □

**Algorithm 6** $AL(\tilde{P}, h)$ zu $(P, V)_k$

(1) $u := 0$;
*** Stufe 1: ***
(2) **do**
(a) $\bar{g} := \bar{g}_1\left(\tilde{P}, w\right), y_1 := y(\tilde{P}, w, e_1)$.
(b) **for** $i = 2, \ldots, k$ **do**
(i) $\bar{g}_i := \bar{g}_i(\tilde{P}, w, e_1, \ldots, e_{i-1}), y_i := y(\tilde{P}, w, e_1, \ldots, e_{i-1})$
(c) $u := u + 1$;
(3) **until** $\left(\mathrm{ERF}_{w,\underline{e}} = 1\right)$;
*** Stufe 2: ***
*** In $u$ stehen #Versuche beim Erzeugen von $w, \mathrm{g}, \underline{\mathrm{y}}, \underline{\mathrm{e}}$ ***
(4) Analog zu Algorithmus 5 Schritt 4.

**Die Erfolgsmatrix.** Wieder wird die Erfolgsmatrix zur Modellierung der Wahrscheinlichkeiten eingesetzt (siehe Abbildung 2.2.1).

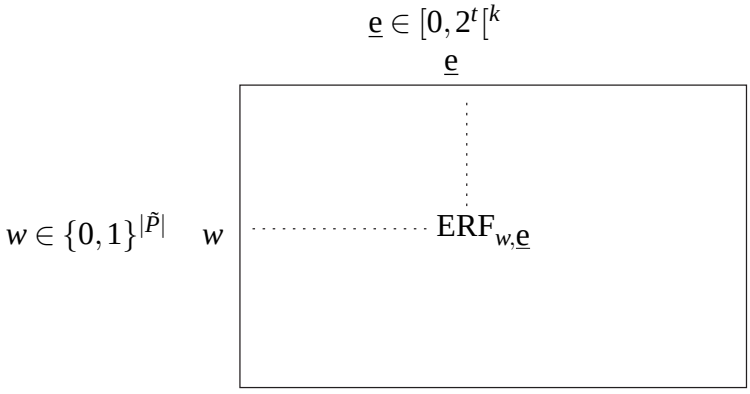



ABBILDUNG 2.2.1. Erfolgsmatrix für die $k$-fach sequentielle Identifikation

Ziel von AL ist es wieder zwei Einsen in derselben Zeile $w$ zu finden. Zwei solche Einsen liefern dann $\log_g h$. Wegen $\varepsilon \geq 2^{-tk+1}$ ist der Anteil der Eins-Einträge $\geq 2^{-tk+1}$. Mindestens die Hälfte der Einsen ist in schweren Zeilen (mit $\geq 2$ Einsen).

**Fazit:**
Entweder gilt $\varepsilon < 2^{-tk+1}$ oder $E\,|AL| \leq O(|\tilde{P}|/\varepsilon)$.

LEMMA 2.2.3. *Zu $(P^k,V^k)$ gibt es einen perfekten Simulator*

$$\varphi : (\tilde{V},h) \to (\underline{\bar{g}},\underline{e},\underline{y})$$

*mit $E\left|\varphi(\tilde{V},h)\right| \leq (|P^k|+|\tilde{V}|)2^t$, wobei $\underline{g} = (\bar{g}_1,\ldots,\bar{g}_k)$, $\underline{e} = (e_1,\ldots,e_k)$, $\underline{y} = (y_1,\ldots,y_k)$.*

BEWEIS. (Analog zu Lemma 2.1.9 auf Seite 27) $\varphi$ simuliert die $k$ Runden iterativ. Sobald $\tilde{e}_1 = e_1,\ldots,\tilde{e}_i = e_i$ probt er $\tilde{e}_{i+1}$ solange bis $\tilde{e}_{i+1} = e_{i+1}$. Dabei wird nur Runde $i+1$ zurückgedreht. □

PROPOSITION 2.2.4. *Die k-fach sequentielle DL-Identifikation ist perfekt zero-knowledge falls $2^t$ polynomial ist, d.h.*

$$t = O(\lg(\textit{Laenge der Eingabe}))$$

*mit z.B. Länge der Eingabe gleich* $\lg p$ *falls* $G \subset \mathbb{Z}_p^*$.

REMARK 2.2.5. $\varphi$ in Lemma 2.2.3 ist um $2^{tk}2^{-t}$ schneller als jedes $\tilde{P}$ in Satz 2.2.2 auf der vorherigen Seite sofern DL schwer ist.

**Fazit:**
Die $k$-fach sequentielle DL-Identifikation mit $t = O(1)$ ist sicher gegen passive Angreifer falls der DL schwer ist.

**Aktive Angreifer.** $A$ fordert poly-viele Identifikationen von $P^k$ und versucht dann $P^k$ zu personifizieren: $l$-mal $(P^k,\tilde{V})$, dann $(\tilde{P},V^k)$ mit $\tilde{V} = \tilde{V}_A$, $\tilde{P} = \tilde{P}_A$.

PROPOSITION 2.2.6. *Zu $(P^k,V^k)$ mit $t = O(1)$ gibt es prob. Alg.*

$$AL : (A,h) \mapsto \log_g h \quad \textit{mit} \quad E_{w_{AL}}|AL| = O(|AL|/\varepsilon),$$

*sofern A Erfolgswahrscheinlichkeit $\varepsilon \geq 2^{-tk+1}$ hat.*

BEWEIS. (Idee) Der Algorithmus AL arbeitet im Prinzip wie AL von Satz 2.2.2 – nur wird vor jedem Aufruf von $\tilde{P}$ das Protokoll $l$-mal $(P^k,\tilde{V})$ durch einen pol.-Zeit Simulator perfekt simuliert. □

**Fazit:**
Die $k$-fach sequentielle DL-Identifikation ist – für $2^t$ polynomiell – sicher gegen aktive Angreifer sofern der DL schwer ist.

**Man-in-the-middle-Attacke.** (Desmedt, Goutier, Bengo, LNCS 293 (1988), CRYPTO 87) $A$ personifiziert $P$ in $(\tilde{P},V)$, indem er die Fragen von $V$ in $(\tilde{P},V)$ von $P$ beantworten lässt.

### 2.3. DL-Identifikation mit kurzer Kommunikation

**Öffentlich:** $g, G = \langle g \rangle, |G| = q\text{prim}, h = g^x$ und die Hash-Funktion $H$ mit

$$H : G \times \{0,1\}^* \to [0, 2^t[$$

wie bei den Schnorr-Signaturen.

**Privat:** $x \in_R \mathbb{Z}_q$.

Das Protokoll verläuft wie folgt:

| $(P,V)_I$ | $P$ | | $V$ |
|---|---|---|---|
| 1. | | $\leftarrow m$ | $m \in_R \{0,1\}^*$ |
| 2. | $r \in_R \mathbb{Z}_q$, $e := H(g^r, m)$, $y := r + xe \in \mathbb{Z}_q$ | $e, y \rightarrow$ | |
| 3. | | | $H(g^y h^{-e}, m) \stackrel{?}{=} e$ |

$V$ generiert also eine zufällige Nachricht $m$ und schickt sie an $P$, der daraufhin eine **Schnorr-Unterschrift** $(e,y)$ zu $m$ generiert und diese an $V$ zurückschickt.

$\tilde{P}$ kann $m$ nicht zur Personifizierung benutzen, weil $m$ zufällig ist. Die man-in-the-middle Attacke wird nicht verhindert!

**Länge der Kommunikation.**

$$\begin{array}{lrcll} (P,V)_I & |(m,e,y)| & = & t+t+\lg q & = & 320 \\ (P,V) & |(g^r,e,y)| & = & |g^r|+t+\lg q & & \\ & G \subset \mathbb{Z}_p^* & = & 1024+80+160 & = & 1264 \\ & G \subset E_{A,B}(\mathbb{F}_q) & = & 2\lg q+80+160 & = & 540 \end{array}$$

Hierbei werden als Parametergrößen $t = 80$ und $\log_2 q = 160$ empfohlen.

**Random Oracle Modell (ROM).**

DEFINITION 2.3.1. Unter dem **Random Oracle Modell (ROM)** verstehen wir eine Zufallsfunktion $H : G \times \{0,1\}^* \to [0, 2^t - 1]$ die zufällig aus allen Funktionen dieses Typs nach der Gleichverteilung gezogen wurde. $H$ wird als $H$-Orakel modelliert.

$\big(H(g_1,m_1),\ldots,H(g_l,m_l)\big) \in_R [0,2^t[^l$ ist zufällig, gleichverteilt für verschiedene $(g_1,m_1),\ldots,(g_l,m_l)$.

NOTE. In [**PointchevalStern1996**] wird verlangt, dass im Random Oracle Model eine Hashfunktion als ein Orakel angesehen werden kann, das für jede neue Anfrage einen zufälligen Wert liefert. Auf identische Anfragen müssen jedoch gleiche Antworten geliefert werden. Beweise, denen dieses Modell zu Grunde liegt, belegen die Sicherheit eines Signatur-Schemas nur dann, wenn die verwendete Hashfunktion keine Schwachstellen hat.

REMARK 2.3.2. In $(P,V)_I$ ist der aktive Angreifer $A$ eingeschränkt, denn $\tilde{V}$ kann die Frage $e$ nicht frei wählen. $\tilde{V}$ lernt durch die Unterschriften zu Nachrichten seiner Wahl also nichts.

PROPOSITION 2.3.3. *Sei $\tilde{P}$ ein Angreifer auf $(P,V)_I$ aus dem Stand mit $l$ $H$-Aufrufen. Im ROM gibt es einen prob. Alg.*

$$AL : (\tilde{P}, h) \to \log_g h \quad \textit{mit} \quad E_{H,w}|AL| = O(l|\tilde{P}|/\varepsilon),$$

*sofern $\tilde{P}$ Erfolgswahrscheinlichkeit $\varepsilon \geq 2^{-t+1} l$ hat.*

Zum Beweis wird noch folgender Korollar benötigt:

COROLLARY 2.3.4. *Es gilt: DL $\leq_{pol}$ Angriffe aus dem Stand auf $(P,V)_I$.*

BEWEIS. (Satz 2.3.3) O.B.d.A. gelte $l \leq 2^{t-1}$.

LEMMA 2.3.5. *Hat $\tilde{P}$ für $(m,e,y)$ mit $Ws_H > 2^{-t}$ Erfolg, dann hat $\tilde{P}$ das H-Orakel über $H(g^y h^{-e}, m)$ befragt.*

BEWEIS. Andernfalls gilt im ROM $Ws_H[H(g^y h^{-e}, m) = e] \leq 2^{-t}$. □

□

Damit können wir o.B.d.A. annehmen, dass $\tilde{P}$ nur mit solchen $(H,e,y,m)$ Erfolg hat, für die $H(g^y h^{-e}, m)$ vor der Berechnung von $(y,e)$ erfragt wurde. Die erfragten H-Werte seien $H(g_1,m_1),\ldots,H(g_l,m_l)$. Dabei hängt $(g_{i+1},m_{i+1})$ beliebig von $H(g_1,m_1),\ldots,H(g_i,m_i)$ ab.

AL benutzt $\tilde{P}$ zusammen mit einer statistisch unabhängigen Hashfunktion $H$.

LEMMA 2.3.6. *Angenommen, $\tilde{P}$ hat mit $(g_i,m_i)$ für $(H,e,y)$, $(\bar{H},\bar{e},\bar{y})$, $(e,y) \neq (\bar{e},\bar{y})$ Erfolg. Dann gilt*

$$\log_g h = \frac{y-\bar{y}}{e-\bar{e}}.$$

BEWEIS. Weil $\tilde{P}$ zur $H$-Anfrage $(g_i,m_i)$ für $(H,e,y)$ und $(\bar{H},\bar{e},\bar{y})$ Erfolg hat, gilt

$$g^y h^{-e} = g_i = g^{\bar{y}} h^{-\bar{e}}$$

und somit $\log_g h = \frac{y-\bar{y}}{e-\bar{e}}$. □

Wir beweisen nun Satz 2.3.3 auf der vorherigen Seite:

BEWEIS. Grob skizziert sieht Algorithmus AL folgendermaßen aus: AL sucht zu festem $(g_i,m_i)$ ein zufälliges $w_{\tilde{P}} \in \{0,1\}^{|\tilde{P}|}$ und zufällige $H(g_i,m_i)$, $\bar{H}(g_i,m_i)$, für die $\tilde{P}$ mit $(e,y)_{H,w_{\tilde{P}}}$ und $(\bar{e},\bar{y})_{\bar{H},w_{\tilde{P}}}$ Erfolg hat und $e \neq \bar{e}$. Nach Lemma 2.3.6 liefert dieses $\log_g h$.

In Stufe 1 probt AL solange zufällige $w_{\tilde{P}}$ und $H(g_i,m_i)$ bis $\tilde{P}$ mit $(e,y)_{H,w_{\tilde{P}}}$ Erfolg hat. Sei $u$ die Anzahl der Proben.

In Stufe 2 probt AL zu diesem $w_{\tilde{P}}$ bis zu $u$ unabh. Zufallswerte $\bar{H}(g_i,m_i)$ bis zum zweiten Erfolg mit $(\bar{e},\bar{y})_{\bar{H},w_{\tilde{P}}}$.

Dabei ist $i$ fest, die Werte $(g_1,m_1),\ldots,(g_i,m_i)$,

$$H(g_1,m_1) = \bar{H}(g_1,m_1),\ldots,H(g_{i-1},m_{i-1}) = \bar{H}(g_{i-1},m_{i-1})$$

sind durch $w_{\tilde{P}}$, $\tilde{P}$ bestimmt.

Für festes $i$, $1 \leq i \leq l$ hat die Erfolgsmatrix die Form wie in Abbildung 2.3.1 angegeben. □

FACT 2.3.7. *$\exists i$, $1 \leq i \leq l$, so dass $\tilde{P}$ mit $H(g_i,m_i)$ mit $Ws_{H,w_{\tilde{P}}} \geq 2^{-t+1}$ Erfolg hat.*

BEWEIS. Nach Voraussetzung hat $\tilde{P}$ mit $Ws_{H,w_{\tilde{P}}} \geq 2^{-t+1} l$ Erfolg.

Für festes $i$ arbeitet AL wie AL von Satz 2.1.2 auf Seite 22. Die erwartete Anzahl der Proben von AL ist dabei $\leq 16\varepsilon^{-1}$ sofern $\tilde{P}$ mit $i$ Erfolgswahrscheinlichkeit $\varepsilon \geq 2^{-t+1}$ hat. □

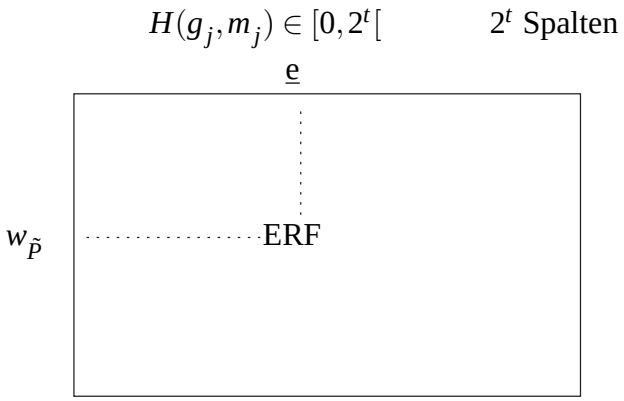



ABBILDUNG 2.3.1. Erfolgsmatrix bei der DL-Identifikation mit kurzer Kommunikation

Wir versuchen nun, Satz 2.3.3 auf Seite 30 auf aktive Angreifer $A$ zu übertragen: $A$ führt dazu $l$-mal $(P,\tilde{V})_I$ und dann $(\tilde{P},V)_I$ mit $\tilde{V} = \tilde{V}_A$, $\tilde{P} = \tilde{P}_A$ aus.

**Kernidee:** AL erzeugt die Verteilung $(P,\tilde{V})_I$ selbst ohne $x$. AL wählt $r \in_R \mathbb{Z}_q$, $e \in_R [0, 2^t[$ und setzt

$$H(g^y h^{-e}, m_{\tilde{V}}) = e.$$

AL erzeugt damit exakt die Verteilung von $(P,V)_I$ im ROM. Dieser Schritt zählt als Orakel-Anfrage und eine nochmalige, echte Befragung des $H$-Orakels zu $H(g^y h^{-e}, m_{\tilde{V}})$ wird verboten.

PROPOSITION 2.3.8. *Es sei $A$ ein aktiver Angreifer zu $(P,V)_I$, der das $H$-Orakel $l$-mal befragt. Im ROM gibt es einen prob. Alg. $AL : (A,h) \mapsto \log_g h$ mit $E_{h,w}|AL| = O(l|A|/\varepsilon)$, sofern $A$ Erfolgswahrscheinlichkeit $\varepsilon \geq 2^{-t+1}$ hat.*

Die Anzahl $l$ der $H$-Orakel Fragen spielt dieselbe Rolle wie im Bew. von Satz 2.3.3 auf Seite 30. Nach Satz 2.3.8 gilt also

$$DL \leq_{pol} \text{aktive Angriffe zu } (P,V)_I.$$

**Chosen Message Attack (CMA) auf Schnorr Signaturen.** Ein Angriff könnte folgendermassen aussehen:

(1) Der Angreifer $A$ erzeugt Signaturen zu Nachrichten seiner Wahl mittels des Signatur-Orakels.
(2) Dann erzeugt er eine neue Signatur – aber dieses Mal ohne das Signatur-Orakel.

PROPOSITION 2.3.9. *Sei $A$ ein CMA-Angreifer auf Schnorr-Signaturen mit $l$ Aufrufen des $H$-Orakels. Im ROM gibt es einen prob. Alg. $AL : (A,h) \mapsto \log_g h$ mit $E_{H,w}|AL| = O(l|A|/\varepsilon)$, sofern $A$ Erfolgswahrscheinlichkeit $\varepsilon \geq 2^{-t+1} l$ hat.*

COROLLARY 2.3.10. *Im ROM sind Schnorr-Signaturen sicher gegen CMA sofern DL schwer ist.*

BEWEIS. Der Beweis von Satz 2.3.9 ist analog zu Satz 2.3.3 und Satz 2.3.8. Der Angreifer $A$ erzeugt für die CMA-Attacke die erforderlichen Unterschriften, indem er die

Werte von $H$ selbst zufällig wählt. Jede solche Erzeugung eines $H$-Wertes gilt als Orakel-Aufruf und die nochmalige, echte Befragung des Orakels an derselben Stelle ist verboten. □

**Historie.** Der Vorschlag $H : G \times \{0,1\}^* \to [0, 2^t[$ als Zufallsfunktion zu betrachten geht zurück auf [**FiatShamir1986**]. Weiter ausgearbeitet wurde das ROM durch Bellare, Rogaway in [**BellareRogaway1993**]. Das ROM ist weithin anerkannt aber auch umstritten. Es bedeutet, dass die Werte $H(g_i, m_i)$ für $i = 1, \ldots, l$, die ein Angreifer auswählt, statistisch unabhängig sind.
Der Algorithmus AL von Satz 2.1.2 auf Seite 22 wurde eingeführt in [**FeigeFiatShamir1988**] für das Fiat-Shamir-Schema. Die Übertragung von AL auf die DL-Identifkation ist in [**Schnorr1991**] erschienen.
Die Übertragung von Satz 2.1.2 auf DL-Signaturen mit stat. unabhängigen $H$-Orakeln steht erstmals in [**PointchevalStern1996**] (Orakel Replay Attacke, Forking Lemma). Das dort bewiesene Lemma 1 ist viel schwächer als Satz 2.3.9.

**Faustregel.** Die Analyse einer 3-Runden Identifikation ist einfacher als die des zugehörigen Signatur-Verfahrens. Ist die Identifikation sicher gegen aktive Angriffe, dann sind die zugehörigen Signaturen sicher gegen CMA sofern die Hashfunktion keine Schwächen hat.

## 2.4. Abwehr der Man-in-the-middle-Attack

**Idee:** $P$ muss wissen, mit welchem $V$ er kommuniziert. Dazu fordert $P$ den öffentlichen Schlüssel $h_V$ von $V$ an.

Das Protokoll läuft dann wie folgt ab:

| $(P,V)_{II}$ | $P$ | $\leftarrow h_V \rightarrow$ | $V$ |
|---|---|---|---|
| 1. | | $\leftarrow m$ | $m \in_R \{0,1\}^*$ |
| 2. | $r \in_R \mathbb{Z}_q$, $e := H(g^r, (m, h_V))$, $y := r + xe \in \mathbb{Z}_q$ | $e, y \rightarrow$ | |
| 3. | | | $H(g^y h^{-e}, (m, h_V)) \stackrel{?}{=} e$ |

Die Kommunikation in $(P,V)_{II}$ identifiziert $P$ und $V$. Die Identifikation ist flüchtig, ohne bleibende Bedeutung. Häufig sind man-in-the-middle-Attacken und aktive Attacken durch die Situation praktisch ausgeschlossen, z.B. bei der Identifikation einer Chipkarte gegenüber dem Geldausgabeautomat (GA).

Die Identifizierung des GA gegenüber der Chipkarte ist kryptographisch sinnlos und dient nur zu Prüfzwecken. Ein betrügerischer GA erfragt die Pinnummer, behält die Karte und meldet „Störung".

**Schwächere Annahmen als ROM.**

DEFINITION 2.4.1. $H$ ist **kollisionsresistent**, wenn keine Kollision $(w, w')$ mit $w \neq w'$, $H(w) = H(w')$ bekannt ist.

In der asymptotischen Theorie verlangt man, dass Kollisionen nicht in pol. Zeit konstruierbar sind.

LEMMA 2.4.2. *Damit DSA-Signaturen sicher gegen CMA sind, muss H kollisionsresistent sein.*

BEWEIS. Ang. $H(m) = H(m')$, dann ist jede DSA-Signatur von $m'$ auch DSA-Signatur von $m$. □

Lemma 2.4.2 gilt nicht für Schnorr-Signaturen. Kollisionen $H(\overline{g}, m) = H(\overline{g}', m')$ mit $\overline{g} \neq \overline{g}'$ sind unschädlich.

LEMMA 2.4.3. *Damit Schnorr-Signaturen sicher gegen CMA sind, muss $H(\overline{g}, *)$ für fast alle $\overline{g}$ schwer invertierbar sein.*

BEWEIS. Andernfalls erfragt $A$ Signaturen $(e, y)$ zu zufälligen $m$ und löst $H(g^y h^{-e}, m') = e$ nach $m'$. Dann ist $(e, y)$ Signatur für $m'$. □

In [**FeigeFiatShamir1988**] werden die Begriffe „completeness" und „sound" im Rahmen von Identifikationen definiert:

DEFINITION. Ein Paar interaktiver prob. Polynomialzeit Turingmaschinen $\overline{A}, \overline{B}$ werden **interaktives Beweissystem über das Wissen für das Polynomialzeit Prädikat** $P(I, S)$ genannt, wenn gilt:

(1) **Completeness:** Für alle $I$, für die $P(I, S)$ erfüllbar ist ist die Ausführung von $(\overline{A}, \overline{B})$ für die Eingabe $I$ mit sehr großer Wahrscheinlichkeit erfolgreich. Formal: $\forall a, \exists c, \forall |I| > c$ : **if** $\overline{A}$ hat auf seinem „Wissensband" ein $S$,so dass $P(I, S)$, und $\overline{B}$ hat einen leeren String auf seinem „Wissensband" **then**

$$Ws((\overline{A}, \overline{B}) \text{ accepts } I) > 1 - 1/|I|^a.$$

Wenn wir es mit dem echten Prover $\overline{A}$ zu tun haben, so enthält sein „Wissensband" ein geeignetes $S$. Das Wissensband des Verifiers wird als leer angenommen.

(2) **Soundness:** Es gibt eine prob. pol.zeit Turing Maschine $M$ (mit voller Kontrolle über $A$), so dass es für alle $A$, jeden beliebigen initialen Inhalt von $A$'s Wissensband $KA$ und Zufallsband $RA$, ein genügend langes $I$ gibt, so dass, wenn die Ausführung von $(A, \overline{B})$ bei der Eingabe $I$ mit einer nicht zu vernachlässigenden Wahrscheinlichkeit erfolgreich ist, dann genügt die Ausgabe, die von $M$ am Ende der Ausführung von $M(A, RA, KA)$ auf die Eingabe $I$ produziert wird, dem Prädikat $P$ mit sehr großer Wahrscheinlichkeit. Formal:
$\forall a, \exists M, \forall b, \forall A, \exists c, \forall |I| > c, \forall RA, \forall KA$ gilt: $Ws((A, \overline{B})$ akzeptiert $I) > 1/|I|^a \Rightarrow$ $Ws(\text{Ausgabe von } M(A, RA, KA) \text{ fuer } I \text{ genuegt } P) > 1 - 1/|I|^b$.

## 2.5. Okamoto DL-Identifikation

[**Okamoto1992**]

**Ziel:** Beweisbar sichere DL-Identifikation ohne ROM.
**Öffentlich:** $g_1, g_2, G = \langle g_1 \rangle = \langle g_2 \rangle$ und $|G| = q$
**Private-Key:** $(x_1, x_2) \in_R \mathbb{Z}_q^2$.
**Public-Key:** $h = g_1^{x_1} g_2^{x_2}$.

Das Protokoll hat folgenden Ablauf:

| $(P,V)_{Ok}$ | $P$ | | $V$ |
|---|---|---|---|
| 1. | $r_1, r_2 \in_R \mathbb{Z}_q$ | $\bar{g} := g_1^{r_1} g_2^{r_2} \rightarrow$ | |
| 2. | | $\leftarrow e$ | $e \in_R [0, 2^t[$ |
| 3. | $y_i := r_i + e x_i$ | $y_1, y_2 \rightarrow$ | $\bar{g} \stackrel{?}{=} g_1^{y_1} g_2^{y_2} h^{-e}$ |

**Korrektheit.** Die Korrektheit ergibt sich aus der folgenden Gleichung:

$$\begin{aligned} g_1^{y_1} g_2^{y_2} h^{-e} &= g_1^{r_1 + e x_1} g_2^{r_2 + e x_2} h^{-e} \\ &= g_1^{r_1} g_2^{r_2} = \bar{g}. \end{aligned}$$

LEMMA 2.5.1. *Es gibt ein pol. Zeit $\tilde{P}$ mit Erfolgswahrscheinlichkeit $2^{-t}$.*

BEWEIS. Der Ablauf ist:

1. $\tilde{P}$ wählt $\tilde{e} \in_R [0, 2^t[$, $r_1, r_2 \in_R \mathbb{Z}_q$ und sendet dann $\bar{g} = g_1^{r_1} g_2^{r_2} h^{-\tilde{e}}$.
2. $B$ sendet $e \in_R [0, 2^t[$.
3. $\tilde{P}$ sendet $y_1 := r_1$, $y_2 := r_2$.

Falls nun $e = \tilde{e}$ gilt ist $\bar{g} = g_1^{y_1} g_2^{y_2} h^{-e}$. □

Offenbar ist $(P,V)_{Ok}$ honest verifier zero-knowledge, da $\tilde{V}$ zur Simulation genauso verteilt sein muss wie $V$.

LEMMA 2.5.2. *Hat $\tilde{P}$ in $(\tilde{P},V)_{Ok}$ für $w_{\tilde{P}}$ bei zwei Paaren $(e,y)$, $(\bar{e},\bar{y}) \in [0, 2^t[ \times \mathbb{Z}_q$ Erfolg, dann gilt*

$$\log_{g_2} g_1 = \frac{\bar{y}_2 - y_2 + x_2(e - \bar{e})}{y_1 - \bar{y}_1 + x_1(\bar{e} - e)}.$$

BEWEIS. Es gilt

$$g_1^{y_1} g_2^{y_2} h^{-e} = \bar{g}_{w_{\tilde{P}}} = g_1^{\bar{y}_1} g_2^{\bar{y}_2} h^{-\bar{e}}.$$

Somit gilt

$$g_1^{y_1 - \bar{y}_1} g_2^{y_2 - \bar{y}_2} = h^{e - \bar{e}} = (g_1^{x_1} g_2^{x_2})^{e - \bar{e}}.$$

Also ist

$$g_1^{y_1 - \bar{y}_1 + x_1(\bar{e} - e)} = g_2^{\bar{y}_2 - y_2 + x_2(e - \bar{e})}$$

und es folgt die Behauptung. □

O.B.d.A. ist $\log_{g_2} g_1$ dem Erzeuger von $g_1, g_2$ nicht bekannt.

**Betrachte den aktiven Angreifer** $A$. $A$ fordert $l$-mal $(P,\tilde{V})_{Ok}$ und erzeugt dann $(\tilde{P},V)_{Ok}$ Signatur mit $\tilde{V}_A,\tilde{P}_A$. Hat $A$ Erfolgswahrscheinlichkeit $\geq 2^{-t+1}$, dann liefert die **Koalition** $[P,A]$ den geheimen $\log_{g_2} g_1$.

PROPOSITION 2.5.3. *Zu $(V,P)_{Ok}$ gibt es einen prob. Alg. AL : $(A,x)\mapsto \log_{g_2} g_1$ mit $E_w|AL| = O(|A|/\varepsilon)$, sofern A Erfolgswahrscheinlichkeit $\varepsilon \geq 2^{-t+1}$ hat – o.B.d.A. enthalte $|A|$ auch $|P|$.*

BEWEIS. AL konstruiert zu einem $w_A$ Erfolge zu zwei Paaren $(e,y)$, $(\overline{e},\overline{y})$ und berechnet $\log_{g_2} g_1$ nach Lemma 2.5.2. Zur Simulation von $A$ wird die Kommunikation $(P,\tilde{V})_{Ok}$ mit Hilfe von $P$ erzeugt. AL entsteht durch eine Koalition von $P$ und $A$. Im Übrigen arbeitet AL wie der Alg. AL von Satz 2.3.8 auf Seite 32. □

COROLLARY 2.5.4. *$(V,P)_{Ok}$ ist sicher gegen aktive Angriffe, gdw. der DL schwer ist.*

Man beachte, dass für aktive Angriffe die Protokolle $(P^k,\tilde{V})$, $(P,\tilde{V})_I$, $(P,\tilde{V})_{Ok}$ auf verschiedene Weise simuliert werden:

- $(P^k,\tilde{V})$ **in Satz 2.2.6 auf Seite 29:** Das Protokoll $(P^k,\tilde{V})$ wird mit $t = O(1)$ durch einen pol. Zeit Simulator perfekt simuliert.
- $(P,\tilde{V})_I$ **in Satz 2.3.8 auf Seite 32:** Das Protokoll $(P,\tilde{V})_I$ wird im ROM simuliert, indem AL geeignete zufällige $H$ selbst wählt.
- $(P,\tilde{V})_{Ok}$ **in Satz 2.5.3:** Das Protokoll $(P,\tilde{V})_{Ok}$ wird mit Hilfe des geheimen Schlüssels $x$ simuliert. Dies geht, weil AL in Satz 2.5.3 $\log_{g_2} g_1$ und nicht $x = \log_g h$ berechnet.

**Fazit:**
In der Praxis wird oft das Schnorr-Schema dem Okamoto-Schema vorgezogen, obwohl es für dieses keinen Beweis der Sicherheit gibt – nicht einmal unter der Annahme, dass das DL-Problem schwer ist. Das liegt daran, dass bis heute keine Schwäche im Schnorr-Schema gefunden wurde und es in der Praxis um einiges schneller ist als das Okamoto-Schema.

## 2.6. Okamoto Signaturen

**Öffentlich:** $g_1,g_2,G = \langle g_1\rangle = \langle g_2\rangle$ und $|G| = q$ mit Hashfunktion

$$H : G\times\{0,1\}^* \to [0,2^t[.$$

**Public-Key:** $h = g_1^{x_1}g_2^{x_2}$.
**Private-Key:** $(x_1,x_2) \in_R \mathbb{Z}_q^2$.
**Signaturerzeugung:** $r_1,r_2 \in_R \mathbb{Z}_q$, $\overline{g} := g_1^{r_1}g_2^{r_2}$, $e := H(\overline{g},m)$, $y_i := r_i + ex_i$ für $i = 1,2$.
**Signatur zu** $m$**:** $(e,y_1,y_2) \in [0,2^t[\times\mathbb{Z}_q^2$.
**Prüfung:** $H(g_1^{y_1}g_2^{y_2}h^{-e},m) \stackrel{?}{=} e$.

In [**Okamoto1992**] wird in Theorem 20 versucht, Sicherheit gegen CMA zu beweisen. Dieses Theorem ist jedoch in sich widersprüchlich und kompliziert.

## 2.7. Brickell-McCurley-Identifikation

[**BrickellMcCurley92**, Seite 29-39]

„DL-Identifikation sicher wie Faktorisierung".

Seien $p,q,w$ Primzahlen mit $p-1=2qw$, $w\neq q$, $g\in\mathbb{Z}_p^*\setminus\{1\}$ und $g^q\equiv 1$.

**Öffentlich:** $p,g,qw$.
**Public-Key:** $h=g^x$.
**Private-Key:** $q,w$ mit $q\cdot w$ schwer zerlegbar, $x$.

In dem Protokoll nach [**BrickellMcCurley92**] wird einfach in $(V,P)$ $q$ durch $qw=(p-1)/2$ ersetzt:

| $(P,V)_{BM}$ | $P$ | | $V$ |
|---|---|---|---|
| 1. | $r\in_R\mathbb{Z}_{qw}$ | $\bar{g}:=g^r\ \rightarrow$ | |
| 2. | | $\leftarrow e$ | $e\in_R[0,2^t[$ |
| 3. | $y:=r+ex \mod qw$ | $y\ \rightarrow$ | $g\stackrel{?}{=}g^y h^{-e}$ |

Wir übertragen Satz 2.1.2 auf Seite 22 von $(P,V)$ auf $(P,V)_{BM}$. $\tilde{P}$ ist Angreifer aus dem Stand.

PROPOSITION 2.7.1. *Es gibt einen prob. Alg. AL* : $(\tilde{P},h)\mapsto\log_g h$ *mit* $E_w|AL|=O(|\tilde{P}|/\varepsilon)$, *sofern* $\tilde{P}$ *Erfolgswahrscheinlichkeit* $\varepsilon\geq 2^{-t+1}$ *hat.*

BEWEIS. (Skizze) AL konstruiert zu einem $w,\bar{g}$ zwei Lösungen $(e,y)$ und $(\tilde{e},\tilde{y})$:

$$g^y h^{-e}=\bar{g}=g^{\tilde{y}}h^{-\tilde{e}}\quad\Rightarrow\quad\log_g h=\frac{y-\tilde{y}}{e-\tilde{e}}\mod qw.$$

□

O.B.d.A. gilt $\mathrm{ggT}(e-\tilde{e},qw)=1$, andernfalls erhält AL die Zerlegung $(p-1)/2=q\cdot w$.

Wir beschreiben die Reduktion

$$\text{Zerlegung von } (p-1)/2\leq_{pol}\tilde{P}.$$

Wegen $g^q=1$ und $h\in\langle g\rangle$ gibt es ein eind. bestimmtes $z\in\mathbb{Z}_q$ mit $h=g^z$. Es gilt

$$\begin{aligned} x &= z=\frac{z-\tilde{z}}{e-\tilde{e}}\mod qw\\ \left[\log_g h\right] &= \{z+iq\mid u=0,\ldots,w-1\}.\end{aligned}$$

FACT 2.7.2. *x ist stat. unabh. von* $\frac{y-\tilde{y}}{e-\tilde{e}}\in[\log_g h]$.

BEWEIS. Sei $x=z+iq$. AL hängt nur von $h=g^x$ und nicht von $i$ ab. Es folgt

$$Ws_x[\mathrm{ggT}\left(\frac{y-\tilde{y}}{e-\tilde{e}}-x,qw\right)=q]=1-\frac{1}{w},$$

da nur im Fall $i=0$ ein ggT ungleich $q$ herauskommt. □

PROPOSITION 2.7.3. *Es gibt einen prob. Alg. AL* : $(\tilde{P},qw)\mapsto\{q,w\}$ *mit* $E_w|AL|=O(|\tilde{P}|/\varepsilon)$ *sofern* $\tilde{P}$ *Erfolgsw.* $\varepsilon\geq 2^{-t+1}$ *hat.*

BEWEIS. (Skizze) AL erzeugt $x \in_R \mathbb{Z}_{qw}^*$, setzt $h = g^x$ und berechnet wie in Satz 2.7.1 $(e,y)$, $(\tilde{e},\tilde{y})$ mit $x \equiv \frac{y-\tilde{y}}{e-\tilde{e}} \mod qw$. AL erhält mit Ws $1-\frac{1}{w}$ die Zerlegung gemäß

$$q = \mathrm{ggT}(\frac{y-\tilde{y}}{e-\tilde{e}}, wq).$$

□

**Fazit:**
Somit ist $(P,V)_{BM}$ sicher gegen Angriffe aus dem Stand sofern $(p-1)/2$ schwer zerlegbar ist UND $\log_g h$ schwer berechenbar ist: Sogar wenn $P$ und $V$ zusammenarbeiten können sie höchsten $(p-1)/2$ zerlegen, was aber nichts bringt, da dann immer noch der DL in $\mathbb{Z}_q$ berechnet werden muss.

**Aktive Angreifer** *A.* Ein aktiver Angreifer $A$ fordert $l$-mal $(P,\tilde{V})_{BM}$ und versucht dann $(\tilde{P},V)_{BM}$ mit $\tilde{V}_A, \tilde{P}_A$ zu erzeugen.

PROPOSITION 2.7.4. *Es gibt einen prob. Alg. AL* : $(A,qw) \mapsto \{q,w\}$ *mit* $E_w|AL| = O(|A|/\varepsilon)$ *sofern A Erfolgswahrscheinlichkeit* $\varepsilon \geq 2^{-t+1}$ *hat.*

BEWEIS. Der Beweis geht wie der von Satz 2.7.3: Die Simulation von $(P,\tilde{V})_{BM}$ erfolgt mittels $x,P$. Dies ist möglich, weil AL nicht $x = \log_g h$, sondern $\{q,w\}$ berechnet. □

**Brickell-McCurley Signaturen.**

**Erzeugung:** $r \in_R \mathbb{Z}_{qw}$, $e := H(g^r, m)$, $y := r + xe \mod qw$.
**Signatur:** $(e,y)$
**Prüfen:** $H(g^y h^{-e}, m) = e$.

PROPOSITION 2.7.5. *Sei A ein CMA-Angreifer auf BM-Signaturen mit l Aufrufen des H-Orakels. Im ROM gibt es einen prob. Alg. AL* : $(A,qw) \mapsto \{q,w\}$ *mit* $E_{w,H}|AL| = O(l|A|/\varepsilon)$, *sofern A Erfolgsw.* $\varepsilon \geq 2^{-t+1}$ *hat* .

BEWEIS. (Wie Satz 2.3.8 auf Seite 32 und in Satz 2.3.9 auf Seite 32) AL konstruiert zu einer Orakel-Frage $(g_i, m_i)$ zwei erfolgreiche Tripel $(H,e,y)$, $(\overline{H},\overline{e},\overline{y})$. Die angeforderten Signaturen der CMA-Attacke werden im ROM verteilungsgleich erzeugt, indem $A$ geeignete zufällige $H$-Werte selbst wählt. □

**Fazit:**
Im ROM sind BM-Signaturen sicher gegen CMA, sofern $h \mapsto \log_g h$ schwer ist UND $(p-1)/2$ ist schwer zerlegbar.

## 2.8. Sicherheit gegen Message Recovery

Betrachte NRH-Signaturen (siehe Abschnitt 1.9 auf Seite 19).

DEFINITION 2.8.1. Der CMA-Angreifer **reüssiert**, wenn er die Signatur $(e,y)$ einer Nachricht $m$, die nicht von $H$ abhängt, erzeugt, so dass $H(g^y h^{\overline{e}})$ erfragt wurde und in die Berechnung von $(e,y)$ eingeht. ($H(\overline{g}) \rightsquigarrow (e,y)$ mit $g^y h^{\overline{e}} = \overline{g}$).

PROPOSITION 2.8.2. *Sei A ein CMA-Angreifer auf NRH-Signaturen mit l Aufrufen des H-Orakels. Im ROM gibt es einen prob. Alg. AL* : $(A,h) \mapsto \log_g h$ *mit* $E_{H,w}|AL| = O(l|A|/\varepsilon)$, *sofern A Erfolgsw.*$_{H,w}$ $\varepsilon \geq 2l/q$ *hat.*

BEWEIS. AL befrage das $H$-Orakel über $H(g_1),\ldots,H(g_l)$. Nach Annahme gibt es $1 \le i \le l$, so dass mit $H(g_i)$ Erfolgsw.$_{H,w} \ge 2/q$ hat. Schließe weiter wie bei Satz 2.3.9 auf Seite 32 (Angriff auf Schnorr-Signaturen) und Satz 2.7.3 (Angriff auf BM-Signaturen) mit $q \approx 2^t$. AL konstruiert die Signaturen der gewählten Nachrichten der CMA-Attacke verteilungsgleich im ROM, indem er geeignete zufällige $H$-Werte selbst wählt. AL wählt $e \in_R \mathbb{Z}_q^k, m, y$ und setzt $H(g^y h^{\bar{e}}) := e - m$. Falls das $H$-Orakel über $H(g^y h^{\bar{e}})$ schon befragt war, wird ein neues $e$ gewählt. Künftige Orakelfragen zu $H(g^y h^{\bar{e}})$ sind ausgeschlossen.

AL konstruiert zu einem $w_{\tilde{A}}$, $g_i = (g_1)_{w_{\tilde{A}}}$ zwei erfolgreiche Tripel $(H(g_i), e, y)$, $(\tilde{H}(g_i), \tilde{e}, \tilde{y})$ – eine Orakel replay Attacke. Dabei ist $H(g_1) = \tilde{H}(g_1), \ldots, H(g_{i-1}) = \tilde{H}(g_{i-1})$ Teil von $w_{\tilde{A}}$. Es gilt dann

$$g^y h^e = h_i = g^{\tilde{y}} h^{\tilde{e}}$$

und somit $\log_g h = \frac{y-\tilde{y}}{\tilde{e}-e}$.

Die Konstruktion wird für $i = 1,\ldots,l$ wiederholt. □

## 2.9. Offene Sicherheitsprobleme

- Sicherheit der DL-Identifikation $(P,V)$ gegen aktive Angreifer $A$.
- Sicherheit von signierten ElGamal Ziffertexten gegen CCA:
  **Öffentlich:** $h, g \in G, H$
  **Geheim:** $x = \log_g h$
  **ElGamal-Ziffertext** zu $m \in G$: $(g^r, mh^r)$.
  **Signierter ElGamal Ziffertext** $(g^r, mh^r, e, y)$ mit $e = H(g^y h^{-e}, g^r, mh^r)$. $(e,y)$ ist Schnorr-Signatur zu Nachricht $(g^r, mh^r)$ mit Schlüsselpaar $(r, g^r)$.

Das Protokoll $(P,V)$ zur DL-Identifikation ist ein **proof of knowledge** von $\log_g h$:

PROPOSITION 2.9.1. *Es gibt einen prob. pol. Zeit Alg. AL* : $(\tilde{P}, h) \mapsto \log_g h$ *mit* $E_w|AL| = O(|\tilde{P}|/\varepsilon)$, *sofern* $\varepsilon \ge 2^{-t+1}$.

DEFINITION 2.9.2. Der Algorithmus AL ist ein **knowledge extractor**.

## 2.10. Allgemeine proofs of knowledge

Sei $R \subset \mathbb{Z} \times \mathbb{Z}$ eine pol. Zeit Relation, d.h. $R$ ist pol. Zeit entscheidbar, $\chi_R$ ist pol. Zeit berechenbar.

**$R$-Inversion.**

**Gegeben:** $h \in \mathbb{Z}$.
**Finde:** $x \in \mathbb{Z}$ mit $(h,x) \in R$ oder zeige $\neg\exists x \in \mathbb{Z}\ (h,x) \in R$.

**Beispiele:**

(1) $R = \{(G, Ha) \mid G = (V,E)$ ger. Graph und $Ha \subset G$ Hamiltonzyklus$\}$
(2) $R = \{((A,b),c) \mid (A,b) \in M(\mathbb{Z}), c \in \mathbb{Z}^m$ und $Ac \ge b^{m,n+1}\}$
(3) Ganzzahlige lineare Ungleichungen.

PROPOSITION 2.10.1. *Zu jeder pol. Zeit Relation R gibt es ein 3-Rundenprotokoll* $(P,V)_R$, *welches proof of knowledge ist zu* $(h,x) \in R$.

BEWEIS. Man zeigt es für ein NP-vollständiges Problem. □

Ein Proof of Knowledge für einen Hamiltonschen Kreis kann folgendermassen konstruiert werden:

$P$ will also $V$ beweisen, dass er einen Hamiltonschen Kreis in $G$ kennt, $V$ diesen jedoch nicht verraten:

**Eingabe:** $I \in L_{Ham}$

(1) $P$ wählt $|I|$ zufällige Paare von Kodier- und Dekodierschlüssel.
(2) $P$ permutiert die Knoten von $I$ zufällig.
(3) $P$ kodiert die Einträge der Adjazenzmatrix des permutierten Graphen, jeden Eintrag mit einem eigenen Schlüssel und sendet die Kodierung mit den Kodierschlüsseln an $V$.
(4) $V$ wählt $e \in_R \{0,1\}$.
(5) **if** $e = 0$ **then**
   (a) $P$ sendet $V$ die privaten Schlüssel und die zufälligen Knotenpermutation.
(6) **else**
   (a) $P$ sendet $V$ die privaten Schlüssel zu den Kanten eines Hamiltonschen Kreis im permutierten Graphen.

Ein Simulator $\varphi$ für dieses Protokoll könnte so vorgehen, dass er $e$ errät und dann wie folgt vorgeht:

$e = 0$**:** $\varphi$ wählt eine zufällige Permuation von $I$ mit Kodierschlüsseln.
$e = 1$**:** $\varphi$ kodiert einen beliebigen Hamiltonkreis mit $|I|$-vielen Knoten.

Wir betrachten die ElGamal Verschlüsselung

$$E_k(m,r) = (g^r, mh^r)$$

der Nachricht $m$ zum öffentlichen Schlüssel $h$.

DEFINITION 2.10.2. Die ElGamal Verschlüsselung ist **semantisch sicher**, wenn für jeden prob. pol. Zeit Alg.

$$\begin{aligned} AL : (g,h,m_0,m_1,E_k(m_b,r)) &\mapsto b' \in \{0,1\} \\ Ws_{r,b}[b = b'] &= \frac{1}{2} + \varepsilon \end{aligned}$$

mit vernachlässigbar kleinem $\varepsilon$.

(Kurz: Die Verteilungen $E_k(m_0,r)$, $E_k(m_1,r)$ sind bei gegebenem $g,h,m_0,m_1$ **pol. Zeit unentscheidbar**, Bez.: **puu**)

Desweiteren benötigen wir folgenden Begriff:

DEFINITION 2.10.3. $\varepsilon$ ist **vernachlässigbar klein**, wenn $\varepsilon \leq O(n^{-t})$ für alle $t > 0$ mit $n =$Eingabelänge.

PROPOSITION 2.10.4. *Folgende Aussagen sind äquivalent:*

(1) *ElGamal Verschlüsselung ist semantisch sicher.*
(2) *Die beiden folgenden Verteilungen sind puu:*

$$(g, g^x, g^r, g^{xr}) \quad , \quad (g, g^x, g^r, g^z) \quad \text{mit } r,x,z \in_R \mathbb{Z}_q.$$

DEFINITION 2.10.5. Das Entscheidungsproblem 2) ist das **Decisional Diffie Hellman Problem** (**DDH**).

BEWEIS. (Satz 2.10.4)

$2) \Rightarrow 1)$ : Betrachte folgende Verteilungen bzgl. $r,x,z \in_R \mathbb{Z}_q$, $h = g^x$. Wir zeigen sukzessive, dass die Verteilungen in a), b) und c) puu sind.

a) $(g,h,g^r,h^r)$, $(g,h,g^r,g^z)$ sind puu.
b) Zu gegebenem $m \in G$ sind $(g,h,g^r,mh^r)$, $(g,h,g^r,g^z)$ puu.
c) Zu gegebenem $m_0,m_1 \in G$ sind $(g,h,g^r,m_0h^r)$, $(g,h,g^r,m_1h^r)$ puu.

Es gilt a) nach Voraussetzung 2). Bei gegebenem $m$ kann man $mh^r$ in $h^r$ überführen und umgekehrt $h^r$ in $mh^r$ überführen. Damit sind a) und b) äquivalent. Aus b) folgt c), weil die Relation $\neg$puu transitiv ist. Wären die Verteilungen in c) unentscheidbar, dann sind die Verteilungen in b) entweder für $m = m_0$ oder $m = m_1$ unentscheidbar, im Widerspruch zu b).

$1) \Rightarrow 2)$ : Es genügt c) $\Rightarrow$ b) zu zeigen. Die Verteilungen in c) bleiben puu wenn man $m_1$ durch eine zufällige, unbekannte Nachricht ersetzt. Durch diese Ersetzung geht c) in b) über. $\square$

KAPITEL 3

# Faktorisierungsproblem, Protokolle

Für dieses Kapitel sei auf [**SchnorrKrypto1.3**] verwiesen, wo eine elektronische Form dieses Kapitels heruntergeladen werden kann.

## 3.1. Ong-Schnorr Identifikation

[**OngSchnorr1990**]
Shoup, LNCS 1070 sowie J. Cryptology 1999
Schnorr LNCS 1109, 1294.

Ziel war es die Kommunikation im Fiat Shamir Schema zu verkürzen. Es werden $k$ Runden des FS-Schemas in eine Runde im OS-Schema komprimiert.

**Öffentlich:** $N = p \cdot q$.
**Geheim:** $p, q$.
**Public-Key:** $\underline{v} = (s_1^{2^k}, \ldots, s_t^{2^k}) \in (\mathbb{Z}_N^{*2^k})^t$.
**Private-Key:** $\underline{s} = (s_1, \ldots, s_t) \in (\mathbb{Z}_N^*)^t$.

| $(P,V)_{OS}$ | $P$ | | $V$ |
|---|---|---|---|
| 1. | $r \in_R \mathbb{Z}_q$ | $x := r^{2^k} \rightarrow$ | |
| 2. | | $\leftarrow e$ | $\underline{e} \in_R [0, 2^k[^t$ |
| 3. | $y := r \prod_{i=1}^t s_i^{e_i}$ | $y \rightarrow$ | $y^{2^k} \stackrel{?}{=} x \prod_{i=1}^t v_i^{e_i}$. |

$P$ und $V$ benötigen jeweils maximal $kt + k - 1$ – im Mittel $\frac{t+2}{2}k$ – Multiplikationen/Quadrierungen. Dies gilt auch für Erzeugen/Prüfen der OS-Signaturen. Speziell für $k = 10, t = 8$.

| | max. | im Mittel | |
|---|---|---|---|
| $P$, Sign.erzeugung | 90 | 50 | Mult./Quadr. |
| davon on-line | 80 | 40 | – ” – |
| $V$, Sign.prüfung | 90 | 50 | – ” – |

Sei $\tilde{P}$ Angreifer auf $(P,V)_{OS}$ aus dem Stand. Es gelte $|P|, |V| = O(|\tilde{P}|)$.

PROPOSITION 3.1.1. *Es gibt einen prob. Alg. AL* : $(\tilde{P}, \underline{s}, N) \mapsto (X, Y, l)$ *mit*

- $X, Y \in \mathbb{Z}_N$, $Y^{2^k} = X^{2^{k+l}}$, $0 \leq l < k$,
- $Y = Y(\underline{v})$ *für* $\underline{v} := (s_1^{2^k}, \ldots, s_t^{2^k})$,
- $X(\underline{s})$ *ist linear in einem* $s_\nu$, $1 \leq \nu \leq t$,
- $E_w|AL| = O(|\tilde{P}|/\varepsilon)$, *sofern* $\tilde{P}$ *mit* $\underline{v}$ *Erfolgsw.* $\varepsilon \geq 2^{-kt+1}$ *hat.*

BEWEIS. AL konstruiert wie in Algorithmus 5 in Abschnitt 2.1 zwei 1'en in derselben Zeile $w_{\tilde{P}}$ der Erfolgsmatrix von $\tilde{P}$ (siehe Abbildung 3.1.1).

D.h. also, *AL* konstruiert $\tilde{x} = x(w_{\tilde{P}})$, $(\underline{e}, y), (\underline{e}', y') \in [0, 2^k[^t \times \mathbb{Z}_N$ mit

$$y^{2^k} = \tilde{x} \prod_{i=1}^t v_i^{e_i} \quad , \quad y'^{2^k} = \tilde{x} \prod_{i=1}^t v_i^{e'_i}.$$

Die Behauptungen gelten für

$$\begin{aligned} Y &:= y/y', l := \max\{ j \mid \underline{e} = \underline{e}' \mod 2^j \} \\ X &:= \prod_{i=1}^t s_i^{(e_i - e'_i)/2^l}. \end{aligned}$$

$X$ ist linear in $s_\nu$ mit $2^{l+1} \nmid (e_\nu - e'_\nu)$. □

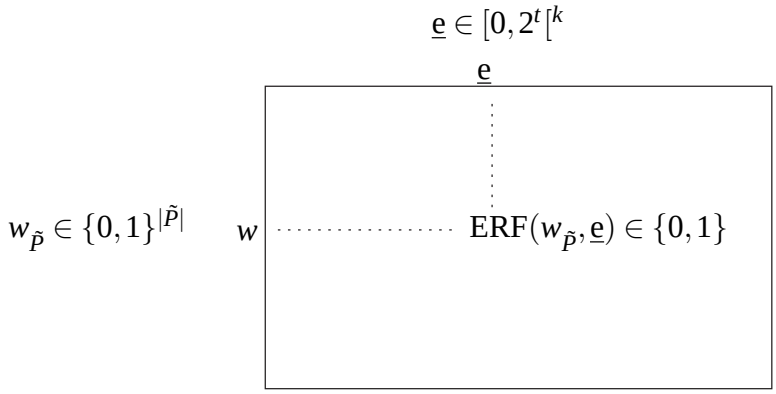



ABBILDUNG 3.1.1. Erfolgsmatrix von $\tilde{P}$

PROPOSITION 3.1.2. *Für $N$ mit $2^k \mid p-1$ gibt es einen prob. Alg. $FA : (\tilde{P}, N) \mapsto \{p,q\}$ mit $E_w|FA| = O(k|\tilde{P}|/\varepsilon)$ sofern $\tilde{P}$ für zufällige $\underline{v} \in_R (\mathbb{Z}_N^{*2^k})^t$ Erfolgsws. $\varepsilon \geq 2^{-kt+1}$ hat.*

BEWEIS. **Faktorisierungsverfahren *FA* für $2^k \mid p-1$**

(1) Wähle $\underline{s} \in_R (\mathbb{Z}_N^*)^t$.
(2) $AL : (\tilde{P}, \underline{s}, N) \mapsto (X, Y, l)$ nach Satz 3.1.1.
(3) $Z := Y^{2^{k-l-1}} / X^{2^{k-1}}$.
Wegen $Y^{2^k} = X^{2^{k+l}}$ gilt $Z^{2^{l+1}} = 1$, $Z \mod p$ nimmt wegen $2^k \mid p-1$ zufällig zwei Werte an – statistisch unabh. von $Z \mod q$. Beachte, dass $X(\underline{s})$ linear in einem $s_\nu$ ist.
(4) Berechne das kleinste $i \leq l+1$ mit $Z^{2^i} = 1$.
Es sind $i$ und $l$ Funktionen in $\underline{v}$ und $w_{\tilde{P}}$. Sie sind bei festem $\underline{v}$ konstant in $\underline{s}$.
   - Fall 1: $i \leq 1$, $Z^2 = 1$. Dann gilt mit $Ws_{\underline{S}} \geq \frac{1}{2}$, dass
   $$\mathrm{ggT}(Z \pm 1, N) = \{p,q\}.$$
   - Fall 2: $i \geq 2$, $Z^{2^{i-1}} \neq -1$. Dann gilt
   $$\mathrm{ggT}(Z^{2^{i-1}} \pm 1, N) = \{p,q\}.$$
   - Fall 3: $i \geq 2$, $Z^{2^{i-1}} = -1$.

Wiederhole die Schritte 1-3 mit unabhängigen $\underline{s} \in_R (\mathbb{Z}_N^*)^t$. Tritt der Fall Wert $i$, $2 \leq i < k$, im Fall 3 doppelt auf, so gilt für die beiden Werte $Z, Z'$:

$$Z^{2^{i-1}} = -1,\ Z'^{2^{i-1}} = -1,\ (ZZ')^{2^{i-1}} = 1.$$

FA führt eine Liste der Paare $(Z,i)$ von Fall 3. In dieser Liste wird jetzt $(Z',i)$ ersetzt durch $(ZZ', i')$ mit dem kleinsten $i' < i$), so dass $(ZZ')^{2^{i'}} = 1$.

Tritt $i'$ bereits mit $(Z'', i')$ in der Liste auf, so wird die Erniedrigung sukzessive fortgesetzt mit $(Z, Z', Z'', i'')$.

FACT 3.1.3. *Jeder Durchlauf von Fall 3 liefert – nach sukzessiver Erniedrigung von $i$ – ein neues $i$ in der Liste der Paare $(Z,i)$.*

Nach $\leq k$ Durchläufen von Fall 3 wird $i \leq 1$ und damit Fall 1 erreicht. Damit wird $N$ nach höchstens $k$ Durchläufen der Schritte 1-3 mit $Ws \geq \frac{1}{2}$ zerlegt. Die Ws. bezieht sich auf die zufälligen $\underline{s}$, welche in das $Z$ mit $Z^2 = 1$ eingehen. □

**Der Fall** $p-1 = 2^m \mod 2^{m+1}$**,** $1 \leq m < k$**.** Der Faktorisierungsalg. muss verändert werden, weil $X^{2^{k-1}} \mod p$ in $\underline{s}$ konstant ist. Wir verbessern hierzu die Aussage von Satz 3.1.1:

LEMMA 3.1.4. *In Satz 3.1.1 gilt*

$$Ws_{w_{AL}}\left[l < \left\lceil \frac{1}{t} \lg 4\varepsilon^{-1} \right\rceil\right] \geq \frac{1}{4},$$

*falls $\tilde{P}$ mit $\underline{v}$ eine Erfolgsw. $\varepsilon$ hat.*

BEWEIS. Für $w := \lceil \frac{1}{t} \lg 4\varepsilon^{-1} \rceil$ ist der Anteil der 1'en in der Erfolgsmatrix von $\tilde{P} \geq 2^{-wt+2}$. Eine Zeile $w_{\tilde{P}}$ mit mehr als $2^{(k-w)t+1}$ 1'en heisse **schwer**.

Mindestens die Hälfte der 1'en liegt in schweren Zeilen. Der Alg. *AL* von Satz 3.1.1 konstruiert mit $Ws \geq \frac{1}{2}$ zwei unabh. 1'en in einer schweren Zeile – $ERF(w_{\tilde{P}}, \underline{e}) = ERF(w_{\tilde{P}}, \underline{e}')$. Weil $\underline{e}, \underline{e}'$ unabhängig in $[0, 2^k[^t$ gezogen werden, gilt

$$Ws_{\underline{e}'}[\underline{e} = \underline{e}' \mod 2^w] \leq \frac{1}{2}.$$

Zu festem $\underline{e}$ gibt es nämlich nur $2^{(k-w)t}$-viele $\underline{e}'$ mit $\underline{e} = \underline{e}' \mod 2^w$.

Mit $Ws \geq \frac{1}{4}$ findet AL somit in einer schweren Zeile $w_{\tilde{P}}$ 1'en in Spalten $\underline{e}, \underline{e}'$ mit $\underline{e} \neq \underline{e}' \mod 2^w$.

**Faktorisierungsverfahren zu** $p-1 = 2^m \mod 2^{m+1}$**,** $1 \leq k < m$

Ang. $\tilde{P}$ hat für zufällige $\underline{v} \in (\mathbb{Z}_N^{*2^k})^t$ eine Erfolgsws. $\varepsilon \geq 2^{-mt+2}$.

Wir ändern den Alg. *AL* von Satz 3.1.1 so ab, dass

$$AL : (\tilde{P}, \underline{v}, N) \mapsto (X, Y, l) \quad \text{mit } 0 \leq l < m.$$

Dies ist nach Lemma 3.1.4 ohne weiteres möglich. Man macht stat. unabh. Versuche $AL(\tilde{P}, \underline{v}, N)$ mit dem ursprünglichen *AL* – im Mittel $\leq 4$ Versuche – bis $0 \leq l < m$ erfüllt ist.

Ändere Schritt 3 des Alg. $FA_m$ zu $m \geq k$ ab zu:

(3) (neu) $Z := Y^{2^{m-l-1}} / X^{2^{m-1}}$.

Dann gilt $Z^{l+1} = 1$, wegen $Y^{2^k} = X^{2^{k+l}}$. $Z \mod p$ nimmt zufällig mit $\underline{s}$ zwei Werte an, denn $X(\underline{s})$ ist linear in einem $s_\nu$ und $s_\nu^{2^{m-1}} \mod p$ nimmt – bei konstantem $s_\nu^{2^m} \mod p$ – zwei Werte an. Außerdem gilt $m-l-1 \geq 0$ und damit ist $Z$ einfach zu berechnen.

Die übrigen Schritte des Faktorisierungsverfahrens und die Analyse sind wie im Fall $2^k \mid p-1$. Der Algorithmus läuft über einen der drei Fälle

$$i \leq 1, Z^{2^{i-1}} \neq -1, Z^{2^{i-1}} = -1.$$

Dabei ist $i$ die kleinste Zahl $\leq l+1$ mit $Z^{2^i} = 1$. Wieder wird $N$ nach höchstens $k$ Durchläufen im Fall $Z^{2^{i-1}} = -1$ mit $Ws \geq \frac{1}{2}$ zerlegt. □

Der obige Faktorisierungsalg. benötigt für kleine $m$ eine große Erfolgsws $\varepsilon \geq 2^{-mt+2}$ von $\tilde{P}$. Für $m = 1$ gibt [**Shoup1999**] einen Faktorisierungsalg. $\widetilde{FA}$ an, für den eine Erfolgsws. $\varepsilon \geq 2^{-kt+1}$ von $\tilde{P}$ ausreicht.

**Aktive Angreifer auf** $(P,V)_{OS}$**.** *A* fordert $l$-mal $(P,\tilde{V}_A)_{OS}$ und erzeugt dann $(\tilde{P}_A,V)_{OS}$.

Die Laufzeit von enthält $|\tilde{P}_A|$, $|\tilde{V}_A|$ und die offene Schrittzahl von *A*. Es sei $N = p \cdot q$ mit $p - 1 = 2^m \mod 2^{m+1}$.

PROPOSITION 3.1.5. *Es gibt einen prob. Alg.* $\overline{FA}\colon (A,N) \mapsto \{p,q\}$ *mit* $E_w|\overline{FA}| = O(l|A|/\varepsilon)$, *sofern A für zufällige* $\underline{v} \in_R (\mathbb{Z}_N^{*2^k})^t$ *Erfolgsw.* $\varepsilon \geq 2^{-tk+1}$ *für* $m \geq k$ *und* $\varepsilon \geq 2^{-mt+2}$ *für* $m < k$ *hat.*

BEWEIS. $\overline{FA}_m$ wählt $\underline{s} \in_R (\mathbb{Z}_N^*)^t$, bildet $\underline{v} := (s_1^{2^k},\ldots,s_t^{2^k})$ und simuliert $(P,\tilde{V}_A)_{OS}$ mittels $\underline{s}$. Dann extrahiert er $\{p,q\}$ mittels $\tilde{P}_A$ – analog zu *FA*. Dabei werden $A,\tilde{P}_A,\tilde{V}_A$ als black box benutzt. Die Koalition $[P,A]$ zerlegt also $N$. □

Nach Satz 3.1.5 ist $N$ sicher gegen aktive Angreifer $A$, es sei denn $N$ ist einfach zu zerlegen.

**Zerlegungsalgorithmus** $\widetilde{FA}$ **für** $m < k$**.** $\widetilde{FA}$ zerlegt mittels $\tilde{P}$ RSA-Moduln $N = p \cdot q$ mit $p - 1 = 2^m \mod 2^{m+1}$, $2^{m+1} \nmid q - 1$. Es bezeichne $RSA_m$ die Menge dieser RSA-Moduln.

REMARK 3.1.6. Beachte:

(1) Jeder RSA-Modul $N$ ist in genau einer Menge $RSA_m$ – gegebenenfalls nach Vertauschung von $p$ und $q$.
(2) $RSA_1$ besteht aus den RSA-Moduln $N = p \cdot q$ mit $p \equiv 3 \mod 4$, $q \equiv 3 \mod 4$. Diese $N$ heissen **Blum-Zahlen**.

LEMMA 3.1.7. *Für die* $N \in RSA_m$ *gilt*

*a)* $\mathbb{Z}_N^{*2^m} = \mathbb{Z}_N^{*2^{m+1}}$,
*b)* $-1 \notin \mathbb{Z}_N^{*2^m}$,
*c)* $x \mapsto x^2$ *permutiert* $Z_N^{*2^m}$.

$\widetilde{FA}$ arbeitet mit einem verfälschten geheimen Schlüssel $\tilde{\underline{s}}$. Für $N \in RSA_m$ wird die Zuordnung $\underline{s} \mapsto \underline{v}$ abgeändert zu

$$\underline{v} := (\tilde{s}_1^{2^{\bar{m}}},\ldots,\tilde{s}_t^{2^{\bar{m}}}) \quad \text{fuer ein } \bar{m},\ m \leq \bar{m} \leq k.$$

Für zufällige $\tilde{\underline{s}} \in (\mathbb{Z}_N^*)^t$ ist $\underline{v}$ nach Lemma 3.1.7 zufällig in $(\mathbb{Z}_N^{*2^m})^t = (\mathbb{Z}_N^{*2^k})^t$ und ist somit korrekt verteilt. Den Wert $\bar{m}$ wählt man so, dass $k + m \geq \bar{m} + l + 1$, diese Ungleichung gilt stets für $m = \bar{m}$.

**Der Zerlegungsalgorithmus $\widetilde{FA}$ für $N \in RSA_m$.**

(1) Wähle $\underline{\tilde{s}} \in_R (\mathbb{Z}_N^*)^t$, $\underline{v} := (\tilde{s}_1^{2^{\bar{m}}}, \ldots, \tilde{s}_t^{2^{\bar{m}}})$.
(2) Wende Alg. *AL* von Satz 3.1.1 an auf $\underline{v}$.
Setze $Y := Y/Y'$, $X := \prod_{i=1}^t \tilde{s}_i^{(e_i - e_i')/2^l}$.
Dann gilt $Y^{2^k} = X^{2^{\bar{m}+l}}$ und $X$ ist linear in einem $\tilde{s}_\nu$.
(3) $Z := Y^{2^{k+\bar{m}-l+m-1}} / X^{2^{m-1}}$.
Es gilt $Z^{2^{\bar{m}+l-m+1}} = 1$. Wegen $k+m \geq \bar{m}+l+1$ ist $Z$ einfach zu berechnen. Ausserdem nimmt $Z(\underline{\tilde{s}}) \mod p$ in Abhängigkeit von $\tilde{s}_\nu$ zwei Werte zufällig an.

Die restlichen Schritte in $\widetilde{FA}$ sind wie bei *FA*. Für die kleinste Zahl $i$ mit $Z^{2^i} = 1$ tritt einer der drei Fälle auf:

(1) $i \leq 1$, $Z^2 = 1$. Dann gilt mit $Ws_{\underline{\tilde{s}}} \geq \frac{1}{2}$
$$\mathrm{ggT}(Z \pm 1, N) = \{p, q\}.$$
(2) $i \geq 2$, $Z^{2^{i-1}} \neq -1$. Dann gilt
$$\mathrm{ggT}(Z^{2^{i-1}} \pm 1, N) = \{p, q\}.$$
(3) $i \geq 2$, $Z^{2^{i-1}} = -1$. Nach Lemma 3.1.7 gilt $i \geq m$. Wiederhole die Schritte 1-3 mit unabhängigen $\underline{\tilde{s}} \in_R (\mathbb{Z}_N^*)^t$. Tritt ein Wert $i$, $2 \leq i \leq m$, doppelt auf, so gilt für die zugehörigen Werte $Z, Z'$:
$$Z^{2^{i-1}} = -1, Z'^{2^{i-1}} = -1, (ZZ')^{2^i} = 1.$$

$\widetilde{FA}$ ersetzt in der Liste der Paare $(Z,i)$ zu Fall 3 das Paar $(Z',i)$ durch $(ZZ', i')$ mit dem kleinsten $i' < i$, so dass $(ZZ')^{2^{i'}} = 1$. Die Ersetzung wird fortgesetzt, wenn es schon ein $(Z'', i')$ gibt, usw.

> **Fakt:** Jeder Durchlauf der Schritte 1-3 liefert – nach sukzessiver Erniedrigung von $i$ – ein neues $i$ in der Liste der Paare $(Z,i)$ von Fall 3.

Nach $\leq m$ Durchläufen von Fall 3 wird $i = 1$ und damit Fall 1 erreicht. Somit wird $N$ nach $\leq m$ Durchläufen der Schritte 1-3 mit $Ws \geq \frac{1}{2}$ zerlegt. Die Ws. bezieht sich auf die zufälligen $\underline{\tilde{s}}$, welche in das $Z$ mit $Z^2 = 1$ eingehen. Somit ist Folgendes gezeigt:

PROPOSITION 3.1.8. *Der Algorithmus $\widetilde{FA}$ zerlegt $N \in RSA_m$ mittels $\tilde{P}$ in mittlerer Laufzeit $O(m|\tilde{P}|/\varepsilon)$, sofern $\tilde{P}$ für zufällige $\underline{v} \in_R (\mathbb{Z}_N^{*2^m})^t$ Erfolgsw. $\varepsilon \geq 2^{-kt+1}$ hat.*

Der Algorithmus $\widetilde{FA}$ lehnt sich an den Beweis von [**Shoup1999**] an. Von dort ist die Idee des verfälschten Schlüssels $\underline{\tilde{s}}$. Shoup betrachtet nur Blum-Zahlen, d.h. den Fall $m = 1$. In diesem Fall ist Alg. $\widetilde{FA}$ besonders einfach.

Wegen $i \leq m = 1$ treten die Fälle 2 und 3 nicht auf. Shoup beweist Satz 3.1.8 für aktive Angreifer anstelle des passiven $\tilde{P}$. Hierzu muss $\widetilde{FA}$ das Protokoll $(P, \tilde{V}_A)_{OS}$ mittels $\underline{\tilde{s}}$ in zero-knowledge simulieren. In dieser Simulation errät $\widetilde{FA}$ von der Frage $\underline{e}$ des Prüfers $\tilde{V}_A$ den Anteil $\underline{e} \mod 2^{k-\bar{m}}$. Dies ist notwendig, weil $\widetilde{FA}$ nur den approximativen Schlüssel $\underline{\tilde{s}}$ an. Wird $\underline{e} \mod 2^{k-\bar{m}}$ falsch geraten, so wird die Runde mit reset wiederholt. Dies führt zu einem Verzögerungsfaktor $2^{(k-\bar{m})t}$.

Für eine schnelle Simulation von $(P,\tilde{V}_A)_{OS}$ muss man $\bar{m}$ möglichst groß wählen, ohne die Bedingung $k+m \geq \bar{m}+l+1$ zu verletzen. Nach Lemma 3.1.4 kann man $l < \frac{1}{t}\lg 4\varepsilon^{-1}$ erreichen. Somit wählt man

$$\bar{m} := k+m-\frac{1}{t}\lg 4\varepsilon^{-1}.$$

Dann gilt $2^{(k-\bar{m})t} \leq 2\varepsilon^{-1}$ und der Verzögerungsfaktor ist hinreichend klein. Damit gilt

PROPOSITION 3.1.9. *Satz 3.1.8 gilt für aktive Angreifer A anstelle des passiven $\tilde{P}$.*

KAPITEL 4

# Sicherheit im Generischen Modell (GM)

[**Nechaev1994, Shoup1997, Schnorr2001**]

Alle Krypto-Algorithmen stützen sich im Wesentlichen auf zwei schwere Probleme: Den diskreten Logarithmus und die Faktorisierung. Zu beweisen, dass ein Algorithmus genauso schwer ist, wie eines dieser Probleme, wenn gewisse Voraussetzungen gegeben sind und der Angreifer keine Vorkenntnisse hat (passiver Angreifer), ist relativ einfach. Kann der Angreifer jedoch aktiv ins Geschehen angreifen, sieht die Sache schon anders aus.

Das Generische Modell geht davon aus, dass ein Algorithmus („generischer Algorithmus") keinen Zugriff auf die Bit-Kodierung der Gruppenelemente hat (Angriffe mit Sieb-Verfahren sind also ausgeschlossen). Diese Annahme ist nicht so abwegig, weil z.B. bei Elliptischen Kurven bis heute nur generische Angriffe bekannt sind. Diese Tatsache macht Hoffnung, dass bewiesene Sicherheit im generischen Modell von großem Nutzen ist.

Im Verlauf dieses Kapitels werden obere Schranken für die Laufzeit von generischen Algorithmen zur Lösung des DL-, DH- und DDH-Problem berechnet. Ausserdem wird die Sicherheit der Schnorr-Identifikation und -Signatur und der ElGamal-Verschlüsselung bewiesen.

## 4.1. Ziele

- Finden einere unteren Komplexitätsschranke im GM für
  DL $(g,h) \mapsto \log_g h$
  DH $(g,g^r,g^x) \mapsto g^{rx}$
  DDH unterscheide die Ws-Verteilungen $(g,g^r,g^x,g^{rx})$, $(g,g^r,g^x,g^z)$ für $r,x,z \in_R \mathbb{Z}_q$.
- Sicherheit der DL-Identifikation $(P,V)$ gegen aktive Angreifer im GM.
- Sicherheit von signierten ElGamal Ziffertexten gegen CCA im GM+ROM.

**Signierte ElGamal Ziffertexte.**

**Öffentlich:** $\langle g \rangle = G$, $H : G \times M \to \mathbb{Z}_q$ zuf. Hash-Funktion.
**Secret-Key:** $x \in_R \mathbb{Z}_q$. Dabei ist $M$ beliebiger Nachrichtenraum.
**Public-Key:** $h = g^x$.
**ElGamal Ziffertext zur Nachricht** $m \in G$**:** $(g^r, mh^r)$
**Signierter ElGamal Ziffertext zu** $m \in G$**:** $(g^r, mh^r, e, y)$ mit
$H(g^y g^{-re}, g^r, mh^r) = e$.

Dabei ist $(e,y)$ Schnorr-Signatur zur Nachricht $(g^r, mh^r)$ mit Schlüsselpaar $(r, g^r)$.

Ein Hauptresultat ist die **Plaintext Awareness** von signierten ElGamal Ziffertexten: Im GM+ROM erfordert die Erzeugung von Ziffertexten $(g^r, mh^r, e, y)$ die Kenntnis der Nachricht $m$ und des geheimen Exponenten $r$. Zu einem beliebigen generischen Alg. $\tilde{P} : (g,h) \mapsto$

$(g^r, mh^r, e, y)$ gibt es einen generischen Extraktor $AL : (\tilde{P}, h) \mapsto (m, r)$. Dieser extrahiert den geheimen Signaturschlüssel $r$ zur Schnorr-Signatur $(e, y)$. Die Besonderheit des generischen Extraktors ist, dass er nur die generischen Schritte von $\tilde{P}$ ausführt.

**Sicherheit gegen CCA.** Ein CCA-Angreifer hat ein $D$-Orakel mit dem er beliebige Ziffertexte – ausgenommen den Ziffertext selbst – entschlüsseln kann. Das $D$-Orakel kann man im GM+ROM durch den generischen Extraktor ersetzen. Damit wird das $D$-Orakel eliminiert ohne die Anzahl der generischen Schritte zu erhöhen. Die Sicherheit von signierten ElGamal Ziffertexten gegen CCA-Angriffe folgt somit im GM+ROM aus der semantischen Sicherheit von ElGamal Ziffertexten.

**Das generische Modell (GM).** Sei $G = \langle g \rangle$ zyklische Gruppe mit Generator $g$ und Primzahlordnung $q$.

- Im GM untersuchen wir kryptographische Angriffe, die für alle Gruppen $G$ mit gegebenem $g, q$ möglich sind.
- Ein generischer Angreifer kann die Kodierung der Gruppenelemente nicht benutzen. Der Zugriff auf die Bits der Kodierung der $h \in G$ ist im GM ausgeschlossen.
- Der generische Angreifer kann nur Gruppenoperationen ausführen und Gruppenelemente auf Gleichheit testen.

Im GM zeigen wir obere Schranken für die Erfolgswahrscheinlichkeit eines generischen Angreifers. Diese Schranken hängen nur von $q$ und der generischen Schrittzahl $t$ des Angreifers ab. Die Erfolgswahrscheinlichkeit bezieht sich auf zufällige Gruppenelemente, wie den zufälligen, öffentlichen Schlüssel $h \in_R G$ und zufällige Elemente $g^r \in_R G$ in gegebenen Signaturen/Ziffertexten.

Dem GM liegt die Beobachtung zu Grunde, dass generische und nicht-generische Angriffe unterschiedliche Anforderungen an die Sicherheitsanalyse stellen. Es gibt nur wenige Beispiele nicht-generischer Angriffe, wie z.B. die Berechnung des diskreten Logarithmus mittels Siebmethoden wie dem quadratischen Sieb, dem Zahlkörpersieb usw. Dies betrifft nur spezielle Gruppen wie $\mathbb{Z}_n^*$. Andererseits gibt es aber auch keine Sicherheitsgarantien gegen nicht-generische Angriffe. Solche Sicherheitsgarantien beruhen auf unbewiesenen Annahmen, wie z.B. die Annahme, dass das DL-Problem schwer ist. Dagegen gestaltet sich die Sicherheitsanalyse zu generischen Angriffen erstaunlich erfolgreich. Die Komplexitätsschranken für generische Angriffe erfordern keine unbewiesenen Komplexitätsannahmen. Darüber hinaus gibt es in der Regel optimale generische Angriffe und es gelingt diese herauszufinden.

Ist ein Protokoll im GM+ROM sicher, dann ist es zusammen mit einer starken Gruppe $G$ und einer starken Hashfunktion $H$ sicher. Im Fall einer erfolgreichen, nicht-generischen Attacke muss daher nur $G$ bzw. $H$ nachgebessert werden, während das Protokoll bleibt.

## 4.2. Definition generischer Algorithmen

**Gruppen / Nichtgruppendaten.** Bei generischen Algorithmen unterscheiden wir Gruppenelemente in $G$ und Nicht-Gruppendaten in $NG$. Operationen auf Nicht-Gruppendaten sind kostenfrei.

Wir definieren sukzessive generische Algorithmen

- ohne Interaktion

- mit Hash / Dekodier / Signatur - Orakel

**Generische Schritte.**

- mexp : $\mathbb{Z}_q^d \times G^d \to G$, $(\underline{a}, (f_1, \ldots, f_d)) \mapsto \prod_{i=1}^d f_i^{a_i}$
- Eingaben in $G$, d.h. Konstanten mexp : $\mathbb{Z}_q^0 \times G^0 \to G$.

Spezialfälle von mexp sind Multiplikation/Division für $d = 2$, $a_1, a_2 \in \{\pm 1\}$.

Die einzige Möglichkeit für generische Algorithmen, Daten in $NG$ aus Elementen in $G$ zu erzeugen, besteht im Test auf Gleichheit $f_i \stackrel{?}{=} f_j$ für gegebene $f_i, f_j \in G$. Der Test auf Gleichheit ist kostenfrei. Durch Vergleiche ergeben sich so Nullen (ungleich) und Einsen (gleich), aus denen eine Binärzahl aus $NG$ erzeugt werden kann. Die Elemente aus $G$ und $NG$ sind aber strikt getrennt. Da das Ergebnis nur aus Elementen aus $G$ besteht, interessiert uns die Anzahl der Operationen mit $NG$-Daten nicht.

DEFINITION 4.2.1. Ein **generischer Algorithmus** ist eine Folge von $t$ generischen Schritten

- $f_1, \ldots, f_t \in G$ (Eingaben) $1 \le t' < t$
- $f_i = \prod_{j=1}^t f_j^{a_j}$, $i = t'+1, \ldots, t$.

Dabei hängt $(a_1, \ldots, a_{i-1}) \in \mathbb{Z}_q^{i-1}$ beliebig von gegebenen $NG$-Daten – der $NG$-Eingabe und den Gleichheiten $f_j = f_k$ mit $1 \le j < k \le i-1$ – ab.

Es bezeichne

$$\mathscr{CO}_{i-1} = \{(j,k) \mid f_j = f_k, 1 \le j < k \le i-1\}$$

die dem $i$-ten Schritt vorausgehende Menge von **Kollisionen.** $\mathscr{CO}_t$ sei die Menge aller Kollisionen. Die Exponenten $a_1, \ldots, a_{i-1}$ des $i$-ten generischen Schritts hängen also beliebig von $\mathscr{CO}_{i-1}$ ab.

Nach [**Shoup1997**] kann ein **generischer Algorithmus** kein Wissen über die spezielle Kodierung der Gruppenelemente ausnutzen – er weiss lediglich, dass jedes Gruppenelement als ein eindeutiger Binär-String kodiert ist.

Die Arbeitsweise des generischen Algorithmus beim Berechnen von $NG$-Daten aus $G$-Daten mittels Kollisionen erläutern wir am Beispiel.

**Die Baby-Step-Giant-Step-Methode.** Wir beschreiben in Algorithmus 7 einen gen. Alg. $GA : G^2 \to \mathbb{Z}_q$, welcher zu gegebenem $q$ $(g,h) \mapsto \log_g h$ in $2\sqrt{q}$ gen. Schritten berechnet.

**Algorithm 7** Generischer Algorithmus zur Baby-Step-Giant-Step-Methode

(1) Berechne $k := \lceil \sqrt{q} \rceil$, $l := \lceil q/k \rceil$, somit gilt $lk - k < q \le lk$.
*** Die Berechnung der $NG$-Daten $k, l$ ist kostenfrei ***
(2) Bilde $L_1 := \{g^i \mid 0 \le i < k\}$ in $k-1$ Mult.
(3) Bilde $L_2 := \{hg^{jk} \mid 0 \le j < l\}$ in $l$ Mult.
*** Offenbar gilt: $L_1 \cap L_2 \neq \emptyset$ ***
(4) Finde eine Kollision durch Testen aller Gleichheiten $g^i = hg^{jk}$.
*** Das ist kostenfrei! ***
(5) Bei Gleichheit gilt $\log_g h = i - jk \mod q - 1$.

Die entsprechende Turingmaschine (TM) zu Algorithmus 7 findet die Kollision $g^i = hg^{jk}$ anders. Sie sortiert die Binärkodierungen der $g^i$ und fügt in die sortierte Liste die Kodierungen der $hg^{jk}$ ein. Dies erfordert höchstens $O(\sqrt{q}\log_2 q)$ Gleichheitstests, viel weniger als im gen. Alg. In der Formulierung des GM nach [**Shoup1997**] wird die Anzahl der Gleichheitstests dadurch reduziert, dass man eine zufällige Kodierung $\sigma : G \to \{0,1\}^*$ zu Grunde legt. Generische Schritte haben dann die Form $(\sigma(f_1),\ldots,\sigma(f_d)) \mapsto \sigma(\prod_{i=1}^d f_i^{a_i})$.

Die Ausgabe des gen. Alg. besteht aus *NG*- und *G*-Daten. Die *NG*-Ausgabe ist eine beliebige Funktion der *NG*-Daten – also der *NG*-Eingabe und $\mathscr{CO}_t$. Die *G*-Ausgabe ist von der Form $(f_{\sigma_1},\ldots,f_{\sigma_d}) \in G^d$ mit beliebigen Funktionen

$$\sigma_i : (NG-\text{Eingabe}, \mathscr{CO}_t) \mapsto \{1,\ldots,t\}$$

für $i = 1,\ldots,d$. Die gen. Schrittzahl $t$ hängt beliebig von den *NG*-Daten ab, d.h. die Entscheidung mit dem $i$-ten Schritt zu stoppen hängt beliebig von der *NG*-Eingabe und $\mathscr{CO}_i$ ab.

Der **Wahrscheinlichkeitsraum** besteht aus den zufälligen Gruppenelementen der Eingabe. Einige Gruppenelemente, wie der öffentliche Schlüssel $h \in_R G$ sind zufällig. Gegebene Ziffertexte und Signaturen enthalten weitere zufällige Elemente in $G$. Die Verteilung der zufälligen Elemente der Eingabe wird in jedem Beispiel eigens beschrieben.

### 4.3. Das DL-Problem

**Gegeben:** $g \in G$, $h \in_R G$, $q \in \mathbb{N}$.
**Berechne:** $x := \log_g h \in \mathbb{Z}_q$.

PROPOSITION 4.3.1. *Sei* $GA : G^2 \to \mathbb{Z}_q$, $(g,h) \mapsto x'$ *ein gen. Alg. mit* $t$ *gen. Schritten,* $h = g^x \in_R G$. *Dann gilt* $Ws_x[x = x'] \leq \frac{1}{q} + \binom{t}{2}/q$.

BEWEIS. *GA* berechne $f_1 = g$, $f_2 = h$, $f_i = \prod_{j=1}^{i-1} f_j^{a_j}$ für $i = 1,\ldots,t$. Die Ausgabe $x' = x'(q,\mathscr{CO}_t)$ ist beliebige Funktion in $q,\mathscr{CO}_t$. Wir betrachten $x = \log_g h \in_R \mathbb{Z}_q$ als formale Variable über $\mathbb{Z}_q$.

FACT 4.3.2. $Lf_i := \log_g f_i$ *für* $i = 1,\ldots,t$ *sind als Funktionen in* $x$ *formale Polynome in* $\mathbb{Z}_q[x]$ *vom Grad* $\leq 1$. *Die Koeffizienten von* $Lf_i$ *hängen beliebig von* $q,\mathscr{CO}_{i-1}$ *ab.*

BEWEIS. (Bew. d. Ind. über $i$) Es gilt $Lf_1 = 1$, $Lf_2 = x \in \mathbb{Z}_q[x]$, $Lf_i = \sum_{j=1}^{i-1} a_j \cdot Lf_j \in \mathbb{Z}_q[x]$.

Dabei hängen $a_1,\ldots,a_{i-1} \in \mathbb{Z}_q$ beliebig von $q,\mathscr{CO}_{i-1}$ ab. □

DEFINITION 4.3.3. $(i,j) \in \mathscr{CO}_t$ ist **triviale Kollision**, wenn $Lf_i \equiv Lf_j$, d.h. die linearen Polynome $Lf_i, Lf_j \in \mathbb{Z}_q[x]$ sind identisch.

REMARK 4.3.4. Triviale Kollisionen geben keine Information über den Wert $x := \log_g h$. Nicht triviale Kollisionen enthüllen diesen Wert. Sei nämlich $Lf_i = a_i + b_i x$, $Lf_j = a_j + b_j x$, dann folgt aus $f_i(x) = f_j(x)$, dass $x = (a_i - a_j)/(b_j - b_i)$. Beachte, dass $b_i \neq b_j$ wegen $Lf_i \not\equiv Lf_j$.

Triviale Kollisionen $(i,j)$ hängen nur von der $NG$-Eingabe ab und gehen nicht in die Berechnung von $x'(q,\mathscr{CO}_t)$ ein. Somit gibt es im minimalen gen. Alg. keine triviale Kollision. Im Folgenden seien triviale Kollisionen ausgeschlossen.

Fortsetzung des Beweis zu Satz 4.3.1: Sei

$$W_{i,j} := Ws_x[Lf_i(x) = Lf_j(x) \mid \mathscr{CO}_{j-1} = \emptyset] \quad \text{und} \quad W_{\neq} := Ws_x[\mathscr{CO}_t \neq \emptyset].$$

Dann gilt $W_{i,j} \leq \frac{1}{q}$ und

$$W_{\neq} = \sum_{1\leq i<j\leq t} W_{i,j} \leq \binom{t}{2}/q, \qquad (1)$$

denn für $Lf_i - Lf_j = a + bx = 0$ gilt $Ws[a+bx=0] \leq \frac{1}{q}$. Weiter gilt

$$Ws_x[x = x' \mid \mathscr{CO}_t = \emptyset] = \frac{1}{q-t},$$

weil $x' = x'(q,\mathscr{CO}_t)$ konstant ist während $x$ im Fall $\mathscr{CO}_t = \emptyset$ zufällig über $q-t$ Werte in $\mathbb{Z}_q$ läuft. Es folgt

$$\begin{aligned} Ws_x[x=x'] &\leq \frac{1}{q-t}(1-W_{\neq}) + W_{\neq} = \frac{1}{q-t} + (1-\frac{1}{q-t})W_{\neq} \\ &\overset{(1)}{\leq} \frac{1}{q-t} + (1-\frac{1}{q-t})\cdot\binom{t}{2}/q = \frac{1}{q-t}(1-\binom{t}{2}/q) + \binom{t}{2}/q \\ &\leq \frac{1}{q} + \binom{t}{2}/q. \end{aligned}$$

Letzteres weil $\frac{1}{q-t}(1-\binom{t}{2}/q) = \frac{1}{q}\frac{(1-\binom{t}{2}/q)}{(1-t/q)} < \frac{1}{q}$. □

Die Detailrechnung im Beweis von Satz 4.3.1 geht auch in die folgenden Sätze ein. Statt sie zu wiederholen, nehmen wir an, dass $x$ im Fall $\mathscr{CO}_t = \emptyset$ gleichverteilt über $\mathbb{Z}_q$ ist. Wir vernachlässigen also, dass $x$ im Fall $\mathscr{CO}_t = \emptyset$ nur über $q-t$ Werte läuft. Der Einfluss dieser Reduktion des Wahrscheinlichkeitsraumes von $x$ auf die Erfolgswahrscheinlichkeit des Angreifers ist nämlich bereits durch die Wahrscheinlichkeit $\binom{t}{2}/q$ für $\mathscr{CO}_t \neq \emptyset$ abgedeckt.

Ausserdem sei der Modul $q$ für alle folgenden gen. Alg. gegeben, ohne dass dies besonders erwähnt wird.

COROLLARY 4.3.5. *Satz 4.3.1 gilt auch für den gen. Alg. mit Münzwurf.*

BEWEIS. Sei $f_i = \prod_{j=1}^{i-1} f_j^{a_j}$ und $(a_1,\ldots,a_{i-1})$ beliebige Funktion in $i$, $\mathscr{CO}_{i-1}$, $w \in_R \{0,1\}^t$. Dann ist $f_i = f_i(x,w)$ Funktion in $x,w$. Zu konstantem $w^* \in \{0,1\}^t$ ist $Ws_x[x=x' \mid w=w^*]$ Funktion in $q,w^*$. Für jede Maximalstelle $w_{\max} \in \{0,1\}^t$ dieser Funktion gilt

$$Ws_{x,w}[x=x'] \leq Ws_x[x=x' \mid w=w_{\max}].$$

Ersetzt man $w \in_R \{0,1\}^t$ durch $w_{\max}$, so wächst die Erfolgsw. monoton. □

### 4.4. DH-Problem

**Gegeben:** $(g, g^r, g^x)$ mit $r, x \in_R \mathbb{Z}_q$.
**Berechne:** $g^{rx}$.

PROPOSITION 4.4.1. *Sei $GA : G^3 \to G$, $(g, g^r, g^x) \mapsto f$ gen. Alg. mit $t$ gen. Schritten. Dann gilt*

$$Ws_{r,x}[f = g^{rx}] \leq \frac{1}{q} + \binom{t}{2}/q.$$

BEWEIS. GA berechne $f_1, \ldots, f_t \in G$, $f_1 = g$, $f_2 = g^r$, $f_3 = g^x$. Nach Voraussetzung gilt

$$f = f_\sigma \text{ mit } \sigma = \sigma(q, \mathscr{CO}_t) \in \{1, \ldots, t\}.$$

Wir betrachten $r, x \in_R \mathbb{Z}_q$ als formale Variablen über $\mathbb{Z}_q$. Dann sind $Lf_i := \log_g f_i \in \mathbb{Z}_q[r,x]$ formale Polynome, $Lf_1 = 1$, $Lf_2 = r$, $Lf_3 = x$, $Lf_i = \sum_{j=1}^{i-1} a_j Lf_j$ und $(a_1, \ldots, a_{i-1}) \in \mathbb{Z}_q^{i-1}$ hängt nur von $i, q, \mathscr{CO}_{i-2}$ ab. Triviale Kollisionen $Lf_i \equiv Lf_j$ seien ausgeschlossen. Dann gilt

$$\begin{aligned} Ws_{r,x}[f_\sigma(r,x) = g^{rx}] &\leq Ws_{r,x}[\mathscr{CO}_t \neq \emptyset] + Ws_{r,x}[Lf_\sigma(r,x) = rx \mid \mathscr{CO}_t = \emptyset] \cdot Ws_{r,x}[\mathscr{CO}_t = \emptyset] \\ &\leq \binom{t}{2}/q + 1/q. \end{aligned}$$

Denn einerseits gilt für $1 \leq i < j \leq t$

$$W_{i,j} := Ws_{r,x}[Lf_i(r,x) = Lf_j(r,x) \mid \mathscr{CO}_{j-1} \neq \emptyset] \leq \frac{1}{q},$$

weil $\{(r,x) \in \mathbb{Z}_q^2 \mid Lf_i(r,x) = Lf_j(r,x)\}$ Hyperebene im $\mathbb{Z}_q^2$ ist. Somit gilt

$$Ws_{r,x}[\mathscr{CO}_t \neq \emptyset] = \sum_{1 \leq i < j \leq t} W_{i,j} \leq \binom{t}{2}/q.$$

Andererseits ist $Lf_\sigma \in \mathbb{Z}_q[r,x]$ ein lineares Polynom. Im Fall $\mathscr{CO}_t = \emptyset$ sind die Koeffizienten von $Lf_\sigma$ konstant. Nach Lemma 4.4.2 gilt $Ws_{r,x}[Lf_\sigma(r,x) = rx] \leq 1/q$ und somit

$$Ws_{r,x}[Lf_\sigma = rx \mid \mathscr{CO}_t = \emptyset] \cdot Ws_{r,x}[\mathscr{CO}_t = \emptyset] \leq \frac{1}{q}.$$

□

LEMMA 4.4.2. [**Schwartz1980**] *Sei $f \in \mathbb{Z}_q[x_1, \ldots, x_u]$, $f \not\equiv 0$ total mit grad $f = d$. Dann gilt $Ws_{x_1,\ldots,x_n}[f(x_1, \ldots, x_n) = 0] \leq \frac{d}{q}$.*

BEWEIS. Induktion über $n$. $n = 1$ ist der Fundamentalsatz der Algebra. □

### 4.5. Das DDH-Problem

**Gegeben:** $(g, g^r, g^x, g^{rxb+z(1-b)}) \in G^4$.
**Bestimme:** $b \in \{0,1\}$.

NOTATION 4.5.1. $g_{rxzb} := g^{rxb+z(1-b)} = \begin{cases} g^{rx} & \text{falls } b = 1 \\ g^z & \text{falls } b = 0 \end{cases}$.

PROPOSITION 4.5.2. *Sei GA* : $G^4 \to \{0,1\}$, $(g, g,^r, g^x, g_{rxzb}) \mapsto b'$ *gen. Alg. mit t gen. Schritten und* $r,x,z \in_R \mathbb{Z}_q$, $b \in_R \{0,1\}$. *Dann gilt*

$$Ws_{r,x,z,b}[b = b'] \leq \frac{1}{2} + \binom{t}{2}/q.$$

BEWEIS. GA berechne $f_1, \ldots, f_t$, mit

$$f_1 = g,\ f_2 = g^r,\ f_3 = g^x,\ f_4 = g_{rxzb}$$

und $f_i = \prod_{j \leq i-1} f_j^{a_j}$ für $i = 5, \ldots, t$.

Dann sind $Lf_i = \log_g f_i \in \mathbb{Z}_q[r,x,z]$ formale Polynome deren Koeffizienten von $b$ abhängen, $Lf_1 = 1$, $Lf_2 = r$, $Lf_3 = x$, $Lf_4 = rxb + z(1-b)$, $Lf_i = \sum_{j=1}^{i-1} a_j \cdot Lf_j$ für $i = 5, \ldots, t$. Damit man triviale Kollisionen $(i,j), Lf_i \equiv Lf_j$ ausschließen kann, dürfen sie nicht von $b$ abhängen. Nun ist $Lf_i - Lf_j$ von der Form

$$c_0 + e_1 r + c_2 x + c_3(rxb + z(1-b)),$$

wobei $c_0, \ldots, c_3 \in \mathbb{Z}_q$ für ein $b \in \{0,1\}$, dann ist $c_3 = 0$ und $Lf_i \equiv Lf_j$ gilt für alle $b \in \{0,1\}$. Damit hängen triviale Kollisionen nicht von $b$ ab und seien ausgeschlossen.

Falls $\mathscr{CO}_t = \emptyset$ ist $b' = b'(q)$ konstant aber $b$ zufällig, somit

$$Ws_{r,x,z,b}[b = b' \mid \mathscr{CO}_t = \emptyset] = \frac{1}{2}. \qquad (1)$$

Ferner gilt für beliebiges konstantes $b \in \{0,1\}$:

$$Ws_{r,x,z}[\mathscr{CO}_t \neq \emptyset] \leq \binom{t}{2}/q. \qquad (2)$$

Nach Lemma 4.4.2 auf der vorherigen Seite gilt nämlich für $1 \leq i < j \leq t$

$$W_{i,j} := Ws_{r,x,z}[Lf_i(r,x,z) = Lf_j(r,x,z) \mid \mathscr{CO}_{j-1} = \emptyset] \leq \frac{1}{q}.$$

Denn $Lf_i - Lf_j \in \mathbb{Z}_q[r,x,z]$ hat max. grad 1. Somit gilt (2). Der Satz folgt dann aus (1) und (2). □

Nach diesem Satz kann ein gen. Alg. mit $t$ gen. Schritten die Ws-Verteilungen $(g, g^r, g^x, g^{rx})$ und $(g, g^r, g^x, g^z)$ für $r,x,z \in_R \mathbb{Z}_q$ höchstens mit Vorteil $\binom{t}{2}/q$ unterscheiden.

## 4.6. Semantische Sicherheit der ElGamal Verschlüsselung

Ein Angreifer, der die verschlüsselte Nachricht erkennen will, hat folgendes Problem:

**Gegeben:** $m_0, m_1, g, g^r, g^x, m_b g^{rx} \in G$ für $r,x \in_R \mathbb{Z}_q$, $b \in_R \{0,1\}$.
**Bestimme:** $b \in \{0,1\}$.

PROPOSITION 4.6.1. *Sei GA* : $G^6 \to \{0,1\}$, $(m_0, m_1, g, g^r, g^x, m_b g^{rx}) \mapsto b'$ *gen. Alg. mit t gen. Schritten und* $r,x \in_R \mathbb{Z}_q$, $b \in_R \{0,1\}$. *Dann gilt*

$$Ws_{r,x,b}[b = b'] \leq \frac{1}{2} + \binom{t}{2}/q.$$

BEWEIS. GA berechne $(f_1,\ldots,f_6)=(m_0,m_1,g,g^r,g^x,m_b g^{rx})$ und $f_7,\ldots,f_t\in G$. Dann sind $Lf_i\equiv\log_g f_i\in\mathbb{Z}_q[r,x]$ formale Polynome, deren Koeffizienten von $b$ abhängen. $Lf_i-Lf_j$ ist von der Form $c_0+c_1r+c_2x+c_3(\log_g m_b+rx)$, wobei $c_0,\ldots,c_3\in\mathbb{Z}_q$ nur von $i,j,q$ und $\mathscr{CO}_{j-1}$ abhängen. Gilt $Lf_i\equiv Lf_j$ für ein $b\in\{0,1\}$, dann ist $c_3=0$ und $Lf_i\equiv Lf_j$ gilt für alle $b\in\{0,1\}$. Triviale Kollisionen hängen damit nicht von $b$ ab und seien im Folgenden ausgeschlossen. Für konstantes $b$ gilt wie in Satz 4.4.1 auf Seite 56 und in Satz 4.5.2, dass $Ws_{r,x}[\mathscr{CO}_t\neq\emptyset]\leq\binom{t}{2}/q$. Im Fall $\mathscr{CO}_t=\emptyset$ hängt $b'$ nicht von $b,r,x$ ab, also $Ws_{r,x}[b=b'\mid\mathscr{CO}_t=\emptyset]=\frac{1}{2}$. Somit gilt der Satz. □

Nach diesem Satz liefert der Ziffertext höchstens einen $\binom{t}{2}/q$-Vorteil beim Erkennen der verschlüsselten Nachricht. Mit dem selben Argument liefert der Ziffertext bei Rückschlüssen auf Eigenschaften der verschlüsselten Nachricht höchstens einen $\binom{t}{2}/q$-Vorteil. **Semantische Sicherheit** bedeutet, dass man den Ziffertext aus der Berechnung beliebiger Prädikate der verschlüsselten Nachricht eliminieren kann und dabei die Erfolgswahrscheinlichkeit nur vernachlässigbar verkleinert. Wir betrachten hier semantische Sicherheit im GM, d.h. bezüglich generischer Berechnungen. Nach Satz 4.6.1 wird die Erfolgswahrscheinlichkeit einer generischen Berechnung eines Prädikats der verschlüsselten Nachricht bei Elimination des Ziffertextes höchstens um $\binom{t}{2}/q$ kleiner. Daher ist die ElGamal-Verschlüsselung im GM semantisch sicher, sofern $q$ so groß ist, dass $\binom{t}{2}/q$ für durchführbare Schrittzahlen vernachlässigbar klein ist. Für heutige Technologien fordert man $q\geq 2^{160}$.

## 4.7. Sicherheit der DL-Identifikation

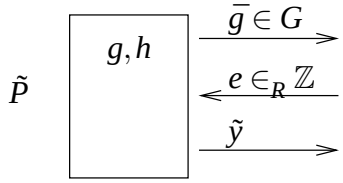
Wir zeigen, dass die DL-Identifikation im generischen Modell sicher gegen aktive Angreifer ist. Dieser Nachweis ist im Turing-Maschinen-Modell noch offen. Wir beweisen die Sicherheit zunächst für Angreifer aus dem Stand (Satz 4.7.1 auf der nächsten Seite) und dann für aktive Angreifer (Satz 4.7.2 auf Seite 60).

Betrachte das Protokoll $(P,V)$ der DL-Identifikation:

| $(P,V)$ | $P$ | | $V$ |
|---|---|---|---|
| 1. | Wählt $r\in_R\mathbb{Z}_q$ und berechnet $\bar{g}:=g^r$ | $\bar{g}\rightarrow$ | |
| 2. | | $\leftarrow e$ | wählt $e\in_R\mathbb{Z}_q$ |
| 3. | Berechnet $y:=r+xe$ | $y\rightarrow$ | Berechnet $\bar{g}\stackrel{?}{=}g^y h^{-e}$ |

Abweichend vom ursprünglichen Protokoll $(P,V)$ wird $e$ zufällig in $\mathbb{Z}_q$ und nicht in $[0,2^t[$ gewählt. Diese Wahl von $e$ dient nur der Vereinfachung.

Betrachte nun einen generischen Angreifer $\tilde{P}$ aus dem Stand:



Angenommen $\tilde{P}$ berechnet $(g,h) \mapsto \bar{g}$ in $t' < t$ gen. Schritten und $(g,h,e) \mapsto \tilde{y}$ in $t - t'$ gen. Schritten. Die Ausgabe $\tilde{y} = \tilde{y}(q, \mathscr{CO}_t, e)$ hängt nur von $q, \mathscr{CO}_t, e$ ab.

PROPOSITION 4.7.1. *Jeder generische $\tilde{P}: (g,h) \mapsto \bar{g} \overset{e}{\mapsto} \tilde{y}$ mit t gen. Schritten hat höchstens Erfolgsw.$_{x,e}$ $\frac{2}{q} + \binom{t}{2}/q$ bzgl. $x, e \in_R \mathbb{Z}_q$.*

BEWEIS. $\tilde{P}$ berechne $f_1, \ldots, f_t \in G$, $f_1 = g$, $f_2 = h$, $f_i = \prod_{j \le i-1} f_j^{a_j}$ für $i = 3, \ldots, t$. Nach Erhalt von $e \in \mathbb{Z}_q$ hängen die Schritte beliebig von $e$ ab. Dann sind $Lf_i \le \log_g f_i \in \mathbb{Z}_q[x]$ formale Polynome, deren Koeffizienten beliebig von $w, \mathscr{CO}_{i-1}$ – und nach Erhalt von $e$ auch von $e$ – abhängen. Triviale Kollisionen $f_i \equiv f_j$, die für konstantes $e$ trivial sind, kann man zu gegebenem $e$ eliminieren.[1] Die Verifikation $g^r \overset{?}{=} g^y h^{-e}$ in $(P,V)$ liefert eine solche triviale Kollision. Nach Elimination der trivialen Kollisionen gilt

$$Ws_c[\mathscr{CO}_t \neq \emptyset] \le \binom{t}{2}/q. \qquad (1)$$

Sei $\bar{g} = f_i$ von der Form $f_i = h^a g^b$, $Lf_i = ax + b \in \mathbb{Z}_q[x]$. Offenbar reüssiert $\tilde{P}$ gdw. $\tilde{y} = Lf_i(x) + ex$. Dabei hängen $a, b, \tilde{y}$ nur von $q, \mathscr{CO}_t$ und $\tilde{y}$ zusätzlich von $e$ ab. Wir zeigen

$$Ws_{x,e}[\tilde{y}(e) = Lf_i(x) + ex \mid \mathscr{CO}_t = \emptyset] \le \frac{2}{q}. \qquad (2)$$

Im Fall $\mathscr{CO}_t = \emptyset$ hängt $\tilde{y}$ nur von $q, e$ während $a, b$ nicht von $e$ abhängen. $\tilde{P}$ reüssiert im Fall $a = -e$ mit $\tilde{y} = b$ für $e \neq -a$ und beliebigem $\tilde{y}$ mit Ws $\frac{1}{q}$. Wegen $Ws_e[e = -a] = \frac{1}{q}$ gilt somit (2). Der Satz folgt aus (1) und (2). □

Wir betrachten nun einen aktiven Angreifer $A$, der $l$-mal $(P, \tilde{V}_A)$ anfordert und dann versucht $(\tilde{P}_A, V)$ selbst zu erzeugen:

| $l$-mal$(P, \tilde{V}_A)$ | $P$ | | $\tilde{V}_A$ |
|---|---|---|---|
| 1. | Wähle $\underline{r} = (r_1, \ldots, r_l) \in_R \mathbb{Z}_q^l$ | $g^{r_1}, \ldots, g^{r_l} \rightarrow$ | |
| 2. | | $\leftarrow \underline{\tilde{e}} = (\tilde{e}_1, \ldots, \tilde{e}_l)$ | wähle $\tilde{e}_1, \ldots, \tilde{e}_l \in_R \mathbb{Z}_q$ |
| 3. | Berechnet $y_i := r_i + \tilde{e}_i x$ | $\underline{y} := (y_1, \ldots, y_l) \rightarrow$ | $g^{r_i} \overset{?}{=} g^{y_i} h^{-\tilde{e}_i}$ |

Die $l$ Durchläufe von $(P, \tilde{V}_A)$ können beliebig parallel oder sequentiell durchgeführt werden – in $t' < t$ gen. Schritten.

| $(\tilde{P}_A, V)$ | $\tilde{P}_A$ | | $V$ |
|---|---|---|---|
| 1. | | $\bar{g} = \bar{g}(\underline{\tilde{e}}, \underline{y}, \mathscr{CO}_{t'}) \rightarrow$ | |
| 2. | | $\leftarrow e \in_R \mathbb{Z}_q$ | |
| 3. | | $\tilde{y}(\underline{\tilde{e}}, \underline{y}, \mathscr{CO}_{t'}) \rightarrow$ | Berechnet $\bar{g} \overset{?}{=} g^{\tilde{y}} h^{-e}$ |

$A$ hat Erfolg gdw. $\tilde{y} = \log_g \bar{g} + e \cdot x$.

Für den generischen, aktiven Angreifer wird jeder Aufruf von $P$ als generischer Schritt gezählt. Jeder solche Aufruf liefert ein zufälliges Element $g^{r_i} \in_R G$, welches einem zufälligen Gruppenelement der Eingabe entspricht.

[1] Bem.: Eliminiert man in Abhängigkeit von $e$ triviale Kollisionen, dann hängt die Schrittzahl $t$ von $e$ ab. Im Allgemeinen ist $t$ beliebige Funktion der NG-Daten, also der NG-Eingabe einschließlich $e$ und der Menge aller Kollisionen.

PROPOSITION 4.7.2. *Jeder aktive, generische Angreifer A mit t gen. Schritten hat*

$$\mathit{Erfolgsw.}_{x,\underline{r},e} \leq \frac{2}{q} + \binom{t}{2}/q.$$

BEWEIS. Sei $\bar{g} = f_i$ von der Form $f_i = g^{a_1}h^{a_2}g^{r_1 a_3}\cdots g^{r_l a_{l+2}}$, dabei hängen $a_1,\ldots,a_{l+2} \in \mathbb{Z}_q$ beliebig von $q,\underline{y},\mathscr{CO}_{i-1}$ aber nicht von $e$ ab. $A$ hat Erfolg gdw. $\tilde{y} = Lf_i(x,\underline{r}) + ex$ dabei hängt $\tilde{y} = \tilde{y}(q,\underline{y},e,\mathscr{CO}_{i-1})$ beliebig von $q,\underline{y},e,\mathscr{CO}_{i-1}$ ab. Wie im Beweis von Satz 4.7.1 zeigt man

$$Ws_{x,\underline{r},e}[\tilde{y} = Lf_i(x,\underline{r}) + ex \mid \mathscr{CO}_t = \emptyset] \leq \frac{2}{q}. \qquad (2)$$

Falls $(a_3,\ldots,a_{l+2}) \neq \underline{0}$ reüssiert $A$ nur mit $Ws \leq 1/q$. Im Fall $(a_3,\ldots,a_{l+2}) = \underline{0}$ reüssiert $A$, wenn $e = -a_2$ und sonst mit $Ws \leq 1/q$.

Für konstante $e,\tilde{y}$ gilt nach Entfernen trivialer Kollisionen offenbar

$$Ws_{x,\underline{r}}[\mathscr{CO}_t \neq \emptyset] \leq \binom{t}{2}/q. \qquad (1)$$

Diese Wahrscheinlichkeit hängt vom Münzwurf $\underline{r}$ von $P$ ab, denn Kollisionen mit $g^{r_1},\ldots,g^{r_l}$ beziehen sich auf $\underline{r}$. Der Satz folgt aus (1) und (2). □

## 4.8. Sicherheit von Schnorr-Signaturen im GM+ROM

Öffentlich seien die Gruppe $G$, der Generator $g$ von $G$, die prime Ordnung $q$ von $G$, der Nachrichtenraum $M$, die eine Hashfunktion $H : G \times M \to \mathbb{Z}_q$, welche zufällig und gleichverteilt aus allen Funktionen $G \times M \to \mathbb{Z}_q$ gezogen ist. $H$ wird durch ein Orakel gegeben, das $H$-Orakel.

Die **Signaturschlüssel** seien $x \in_R \mathbb{Z}_q$ (privat) und $h = g^x \in_R G$ (öffentlich).

DEFINITION. Eine **Schnorr-Signatur** zur Nachricht $m$ ist ein Paar $(e,y) \in \mathbb{Z}_q^2$ mit $H(g^y h^{-e}, m) = e$ (siehe Abschnitt 1.7).

PROPOSITION 4.8.1. *Sei $GA : G^2 \to M \times \mathbb{Z}_q^2$, $(g,h) \mapsto (m,e,y)$ ein gen. Alg. mit t gen. Schritten. Dann gilt*

$$Ws_{x,H}[H(g^y g^{-e}, m) = e] \leq \frac{2}{q} + \binom{t}{2}/q.$$

*Jeder Aufruf des H-Orakels wird als generischer Schritt gezählt. Die Frage $Q \in G \times M$ an das Orakel darf beliebig abhängen von allen Daten in G und NG.*

BEWEIS. GA's generische Schritte seien

$$f_1,\ldots,f_{t'},H(f_{\sigma_i},m_i) \quad i = t'+1,\ldots,t.$$

Die Gruppenschritte und Orakelanfragen sind beliebig vernetzt. Die Entscheidung, ob der nächste gen. Schritt ein Gruppenschritt oder ein Orakelschritt ist, hängt in beliebiger Weise von den $NG$-Daten – der $NG$-Eingabe, den Kollisionen und erfragten $H$-Werten – ab. $f_{\sigma_i}$ ist ein bereits berechnetes Gruppenelement, $\sigma_i, m_i, t', t$ hängen beliebig von den $NG$-Daten ab.

- Fall 1. $(g^y h^{-e}, m) \notin \{(f_{\sigma_i}, m_i) \mid i = t'+1, \ldots, t\}$
  Für beliebiges, konstantes $h$ gilt für zufällige $H$
  $$Ws_H[H(g^y h^{-e}, m) = e] = \frac{1}{q}.$$
- Fall 2. $(g^y h^{-e}, m) = (f_{\sigma_i}, m_i)$.
  Beachte, dass $i$ und $\sigma_i, m_i$ im Fall $\mathscr{CO}_{t'} = \emptyset$ durch $(e,y)$ eindeutig bestimmt sind. GA setzt $e := H(f_{\sigma_i}, m_i)$, denn jeder andere Wert führt in Fall 2 zum Misserfolg. Das formale Polynom $Lf_{\sigma_i} \equiv \log_g f_i \in \mathbb{Z}_q[x]$, $Lf = a + bx$. Dabei hängt das berechnete $y$ im Fall $\mathscr{CO}_{t'} = \emptyset$ nicht von $x$ ab.
  - Fall 2.a $\mathscr{CO}_{t'} = \emptyset$, $a = -e$.
    GA reüssiert mit $y = b$. Für zufälliges $H$ ist $e$ zufällig, also $Ws_H[a = -e] = \frac{1}{q}$.
  - Fall 2.b $\mathscr{CO}_t = \emptyset$, $a \neq -e$.
    GA reüssiert mit $Ws_x \frac{1}{q}$.
  - Fall 2.c $\mathscr{CO}_{t'} \neq \emptyset$.
    Dieser Fall hat $Ws_{x,H} \leq \binom{t}{2}/q$.

Triviale Kollisionen $Lf_i \equiv Lf_j$ seien dabei ausgeschlossen.

Die Koeffizienten der $Lf_i$ hängen beliebig von den $NG$-Daten ab – der $NG$-Eingabe $(e,y)$, den Kollisionen und den erfragten $H$-Werten. Die Elimination von trivialen Kollisionen erfolgt somit in Abhängigkeit von den erfragten $H$-Werten.

GA reüssiert damit höchstens mit $Ws_{x,H} \leq \frac{2}{q} + \binom{t}{2}/q$. Für die Fälle 1 und 2 wird das Maximum der Erfolgswahrscheinlichkeiten genommen, für die Unterfälle Fall 2.a, 2.b und 2.c jedoch die Summe der Wahrscheinlichkeiten. □

Satz 4.8.1 zeigt, dass Schnorr-Signaturen sicher gegen existentielle Fälschungen sind. Wir erweitern den Satz nun auf Chosen Message Attacken (CMA).

Unter einer CMA verstehen wir einen aktiven Angriff auf eine Signatur: Der Angreifer kann zunächst Signaturen zu Nachrichten seiner Wahl anfordern und muss dann eine neue Nachricht signieren. Die Simulation der von dem Signatur-Orakel angeforderten Signaturen geht im ROM problemlos. Man wählt $e \in_R \mathbb{Z}_q$ zufällig und setzt den Wert $H(g^y h^{-e}, m)$ – sofern er noch nicht festgelegt war – zu $e$. Das so erzeugte $H$ ist zufällig verteilt. Die zu $H$ erzeugten Signaturen – bis auf ein Ereignis der Wahrscheinlichkeit $\binom{t}{2}/q$ – sind so verteilt wie vom Signaturorakel erzeugte. Damit gilt Satz 4.8.1 auch für Chosen Message Angriffe.

## 4.9. Signierte ElGamal Verschlüsselung

Das Schlüsselpaar zur Verschlüsselung sei

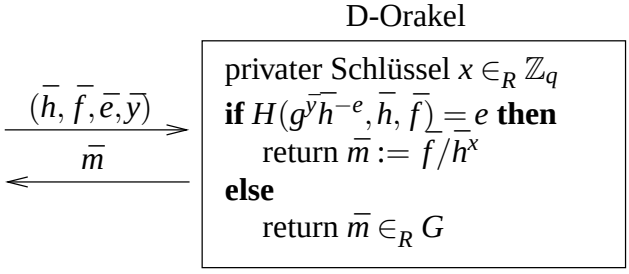
**Privat:** $x \in_R \mathbb{Z}_q$.
**Öffentlich:** $h = g^x \in_R G$.

DEFINITION 4.9.1. Ein **signierter ElGamal Ziffertext** zur Nachricht $m \in G$ ist ein Quadrupel $(g^r, mh^r, e, y) \in G^2 \times \mathbb{Z}_q^2$ mit $H(g^y g^{-re}, g^r, mh^r) = e$.

REMARK 4.9.2. $(g^r, mh^r, e, y)$ besteht aus einem ElGamal Ziffertext $(g^r, mh^r)$ und einer Schnorr Signatur $(e,y)$ zur Nachricht $(g^r, mh^r)$. Zum Signatur-Schlüssel $(r, g^r)$ mit $r$ privat und $g^r$ öffentlich.

Wir betrachten **adaptive chosen ciphertext Attacken** (**CCA**). Ein generischer Angreifer hat zwei Nachrichten $m_0, m_1$ und einen signierten ElGamal Ziffertext $cip_b$ für zufälliges $b \in_R \{0,1\}$ gegeben und soll $b$ erraten. Der Angreifer kann ein Entschlüsselungsorakel beliebig einsetzen, ausgenommen für den Zielziffertext $cip_b$.

**Das Entschlüsselungsorakel, $D$-Orakel.** Auf Eingabe $(\bar{h}, \bar{f}, \bar{e}, \bar{y}) \in G^2 \times \mathbb{Z}_q^2$ prüft das Orakel, ob $H(g^{\bar{y}}, \bar{h}^{-\bar{e}}, \bar{h}, \bar{f}) = \bar{e}$ und sendet einen positiven Fall $\bar{m} := \bar{f}/\bar{h}^x$ und sonst eine Zufallsnachricht $\bar{m} \in_R G$:




Der folgende Satz zeigt, dass signierte ElGamal Ziffertexte sicher gegen adaptive chosen ciphertext Attacken (CCA) sind.

PROPOSITION 4.9.3. *Sei GA* : $G^6 \times \mathbb{Z}_q^2 \to \{0,1\}$, $(m_0, m_1, g, g^r, g^x, m_b g^{rx}, e, y) \mapsto b'$ *gen. Alg. mit H- und D-Orakel und t gen. Schritten. Dann gilt*

$$Ws_{b,r,x,H}[b = b'] \leq \frac{1}{2} + \binom{t}{2}/q.$$

BEWEIS. Die Aufrufe des $H$- und $D$-Orakels werden als gen. Schritt gezählt. GA berechne

(1) $(f_1, f_2, f_3, f_4, f_5, f_6) := (m_0, m_1, g, g^r, g^x, m_b g^{rx})$ mit $f_i = \prod_{j \leq i-1} f_j^{a_j}$ für $i = 7, \ldots, t'$.
(2) $H(f_{\sigma_i}, m_i)$ mit $i = t'+1, \ldots, t'+e'$.
(3) mit $l$ Aufrufen des $D$-Orakels.

Die Schritte vom Typ 1, 2 und 3 sind beliebig verzahnt, $t', l', l$ sind $NG$-Daten, sie hängen beliebig von gegebenen $NG$-Daten – den $NG$-Eingaben, Kollisionen, erfragten $H$-Werten – ab. Die Anzahl der gen. Schritte ist $t = t' + l' + l$.

Induktion über $l$: Für $l = 0$ gilt die Beh. nach Satz 4.6.1 auf Seite 57. Die zusätzlich gegebenen $NG$-Daten – $e, y$ und die erfragten $H$-Werte – helfen nicht.

Induktionsschritt: Betrachte den ersten Aufruf des $D$-Orakels. Beim ersten Aufruf werde das $D$-Orakel über $(\bar{h}, \bar{f}, \bar{e}, \bar{y}) \in G^2 \times \mathbb{Z}_q^2$ befragt.
Wir zeigen, dass $\log_g \bar{h}$ – bis auf ein Ereignis der Wahrscheinlichkeit $2/q$ – nur von gegebenen $NG$-Daten abhängt. Bei Kenntnis von $\log_g \bar{h}$ kann man den $D$-Orakel-Aufruf eliminieren und gemäß $\bar{m} := \bar{f}/h^{\log_g \bar{h}}$ in einem gen. Schritt entschlüsseln.

Die Fälle 1, 2, 2.a, 2.b und 2.c sind wie im Beweis von Satz 4.8.1 auf Seite 60.

- Fall 1: $(g^{\bar{y}}\bar{h}^{-\bar{e}}, (\bar{h}, \bar{f})) \in \{(f_{\sigma_i}, m_i) \mid i = t'+1, \ldots, t'+l'\}$
  Dann gilt für konstante $r, x, b$
  $$Ws_H[H(g^{\bar{y}}\bar{h}^{-\bar{e}}, \bar{h}, \bar{f}) = \bar{e}] = \frac{1}{q}.$$
- Fall 2: $(g^{\bar{y}}\bar{h}^{-\bar{e}}, (\bar{h}, \bar{f})) = (f_{\sigma_{i,m_i}})$.
  GA berechnet Eingabe $\overset{GA}{\mapsto} (f_{\sigma_i}, \bar{h}, \bar{f})$ in höchstens $t'$ Gruppenschritten $f_1, \ldots, f_{t'} \in G$.

Betrachte das formale Polynom $Lf_{\sigma_i} \equiv \log_g f_{\sigma_i} \in \mathbb{Z}_q[r,x]$. $Lf_{\sigma_i}$ ist von der Form

$$Lf_{\sigma_i} = c_0 + c_1 r + c_2 x + c_3(\log_g m_b + rx),$$

dabei hängen $c_0, c_1, c_2, c_3 \in \mathbb{Z}_q$ nur von gegebenen $NG$-Daten – $i, q$, vorangehenden Kollisionen und erfragten $H$-Werten – ab.

Triviale Kollisionen $Lf_i \equiv Lf_j$ hängen nicht von $b$ ab, denn der Koeffizient $c_3$ in $Lf_i - Lf_j$ muss verschwinden. Damit kann man triviale Kollisionen eliminieren, sie seien im Folgenden ausgeschlossen.

Im Fall 2 setzt GA $\bar{e} := H(f_{\sigma_i}, (\bar{h}, \bar{f}))$ und reüssiert gdw. $g^{\bar{y}}\bar{h}^{-\bar{e}} = f_{\sigma_i}$, d.h. gdw. $\bar{y} = Lf_{\sigma_i}(r,x) + \bar{e}L\bar{h}(r,x)$.

Wir unterscheiden den Zielwert $\bar{y}' := Lf_{\sigma_i}(r,x) + \bar{e}L\bar{h}(r,x)$ und den berechneten Wert $\bar{y}$ in $(\bar{h}, \bar{f}, \bar{e}, \bar{y})$.

Sei

$$\begin{aligned} Lf_{\sigma_i} &= a_0 + a_1 r + a_2 x + a_3(\log_g m_b + rx) \\ L\bar{h} &= b_0 + b_1 r + b_2 x + b_3(\log_g m_b + rx) \end{aligned}$$

mit $a_0, \ldots, b_3 \in \mathbb{Z}_q$, die nur von gegebenen $NG$-Daten abhängen, aber nicht von $\bar{e} = H(f_{\sigma_i}, (\bar{h}, \bar{f}))$.

- Fall 2.a: $\mathscr{CO}_{t'} = \emptyset$, $Lf_{\sigma_i} - a_0 = -\bar{e}(L\bar{h} - b_0)$.
  In diesem Fall reüssiert GA mit $\bar{y} = a_0 + \bar{e}b_0$. Weil die Koeffizienten von $L\bar{g}, L\bar{h}$ nicht von $\bar{e} \in_R \mathbb{Z}_q$ abhängen, tritt der Fall $L\bar{g} - a_0 = -\bar{e}(L\bar{h} - b_0)$ höchstens mit $Ws_H = \frac{1}{q}$ auf.
- Fall 2.b: $\mathscr{CO}_{t'} = \emptyset$, $Lf_{\sigma_i} - a_0 \neq -\bar{e}(L\bar{h} - b_0)$.
  In diesem Fall reüssiert GA mit $Ws_x = \frac{1}{q}$.
- Fall 2.c: Der verbleibende Fall $\mathscr{CO}_{t'} \neq \emptyset$ tritt mit $Ws_{r,x,b,H} \leq \binom{t'}{2}/q$ auf.

Zusammenfassend erhält man Lemma 4.9.4 auf der nächsten Seite.

Im Fall $a_1 = \ldots = b_3 = 0$ gilt $L\bar{h} = b_0$, $\bar{h} = g^{b_0}$ und die durch $(\bar{h}, \bar{f}, \bar{e}, \bar{y})$ verschlüsselte Nachricht ist $\bar{m} = \bar{f}/\bar{h}^{b_0}$.

Elimination aller $D$-Orakel Aufrufe. Im Fall $\mathscr{CO}_{t'} = \emptyset$ berechnet GA entweder $(\bar{h}, \bar{f}, \bar{e}, \bar{y})$, so dass $b_0 := \log_g \bar{h}$ nur von gegebenen $NG$-Daten abhängt oder der Ziffertext ist höchstens mit $Ws \leq \frac{2}{q}$ gütlig. Man eliminiert den ersten $D$-Orakel Aufruf, indem man $(\bar{h}, \bar{f}, \bar{e}, \bar{y})$ zu $\bar{m} := \bar{f}/\bar{h}^{b_0}$ entschlüsselt. Dabei werden zwei generische Schritte – der $D$-Orakel Aufruf und die Berechnung von $H(f_{\sigma_i}, (\bar{h}, \bar{f}))$ – durch einen gen. Schritt $\bar{m} := \bar{f}/\bar{h}^{b_0}$ ersetzt. Nach Elimination der $l$ $D$-Orakel Aufrufe hat der reduzierte gen. Algorithmus höchstens $t - l$

gen. Schritte und somit – nach Satz 4.6.1 – höchstens die Erfolgswahrscheinlichkeit $\frac{1}{2} + \binom{t-1}{2}/q$. Damit hat GA höchstens Erfolgswahrsch. $\frac{1}{2} + \binom{t-l}{2}/q + 2l/q$. Es gilt

$$\binom{t-l}{2} + 2l \leq \frac{(t-l+l)(t-1)}{2},$$

sofern $t-1 \geq 4$. Damit gilt der Satz. □

LEMMA 4.9.4. *Entweder gilt $\mathscr{CO}_{t'} \neq \emptyset$ oder es gilt Fall 2 mit $a_1 = a_2 = a_3 = b_1 = b_2 = b_3 = 0$ oder GA reüssiert mit $f_{\sigma_i}$ höchstens mit $Ws \leq \frac{1}{q}$.*

**Plaintext Awareness.** Der Beweis von Satz 4.9.3 zeigt, dass signierte ElGamal Ziffertexte generisch nur zu „gegebenen" Nachrichten erzeugbar sind. Zu jedem gen. Alg. $\tilde{P} : (g,h) \mapsto (\bar{g},\bar{h},\bar{e},\bar{y})$ gibt es einen gen. Extraktor $AL : (\tilde{P},h) \mapsto (\log_g \bar{g}, \bar{m})$. Es wurde gezeigt, dass $\log_g \bar{g}$ nur von in $\tilde{P}$ gegebenen $NG$-Daten abhängt – ausgenommen ein Ereignis der Wahrscheinlichkeit $2/q$. Der Extraktor erhält die verschlüsselte Nachricht $\bar{m} := \bar{h}/g^{\log_g \bar{g}}$ in einem generischen Schritt – im Übrigen führt er die generischen Schritte von $\tilde{P}$ aus.

Unser Nachweis des Extraktors AL benutzt wesentlich das GM. Im Turing-Maschinen Modell gibt es einen anderen Extraktor nach Feige, Fiat, Schamir (siehe Satz 2.1.2 auf Seite 22, sowie Satz 2.3.9 auf Seite 32 in Kapitel 2). Der FFS-Extraktor hat als Verzögerungsfaktor den Kehrwert der Erfolgswahrsch. des Angreifers.

**Sicherheit gegen CCA.** Satz 4.9.3 auf Seite 62 beinhaltet die Sicherheit signierter ElGamal Ziffertexte im GM. Der Beweis benutzt wesentlich den besonderen gen. Extraktor, welcher zu vom gen. Angreifer konstruierten Ziffertexten $(\bar{g},\bar{h},\bar{e},\bar{y})$ den geheimen Signaturschlüssel $\log_g \bar{g}$ extrahiert ohne zusätzliche gen. Schritte zu machen. Dagegen kann man den FFS-Extraktor in polynomial Zeit höchstens logarithmisch oft mit SetBack iterieren. Dies liefert im ROM Sicherheit gegen CCA-Attacken mit logarithmisch vielen Aufrufen des Orakels.

## 4.10. Allgemeine ElGamal Verschlüsselung

Bislang wurde für die ElGamal Verschlüsselung vorausgesetzt dass Nachrichten als Elemente von $G$ kodiert sind. Dies ist für gewisse Gruppen $G$ unpraktisch. Für Untergruppen $G$ von $\mathbb{Z}_N^*$ bzw. von elliptischen Kurven gibt es keine natürliche Kodierung der Nachrichten in $G$. Wir verallgemeinern die ElGamal-Verschlüsselung unter der Annahme, dass der Nachrichtenraum $M$ eine additive Gruppe ist – z.B. $M = \mathbb{Z}_q^n$ – zusammen mit einer Zufallsfunktion $H_n : G \to \mathbb{Z}_q^n$. Es sei $h = g^x \in_R G$ der öffentlichen Schlüssel.

**Verschlüsselung von $m \in \mathbb{Z}_q^n$.** Wähle $r,s \in_R \mathbb{Z}_q$, berechne $g^r$ und $\bar{f} := m \oplus_q H_n(h^r)$, $e := H(g^r,\bar{h},\bar{f})$ und $y := s + er$. Dabei ist $\oplus_q$ die komponentenweise Addition in $\mathbb{Z}_q^n$. Der Ziffertext ist $(g^r,\bar{f},e,y)$.

**Entschlüsselung.** Den Ziffertext $(\bar{g},\bar{f},e,y)$ zu $m := \bar{f} \oplus_q (-H_n(\bar{g}^r))$.

**Korrektheit.** Sei $\bar{g} = g^r$, $\bar{f} = m \oplus_q H_n(h^r)$. Dann gilt $h^r = \bar{g}^x$ und somit $m = \bar{f} \oplus_q (-H_n(g^{-x}))$.

Alle Sicherheitsaussagen für signierte ElGamal Ziffertexte gelten entsprechend auch für die allgemeinen, signierten ElGamal Ziffertexte, sofern $H_n : G \to \mathbb{Z}_q^n$ eine Zufallsfunktion ist. $H_n$ und $H : G \to \mathbb{Z}_q$ müssen nicht stat. unabh. sein. Man kann z.B. $H_n$ wählen als

$$H_n(f) = (H(f,1)H(f,2),\ldots,H(f,n)).$$

## Teil 2

# Kryptographie II

KAPITEL 5

# Identifikation, On-the-fly-Signaturen

[**Girault1991, PoupardStern1998, Schnorr1991**]

On-the-Fly-Signaturen oder auch -Identifikationen müssen in sehr kurzer Zeit durchgeführt werden können. Sie werden z.B. bei der Identifizierung an Maut-Stellen eingesetzt, bei der ein Auto mit 100 km/h eine Maut-Stelle durchfährt und sich in der kurzen Zeit, in der eine Chipkarte Kontakt zu dem Verifizier-Computer hat (z.B. über Funk) zu Abrechnungszwecken identifizieren muss. In der Arbeit von [**PoupardStern1998**] wird hierzu ein auf der Schnorr-Identifikation aufbauendes Verfahren beschrieben, bei dem es – im Gegensatz zu dem in [**Schnorr1991**] beschriebenen Verfahren – nicht mehr nötig ist, den Grad $\lambda(g)$ des Elements $g \in G$ zu kennen.

## 5.1. On-The-Fly-Signaturen

[**PoupardStern1998**]

**Parameter:** $G$ multiplikative Gruppe mit Generator $g, A, B, X, k \in \mathbb{N}$ (nicht $\mathrm{ord}(g)$)
**Secret-Key:** $x \in_R [0, X[$.
**Public-Key:** $h = g^x \in_R G$.

$(P,V)_{Gir}$: Wiederhole Schritte 1-3 $k$-mal.

| $(P,V)_{Gir}$ | $P$ | | $V$ |
|---|---|---|---|
| 1. | Wählt $r \in_R [0,A[$ und berechnet $\bar{g} := g^r$ | $\bar{g} \rightarrow$ | |
| 2. | | $\leftarrow e$ | wählt $e \in_R [0,B[$ |
| 3. | Berechnet $y := r + xe$ | $y \rightarrow$ | Berechnet $\bar{g} \stackrel{?}{=} g^y h^{-e}$, $y \stackrel{?}{\in} [0,A[$ |

DL-Identifikation mit diophantischer Hinterlegung.

**Korrektheit:** Folgt $P$ dem Protokoll, dann akzeptiert $V$ mit $Ws \geq 1 - kBX/A$.

BEWEIS. $g^y = g^{r+ex} = \bar{g}h^e$, $Ws_r[r > A - BX] \leq \frac{BX}{A}$. □

Das Besondere an dem Protokoll ist wie oben bereits gesagt, dass $(P,V)_{Gir}$ nicht die Kentnis von $\mathrm{ord}(g)$ benötigt. Angreifer $\tilde{P}, \tilde{V}$ folgen nicht dem Protokoll.

**5.1.1. Sicherheit von $(P,V)_{Gir}$ gegen $\tilde{V}$.** Erhält $\tilde{V}$ nützliche Information, indem es $P$ befragt?

DEFINITION 5.1.1. Das Protokoll $(P,V)_{Gir}$ ist **$\varepsilon$-zero-knowledge**, wenn es einen prob. pol. Zeit Simulator

$$\varphi : (\tilde{V}, h) \mapsto (g_\varphi, e_\varphi, y_\varphi) \in G \times [0,B[ \times [0,A[$$

gibt, so dass für die ZVen $\varphi(\tilde{V},h)$ und $(\bar{e},\tilde{e},\tilde{y})$ in $(P,\tilde{V})_{Gir}$ gilt

$$\|(\bar{g},\tilde{e},\tilde{y})-(g_\varphi,e_\varphi,y_\varphi)\|_1 \leq \varepsilon.$$

Desweiteren sei definiert:

DEFINITION 5.1.2. Seien $U,V$ Zufallsvariablen über der Menge $S$

$$\|U-V\|_1 =_{def} \sum_{s\in S} |Ws[s=U]-Ws[s=V]|.$$

PROPOSITION 5.1.3. *(Poupard, Stern) Das Protokoll $(P,V)_{Gir}$ ist $\varepsilon$-zero-knowledge mit $\varepsilon = kBX/A$ sofern $B$ polynomial ist.*

BEWEIS. Der Simulator $\varphi$:

(1) **Wiederhole**
  (a) Wähle $y_\varphi \in_R [0,A[$, $e_\varphi \in_R [0,B[$, $g_\varphi := g^{y_\varphi}h^{-e_\varphi}$;
(2) **bis** $e_\varphi = \underbrace{e(\tilde{g},\tilde{V},v_{\tilde{V}},hist)}_{\text{wie in } (P,\tilde{V})_{Gir}}$.

$\varphi$ benötigt im Mittel $B$ Iterationen und ist prob. pol. Zeit, wenn $B$ polynomial ist. □

LEMMA 5.1.4. *(Poupard, Stern) Für $k=1$ gilt*

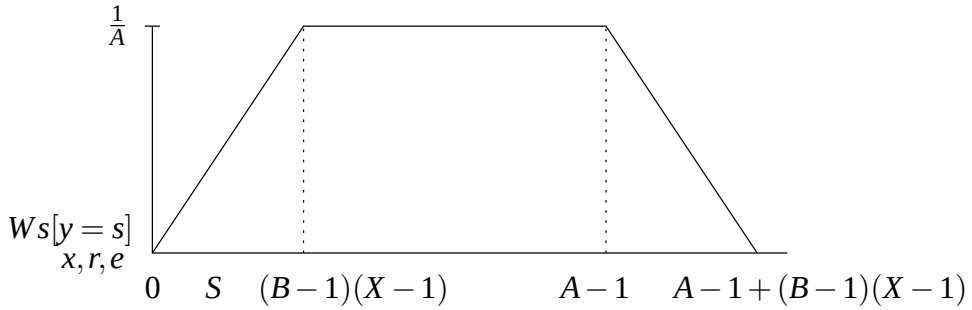
$$\|(\bar{g},\tilde{e},\tilde{y})-(g_\varphi,e_\varphi,y_\varphi)\|_1 \leq BX/A.$$

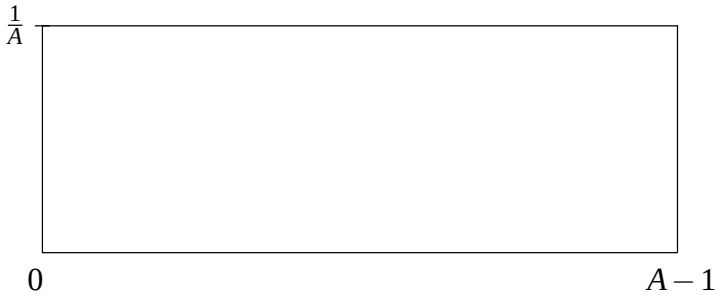
*Damit ist $(P,V)_{Gir}$ stat. zero-knowledge, sofern $kBX/A$ vernachlässigbar klein und $B$ polynomial ist.*

**Fall:** $\tilde{V}=V$ honest verifier. Wir zeigen für $\tilde{V}=V$:

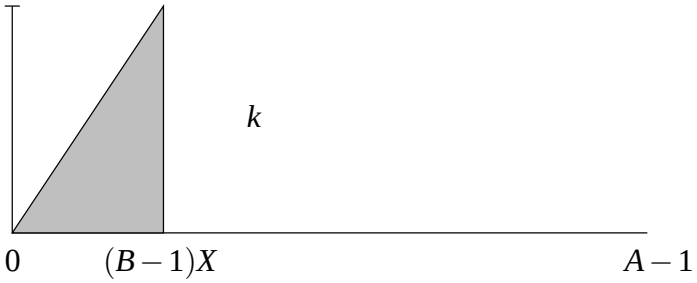
$$\|\tilde{y}-y\|_1 \leq BX/A$$

für $\tilde{y} \in_R [0,A[$, $y=r+xe \in [0,A+(B-1)X[$, $r \in_R [0,A[$, $e \in_R [0,B[$, $x \in_R [0,X[$.






$\|\tilde{y}-y\|_1 \lesssim BS/A$




### 5.1.2. Sicherheit von $(P,V)_{Gir}$ gegen Angreifer $\tilde{P}$ aus dem Stand.

**GPS-Problem nach [GiraultPoupardStern1999].**

**Gegeben:** $g \in G$, $h \in_R g^{[0,X[}$, $A,B,X \in \mathbb{N}$ mit $B \leq A$.
**Finde:** $\sigma \in ]-A,A[$, $\tau \in ]0,B[$, so dass gilt: $g^\sigma = h^\tau$.

**LG-Problem.**

**Gegeben:** $g \in G$, $h \in_R g^{[0,X[}$, $X \in \mathbb{N}$.
**Gesucht:** $x \in ]0,X[$: $h = g^x$.

Die Probleme GPS, LG werden mit wachsendem $X$ schwieriger. Wir reduzieren $X/2$ auf $X$, o.B.d.A. $2 \mid X$.

LEMMA 5.1.5. *Es gilt $LG_{X/2} \leq LG_X$.*

BEWEIS. Wir zeigen, dass das Problem $LG_{X/2}$ mittels $LG_X$ gelöst werden kann:

**Gegeben:** $g \in G$, $h \in_R g^{[0,X/2[}$.
**Wähle:** $b \in_R \{0,1\}$, $h' := g^b h^2 \in_R g^{[0,X]}$.
**Löse:** $h' = g^{x'}$ mit $LG_X$-Alg. $x := \lfloor x'/2 \rfloor$.

□

EXERCISE 5.2. Zeige $GPS_{X/2} \leq GPS_X$.

LEMMA 5.2.1. *Zu gegebenem, primem $ord(g)$ sind $GPS_X, LG_X$ äquivalent.*

BEWEIS. Wir zeigen beide Richtungen:

„$GPS_X \leq_1 LG_X$": Sei $\sigma := 1$, $\tau := x$.

„$LG_X \leq_1 GPS_X$": $x := \tau/\sigma \mod \text{ord}(g)$ wegen $\text{ggT}(\sigma, \text{ord}(g)) = 1$. □

LEMMA 5.2.2. *Für $X \geq 2ord(g)$ geht die Berechnung $g \mapsto ord(g)$ im Mittel in $O(\log_2 X)$ Aufrufen eines $LG_X$-Orakels (bzw. $GPS_X$-Orakels, Übung).*

BEWEIS. Ein Algorithmus könnte wie folgt vorgehen:

(1) **do**
   (a) Wähle stat. unabh. $x'_1, \ldots, x'_j \in_R [0, X[$.
   (b) Setze $h_i := g^{x'_i}$.
   (c) Löse $g^{x_i} = h_i$ mit $LG$-Orakel.
   (d) Dann $\text{ord}(g) \mid (x_i - x'_i)$ mit $Ws_{x'_i}[x_i \neq x'_j] \geq \frac{1}{2}$.
   (e) Setze $\text{ggT}_j := \text{ggT}_{1 \leq i \leq j, x_i \neq x'_i}(x'_i - x_i)$.
(2) **until** $\exists j' < j$: $\text{ggT}_{j'} = \text{ggT}_j$ und $\#\{i \mid j' \leq i \leq j, x_i \neq x'_i\} \geq k$.

CLAIM. $Ws_{\underline{X}'}[\text{ggT}_j = \text{ord}(g)] \geq 1 - 2^{-k}$ und $E_{\underline{X}}[\#\text{Iterationen}] = O(\log_2 \frac{X}{\text{ord}(g)})$.

Betrachte $\tilde{P}$ zu $(P,V)_{Gir}$. Es bezeichne

$$\varepsilon_{\tilde{P}} := Ws_{w,x}[(\tilde{P},V)_{Gir} \text{ akzeptiert mit } h \cdot g^x, w]$$

$w$=Zufallsbits von $(\tilde{P},V)$. Mit $|\tilde{P}|$ wird die Schrittzahl von $\tilde{P}$ bezeichnet, sie sei von der Eingabe unabhängig. □

PROPOSITION 5.2.3. *Es gibt prob. GPS-Lösung $AL : \underset{g,A,B,X}{(\tilde{P},h)} \mapsto (\sigma,\tau)$ mit*

$$E_{w_{AL}}|AL| = O(|\tilde{P}|/\varepsilon_{\tilde{P}}),$$

*sofern $\varepsilon_{\tilde{P}} \geq 2/B$. Siehe Algorithmus 5 auf Seite 23 in Abschnitt 2.1 in Teil 1.*

Dort wird auch die Korrektheit und $E_{w_{AL}}|AL|$ berechnet.

COROLLARY 5.2.4. *Für $X \geq 2 \cdot ord(g)$ gibt es prob. $AL' : (\tilde{P},h) \mapsto (\log_g h, ord(g))$ mit*

$$E_{w_{AL'}}|AL'| = O\left(\log_2(\frac{x}{ord(g)}|\tilde{P}|/\varepsilon_{\tilde{P}})\right)$$

*sofern $\varepsilon_{\tilde{P}} \geq 2/B$.*

BEWEIS. Aus der Lösung zu $GPS_X$ nach Satz 5.2.3 erhält man gemäß Lemma 5.2.2 $\text{ord}(g)$ und gemäß Lemma 5.2.1 $\log_g h$. □

EXERCISE 5.3. $LG_X \leq_2 LG_{\lfloor X/2 \rfloor}$, $GPS_X \leq_2 GPS_{\lfloor X/2 \rfloor}$.

COROLLARY 5.3.1. *Für $X < ord(g)$ und $\varepsilon_{\tilde{P}} \geq 2/B$ geht die Berechnung*

$$(\tilde{P},h) \mapsto (\log_g h, ord(g))$$

*im Mittel in Schrittzahl $O(\frac{ord(g)}{X}|\tilde{P}|/\varepsilon_{\tilde{P}})$.*

BEWEIS. Löse $GPS_X$ nach Satz 5.2.3, $GPS_{X\cdot 2^n}$ mit $GPS_{2^nX} \leq_{2^n} GPS_X$ solange, bis $2^nX \geq 2\cdot \mathrm{ord}(g)$. Dann berechne $\mathrm{ord}(g)$ nach Lemma 5.2.2 und $\log_g h$ nach Lemma 5.2.1. □

**Fazit:**
$\tilde{P}$ habe in $(\tilde{P},V)_{Gir}$ mit $Ws\ \varepsilon_{\tilde{P}} \geq 2/B$ Erfolg. Dann sind für $X < \mathrm{ord}(g)$ $DL_X$, $GPS_X$,, $g \mapsto \mathrm{ord}(g)$ in $O(\frac{\mathrm{ord}(g)}{X}/\varepsilon_{\tilde{P}})$ Aufrufen von $(\tilde{P},V)_{Gir}$ mit stat. unabh. $x \in_R [0,X[$ lösbar.

**RSA-Fall:** Sei $G = \mathbb{Z}_N^*$, $N = p\cdot q$, $p,q$ prim. Sei außerdem

$$\begin{aligned}\mathrm{ord}(g) &= \lambda(N) = \mathrm{kgV}(p-1,q-1)\\ &= \frac{(p-1)(q-1)}{\mathrm{ggT}(p-1,q-1)}\end{aligned}$$

$(\mathbb{Z}_N^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*)$.

## 5.4. Zerlegungsalgorithmus von Miller

**Gegeben:** $L$ mit $\lambda(N) \mid L$.
**Finde:** Zerlegung $N = p\cdot q$.

(1) Zerlege $L = 2^s \cdot r$ mit $r$ ungerade.
(2) Wähle $w \in_R [0,N[$.
(3) Teste $\exists 0 < i \leq s$: $w^{r2^i} = 1 \mod N$ und $w^{r2^{i-1}} \neq \pm 1 \mod N$.
Wenn ja $\mathrm{ggT}(w^{r2^{i-1}} \pm 1, N) = \{p,q\}$.

**Korrektheit.** Für $Y := w^{r2^{i-1}}$ gilt $Y^2 \equiv 1 \mod N$ also $(Y+1)(Y-1) \equiv 0 \mod N$, aber $Y \neq \pm 1\} \mod N$.

**Erfolgswahrscheinlichkeit.** Mit $Ws_w = \frac{1}{2}$ wird $N$ zerlegt.

BEWEIS. Es gibt 4 Quadratwurzeln von $1 \mod N$. Diese treten in $w^{r2^i}$ mit $0 \leq i \leq r$ gleich wahrsch. auf. $\mathbb{Z}_N^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*$, CRT. □

## 5.5. GPS-Signaturen, On-The-Fly-Signaturen

### [PoupardStern1998]

**Parameter:** $G = \langle g\rangle, A,B,X \in \mathbb{N}$ – nicht $|G|$. $H : G\times M \to [0,B[$ starke Hashfunktion $BX/A < 2^{-80}$.
**Privater Schlüssel:** $x \in_R [0,X[$.
**Öffentlicher Schlüssel:** $h = g^x \in_R g^{[0,X[}$.
**Erzeugung einer GPS-Signatur zu** $m \in M$**:** Siehe Algorithmus 8.

**Verifikation der Signatur** $(\bar{g},y)$ **zu** $m \in M$**.** $e = H(g^y h^{-e}, m)$, $y = [0,A[$.

REMARK 5.5.1. Die Bedingung $y \in [0,A[$ normiert die Länge der Signatur und des Authentikationsnachweis. Sie hat keinen sichtbaren Einfluss auf die Sicherheit. Sie wurde benutzt beim Nachweis, dass $(P,V)_{Gir}$ honest-verifier zero-knowledge ist.

**Algorithm 8** Erzeugung einer GPS-Signatur zu $m \in M$

(1) **do**
  (a) Wähle stat. unabh. $r \in_R [0,A[$.
  (b) Setze $\bar{g} := g^r$ und $e := H(\bar{g}, m)$.
  (c) Berechne $y := r + ex$.
(2) **until** $y \leq A$

**Random Oracle Model (ROM).** $H \in_R [0,B[^{G \times M}$ gegeben durch $H$-Orakel.

$H : G \times M \to [0,B[$ ist zufällig, nach der Gleichvert. gezogen aus der Menge aller Funktionen dieses Typs.

FACT 5.5.2. *Folgende Aussagen sind äquivalent*

(1) $H \in_R [0,B[^{G \times M}$.
(2) $(H(g_1,m_1),\ldots,H(g_l,m_l)) \in_R [0,B[^l$ *für beliebige* $l \in \mathbb{N}$ *und verschiedene* $(g_1,m_1),\ldots,(g_l,m_l)$.

**5.5.1. Chosen Message Attacke (CMA) auf GPS-Signaturen.** Der Angreifer $A$ fordert zunächst Signaturen zu Nachrichten seiner Wahl, dann erzeugt er eine neue Signatur. Die eingeforderten Signaturen werden von einem **Signatur-Orakel** geliefert.

DEFINITION 5.5.3. Ein **Signatur-Orakel** $(h,m) \mapsto (e,y)$ liefert eine Signatur zum öffentl. Schlüssel $h$ gemäß Ws-Vert. $e = H(g^r,m)$, $y := r + ex$ mit $r \in_R [0,A[$.

PROPOSITION 5.5.4. *Sei $A$ CMA-Angreifer auf GPS-Signaturen mit $l$ Aufrufen des $H$-Orakels. Im ROM gibt es prob. GPS-Lösung $AL : (A,h) \mapsto (\sigma,\tau)$, so dass $E_{H,w}|AL| = O(l|A|/\varepsilon_A l/q)$ sofern $A$ Erfolgsw. $\varepsilon_A \geq 2/B$ hat.*

LEMMA 5.5.5. *Die Ws.-Verteilung der Signaturen des Signatur-Orakels kann man mit $\| \|_1$-Abstand $\leq BX/A + \binom{l}{2}/q$ in pol. Zeit mit $h$ erzeugen:*

(1) *Wähle $e \in_R [0,B[$, $y \in_R [0,A[$.*
(2) *Setze $H(g^y h^{-e},m) := e$ sofern der Wert $H(g^y h^{-e},m)$ noch nicht festgelegt ist.*

BEWEIS. Kollisionen $g^{y_i} h^{-e_i}$, $m = (\bar{g}_A, m)$ mit $1 \leq i$ treten nur mit $Ws \leq \binom{l}{2}/q$ auf. Der $BX/A$ Beitrag zum $\| \|_1$-Abstand entsteht wie beim honest verifier zero-knowledge Beweis zu $(P,V)_{Gir}$. □

Jetzt der Beweis zu Satz 5.5.4:

BEWEIS. (Analog zu Satz 2.3.8 und Satz 2.3.3 in Abschnitt 2 in Teil 1) $AL$ erzeugt die vom CMA-Angreifer $A$ angeforderten Signaturen, indem er die $H$-Werte geeignet wählt – wie auch in Lemma 5.5.5.

O.B.d.A. gibt es $1 \leq i \leq l$, so dass $A$ beim $i$-ten Aufruf des $H$-Orakels mit $e_i := H(g_i,m_i)$ mit $Ws_{H,w_A} \geq 2/B$ eine Sign. $(e_i,y_i)$ erzeugt. Iteriere $(A,h)$ analog zu $AL(\tilde{P},h)$ von Satz 5.2.3. Konstruiere zwei Lösungen $(e_i,y_i)$, $(\bar{e}_i,\bar{y}_i)$ mit

$$\bar{g}(A,w_A,H) = g^{y_i} h^{-e_i} = g^{\bar{y}_i} h^{-\bar{e}_i}$$

und setze $\sigma := |y_1 - \bar{y}_i|$, $\tau := (\bar{e}_i - e_i) sig(\bar{e}_i - e_i)$.

In Stufe 2 des Verfahrens ersetzt man $H$ durch eine stat. unabh. Hashfunktion $\bar{H}$ und setzt

$$\bar{e} := H(\bar{g},m) \quad , \quad \bar{y} = \bar{y}(A,\bar{g},\bar{e},w_A).$$

□

**Mindestlänge von GPS-Signaturen.** $\mathrm{ord}(g) \geq 2^{160}$, $X \geq 2^{160}$, $B \geq 2^{80}$, $A \geq 2^{320}$. Also $\log_2(y) + \log_2 e \geq 320 + 80 = 400$.

| $(P,V)^H_{Gir}$ | $P$ | | $V$ |
|---|---|---|---|
| 1. | Wählt $r \in_R [0,A[$ und berechnet $c := H(g^r)$ | $c \rightarrow$ | |
| 2. | | $\leftarrow e$ | wählt $e \in_R [0,B[$ |
| 3. | Berechnet $y := r + xe$ | $y \rightarrow$ | Berechnet $c \stackrel{?}{=} H(g^y h^{-e})$ |

**$(P,V)^H_{Gir}$ mit kurzer Kommunikation.**

PROPOSITION 5.5.6. *Für $H \in_R [0,B[^G$ ist $(P,V)^H_{Gir}$ so sicher wie $(P,V)_{Gir}$.*

KAPITEL 6

# Fairer Schlüsseltausch

$A, B$ wollen ihre geheimen Schlüssel $sk_A, sk_B$ austauschen. Hierzu verschlüsseln sie $sk_A, sk_B$ mit dem öffentlichen Schlüssel $pk_{TTP}$ einer $TTP$ – **trusted third party**. Diese Verschlüsselung muss verifizierbar sein, $B$ muss mit dem öffentlichen Schlüssel $pk_A$ überprüfen, dass $E_{pk_{TTP}}(sk_A)$ den zugehörigen geheimen Schlüssel verschlüsselt. Man benutzt die Homomorphie-Eigenschaft der Verschlüsselung $E_{pk_{TTP}}$.

Im Folgenden wird ein Verfahren vorgestellt, bei dem $P$ ein $V$ davon überzeugt, dass er einen privaten Schlüssel gewählt hat, den eine TTP mit nur geringem Aufwand errechnen könnte (Gitterreduktion). Anwendungen finden derartige Verfahren z.B. bei elektronischen Wahlen.

## 6.1. Paillier Kryptoschema

[**Paillier1999**]

$$\begin{aligned} pk &= (N, \mathbb{Z}^*_{N^2}, \alpha) \text{ mit } N = p\cdot q,\ p,q \text{ prim und } \mathrm{ggT}((p-1)(q-1), N) = 1. \\ &\quad \alpha \in \mathbb{Z}^*_{N^2} \text{ mit } \mathrm{ord}(\alpha) = \lambda(N)\cdot N = \lambda(N^2) \\ sk &= \lambda = \lambda(N) = \mathrm{kgV}(p-1, q-1) = \frac{(p-1)(q-1)}{\mathrm{ggT}(p-1, q-1)} \end{aligned}$$

BEWEIS. (Für $\lambda(N^2) = N\lambda(N)$) $\lambda(N^2)$ $\mathbb{Z}^*_{N^2} \cong \mathbb{Z}^*_{p^2} \times \mathbb{Z}^*_{q^2}$ CRT.

$\mathbb{Z}^*_{p^2}, \mathbb{Z}^*_{q^2}$ sind zyklisch, also $\lambda(p^2) = p(p-1)$

$$\begin{aligned} \lambda(N^2) &= \mathrm{kgV}(p(p-1), q(q-1)) \\ &= p\cdot q\cdot \mathrm{kgV}(p-1, q-1) \\ &= N\cdot\lambda(N). \end{aligned}$$

Es gelte $\mathbb{Z}^*_{N^2} \subset \mathbb{Z}_{N^2} \cong [0, N^2[$. □

FACT 6.1.1. *für $w \in \mathbb{Z}^*_{N^2}$ gilt:*

$$\begin{aligned} w^{\lambda} &= 1 \mod N \\ w^{\lambda\cdot N} &= 1 \mod N^2. \end{aligned}$$

$$\begin{aligned} L : [0, N^2[ \supset \mathbb{Z}^{*\lambda}_{N^2} &\to [0, N[ \\ 1 + uN &\mapsto u \end{aligned}$$

**Verschlüsselung von** $m \in \mathbb{Z}_N$**:** Wähle $r \in_R \mathbb{Z}_N^*$, $E_\alpha(m,r) = \alpha^m r^N \mod N^2$.
**Entschlüsselung von** $cip \in \mathbb{Z}_{N^2}^*$**:** $m := L(cip^\lambda)/L(\alpha^\lambda) \mod N$.

FACT 6.1.2. $L(\alpha^\lambda) \in \mathbb{Z}_N^*$.

$$\begin{aligned} &\textit{Ang.} && p \mid L(\alpha^\lambda) \\ &\Rightarrow && \alpha^\lambda = 1 + \underbrace{pa}_{u} N \\ &\Rightarrow && \alpha^{\lambda q} = 1 + u \mod N^2 \\ &\Rightarrow && \textit{ord}(\alpha) \mid \lambda q \quad \textit{Widerspruch zu ord}(\alpha) = \lambda N. \end{aligned}$$

**Homomorphie.**

$$\begin{aligned} E_\alpha : \mathbb{Z}_N \times \mathbb{Z}_N^* &\to \mathbb{Z}_{N^2}^* \\ (m,r) &\mapsto \alpha^m r^N \mod N^2 \\ E_\alpha(m_1,r_1) \cdot E_\alpha(m_2,r_2) &= E_\alpha(m_1+m_2, r_1 \cdot r_2) \end{aligned}$$

**Korrektheit der Dekodierung.** Sei $\alpha^\lambda \equiv 1 + \underbrace{u}_{=L(\alpha^\lambda)} N \mod N^2$ und $\alpha^{\lambda m} \equiv 1 + umN \mod N^2$. Also $L(\alpha^{\lambda m}) \equiv L(\alpha^\lambda) \cdot m \mod N$, weil $r^{N\lambda} \equiv 1 \mod N^2$ folgt:

$$m \equiv L(cip^\lambda)/L(\alpha^\lambda) \mod N.$$

COROLLARY 6.1.3. $E_\alpha : \mathbb{Z}_N \times \mathbb{Z}_N^* \to \mathbb{Z}_{N^2}^*$ *ist Isomorphismus.*

BEWEIS. Sei $E_\alpha(m,r) \equiv \alpha^m r^N \mod N$. Dann $r := \alpha^{-m} E_\alpha(m,r)^{N^{-1}} \mod \lambda$. □

## 6.2. Verifizierbare Verschlüsselung von privaten DL-Schlüsseln

[**PoupardStern2000**]

Schlüsselpaar von $P$: $x$, $h = g^x$.

Paillier Schema mit $(N, \mathbb{Z}_{N^2}^*, \alpha)$.

GPS Schema mit $g, A, B, X$ und $\alpha, A, B, X$.

Verifiziere Verschlüsselung von $x$ $\Gamma :\equiv \alpha^x \mod N^2$.

**Verifikation der Verschlüsselung:** Wiederhole Schritte 1-3 $k$-mal:

| | $P, x \in [0,X[$ | | $V, h = g^x, \Gamma = \alpha^x$ |
|---|---|---|---|
| 1. | Wähle $r \in_R [0,A[$, $\gamma_1 := \alpha^r \mod N^2$ und $\gamma_2 := g^r$ | $\gamma_1, \gamma_2 \to$ | |
| 2. | | $\leftarrow e$ | wählt $e \in_R [0,B[$ |
| 3. | Berechnet $y := r + xe$ | $y \to$ | $\gamma_1 \equiv \alpha^y \Gamma^{-e} \mod N^2$, $\gamma_2 = g^y h^{-e}$, $y \stackrel{?}{\in} [0,A[$ |

**Korrektheit:** $\alpha^r = \alpha^y \Gamma^{-e}$, $g^r = g^y h^{-e}$.

LEMMA 6.2.1. *Es gibt eine GPS-Lösung* $AL : (\tilde{P}, h, \Gamma) \mapsto (\sigma, \tau)$ *zu* $(\alpha, \Gamma)$ *und* $(g,h)$ *mit* $E_{w_{AL}}|AL| = O(|\tilde{P}|/\varepsilon_{\tilde{p}})$ *sofern* $\tilde{P}$ *Erfolgsw.*$_w$ $\varepsilon_{\tilde{p}} \geq 2/B$ *hat.*

BEWEIS. Besteht $\tilde{P}$ das Protokoll mit einem $(\gamma_1,\gamma_2)$ und zwei $(e,y),(\bar{e},\bar{y})$, dann gilt $g^{y-\bar{y}}=h^{e-\bar{e}}$, $\alpha^{y-\bar{y}}=\Gamma^{e-\bar{e}}$. □

$P$ verschlüsselt seinen geheimen Schlüssel einmal in $\mathbb{Z}_q$ und ein anderes Mal Paillier mit dem öffentlichen Schlüssel der TTP. Beides weist $P$ anschliessend zero-knowledge nach, so dass sich $V$ sicher sein kann, dass die TTP $x$ berechnen könnte. Da die TTP lediglich $x \mod N$ und nicht $x \mod q$ berechnen kann, muss sie die Gleichung $\sigma+\gamma\tau\equiv 0 \mod N$ lösen und erhält $x:=\sigma/\tau \mod q$.

Eine praktische Anwendung dieses Verfahren ist z.B. die wiederberechnung gelöschter Schlüssel durch eine TTP nach einem Unglück.

**GPS-Lösung.**

$$\begin{aligned}\varphi &= (y-\bar{y})\mathrm{sig}(e-\bar{e})\\ \tau &= (e-\bar{e})\mathrm{sig}(e-\bar{e}).\end{aligned}$$

Es gilt $\alpha^\sigma=\Gamma^\tau$, $g^\sigma=h^\tau$ mit $\sigma\in]-A,A[$, $\tau\in]0,B[$.

**TTP extrahiert $x$.**

**Fall:**

$$\begin{aligned}\Gamma &= \alpha^x, \quad x\in]0,X[\subset[0,N[\\ x &:= L(\Gamma^\lambda)/L(\alpha^\lambda) \mod N.\end{aligned}$$

Die Rekonstruktion ist korrekt für beliebige

$$\Gamma=E_\alpha(x,r)=\alpha^x r^N \mod N^2.$$

**Fall $\tilde{P}$:** $\exists$ gemeinsame GPS-Lösung $\sigma,\tau$ mit

$$\alpha^\sigma=\Gamma^\tau, g^\sigma=h^\tau, \sigma\in]-A,+A[, \tau\in]0,B[.$$

TTP rekonstruiert $(\sigma,\tau)$ und $x:=\sigma/\tau \mod q$. O.B.d.A. $\mathrm{ggT}(\sigma,\tau)=1$.

**Rekonstruktion von $(\sigma,\tau)$.** Für $\gamma:=L(\Gamma^\lambda)/L(\alpha^\lambda) \mod N$ gilt

$$\alpha^{\gamma\lambda} = \Gamma^\lambda,$$

denn $\alpha^\gamma/\Gamma\in\mathbb{Z}_{N^2}^{*N}$

$$\begin{aligned}\alpha^\sigma=\Gamma^\tau \mod N^2, \alpha^{\gamma\lambda}=\Gamma^\lambda &\Rightarrow \alpha^{\tau\gamma x}=\Gamma^{\tau\lambda}=\alpha^{\sigma\lambda}\\ &\Rightarrow \alpha^\sigma=\Gamma^\tau \mod N^2.\end{aligned}$$

Somit gilt

$$\sigma-\gamma\tau=0 \mod N, \qquad (1)$$

wegen $\mathrm{ord}(\alpha)=\lambda N$. O.B.d.A. gilt $\gamma\in\mathbb{Z}_N^*$, denn andernfalls ist $\mathrm{ggT}(\mathrm{ord}\Gamma,N)$ echter Teiler von $N$.

Die Lösungen $(\sigma,\tau)\in\mathbb{Z}^2$ von (1) bilden ein Gitter $\mathscr{L}\subset\mathbb{Z}^2$ mit Basis $(N,0),(\gamma,1)$ und $\det\mathscr{L}=N$.

Für den kürzesten Vektor $(\sigma,\tau)$ von $\mathscr{L}$ gilt

$$\sigma^2+\tau^2\le\sqrt{\frac{4}{3}}N.$$

Für den zweitkürzesten lin. unabh. Vektor $(\overline{\sigma},\overline{\tau})$ gilt

$$(\overline{\sigma}^2+\overline{\tau}^2)(\sigma^2+\tau^2) \geq (\det\mathscr{L})^2 = N^2.$$

Also $\overline{\sigma}^2+\overline{\tau}^2 \geq \sqrt{\frac{3}{4}}N$.

Im Fall $A^2+B^2 < \sqrt{\frac{3}{4}}N$ ist die gemeinsame GPS-Lösung $(\sigma,\tau)$ der eind. best. kürzeste Vektor von $\mathscr{L}$.

Falls $A,B$ verschiedene Größenordnung haben, skaliert man $(\sigma,\tau)$ zu $(\sigma/A,\tau/B)$. Das skalierte Gitter $\mathscr{L}_s$ hat $\det\mathscr{L}_s = N/(AB)$. Es gilt

$$\begin{aligned}\lambda_1(\mathscr{L}_s)^2 &\leq \sqrt{\frac{4}{3}}\frac{N}{AB}\\ \lambda_2(\mathscr{L}_s)^2 &\geq \sqrt{\frac{3}{4}}\frac{N}{AB}.\end{aligned}$$

Im Fall $2AB < \sqrt{3/4}N$ liefert die gemeinsame GPS-Lösung $(\sigma,\tau)$ den eind. best. kürzesten Vektor $(\sigma/A,\tau/B)$ in $\mathscr{L}_s$ : $(\sigma/A)^2+(\tau/B)^2 \leq 2 < \sqrt{\frac{3}{4}}\frac{N}{AB}$.

PROPOSITION 6.2.2. *Falls $\sqrt{\frac{4}{3}}2AB < N$ erhält TTP aus $\Gamma$ den geheimen Schlüssel $x$ sofern es eine gemeinsame GPS-Lösung $(\sigma,\tau)$ mit $\alpha^\sigma = \Gamma^\tau$, $h^\sigma = g^\tau$, $\sigma \in ]-A,A[$, $\tau \in ]0,B[$ gibt.*

### 6.3. Verifizierbare Verschlüsselung geheimer RSA-Schlüssel

**Geheimer RSA-Schl.:** Zerlegung $p,q$ von $n=pq$ bzw. $x = n-\varphi(n) = p+q+1$.

($p,q$ erhält man durch Lösen von

$$\begin{aligned}p+q &= n-\varphi(n)-1\\ p-q &= ((p+q)^2-4N)^{1/2}\end{aligned}$$

)

Vorgehen:

(1) kurze Nachweise der Kenntnis von $x$.
(2) Verschl. von $x$ mit $p_{k_{TTP}}$.

**Prover $P$ beweist $V$ die Kenntnis von $x$.** $(P,V)$ erzeugen gemeinsam $z_i \in_R \mathbb{Z}_n^*$ mit $i=1,\ldots,k$. Wiederhole Schritte 1-3 $k$-mal:

| | $P, x \in [0,X[$ | | $V, h=g^x, \Gamma \stackrel{?}{=} \alpha^x$ |
|---|---|---|---|
| 1. | Wähle $r \in_R [0,A[$,<br>$\gamma_i := z_i^r \mod n$<br>für $i=1,\ldots,K$ | $\underline{\gamma} := (\gamma_1,\ldots,\gamma_K) \rightarrow$ | |
| 2. | | $\leftarrow e$ | wählt $e \in_R [0,B[$ |
| 3. | Berechnet $y := r+xe$ | $y \rightarrow$ | $z_i^{y-ne} \stackrel{?}{=} \gamma_i$ mit<br>$i=1,\ldots,K$, $y \stackrel{?}{\in} [0,A[$ |

**PSG-Protokoll.**

**Korrektheit.**

$$y-ne \equiv r+xe-ne \mod \varphi(n) \equiv r-e(\underbrace{n-x}_{=\varphi(n)}) \mod \varphi(n) = r \mod \varphi(n)$$

$$z_i^{y-ne} = z_i^{r-e\varphi(n)} = z_i^r = \gamma_i.$$

Der Test $y \in [0,A[$ verhindert die Wahl $y := r+me$.

**Größe der Parameter:** $p,q \le 2\sqrt{n}$, $n-\varphi(n) \le 2{,}5\sqrt{n}$, $2{,}5B\sqrt{n} < A \cdot 2^{-80}$, $B^k \ge 2^{80}$.

**Proof of knowledge für $x$.**

**Angenommen:** $\tilde{P}$ kann zu $\underline{\gamma}$ zwei Fragen $e,\bar{e}$ erfolgreich mit $y,\bar{y}$ beantworten. Dann gilt

$$z_i^{y-ne} = \gamma_i = z_i^{\bar{y}-n\bar{e}} \quad i=1,\ldots,k$$
$$\Rightarrow \quad \mathrm{ord}(z_i) \mid \underbrace{y-\bar{y}+n(\bar{e}-e)}_{\neq 0 \text{ fuer } \bar{e}\neq e \text{ wg. } y,\bar{y}\in[0,A[} \quad .$$

Mit hoher Wahrscheinlichkeit gilt $\lambda(n) \mid \mathrm{kgV}(y-\bar{y}+n(\bar{e}-e))$. Denn für $\lambda(n) = \prod_{i\in I} p_i^{e_i}$ gilt

$$Ws_{z_i}[p_i^{e_i} \nmid \mathrm{ord}(z_i)]) = \frac{1}{p_i}.$$

**RSA-Schlüssel der TTP:** $N = PQ$ mit $P,Q$ prim.
**Paillier-Schema mit $\alpha \in \mathbb{Z}_{N^2}^*$:** $\mathrm{ord}(\alpha) = N\lambda(N)$.
**Verifizierbare Verschlüsselung:** von $x = n-\varphi(n)$.

$$\Gamma = \alpha^x \mod N^2.$$

**Verifikation von $\Gamma \stackrel{?}{=} \alpha^x$.** Wiederhole Schritt 1-3 $k$-mal.

| | $P, x \in [0,X[$ | | $V, h = g^x, \Gamma \stackrel{?}{=} \alpha^x$ |
|---|---|---|---|
| 1. | Wähle $r \in_R [0,A[$, $\gamma_0 := \alpha^x$, $\gamma_i := z_i^r$ für K | $\underline{\gamma} := (\gamma_0,\ldots,\gamma_K) \rightarrow$ | |
| 2. | | $\leftarrow e$ | wählt $e \in_R [0,B[$ |
| 3. | Berechnet $y := r+xe$ | $y \rightarrow$ | $\gamma_0 = \alpha^y\Gamma^{-e}$, $\gamma_i = z_i^{y-en}$ mit $i=1,\ldots,K$, $y \stackrel{?}{\in} [0,A[$ |

GPS-Schema zu $(\alpha,\Gamma)$, PSG-Schema zu$(z_i,\gamma_i)$ für $i=1,\ldots,k$.

LEMMA 6.3.1. *Es gibt eine gemeinsame GPS/PSG-Lösung*[1] *AL* : $(\tilde{P},\Gamma) \mapsto (\sigma,\tau)$ *mit* $E_w|AL| = O(|\tilde{P}|/\varepsilon_{\tilde{P}})$ *sofern* $\tilde{P}$ *Erfolgsw.*$_w$ $\varepsilon_{\tilde{P}} \ge 2/B$ *hat.*

BEWEIS. Ang. $\tilde{P}$ kann zu einem $\underline{\gamma}$ zwei Fragen $e,\bar{e}$ erfolgreich beantworten mit $y,\bar{y}$. Dann gilt

$$\alpha^y\Gamma^{-e} = \gamma_0 = \alpha^{\bar{y}}\Gamma^{-\bar{e}}$$
$$z_i^{y-en} = \gamma_i = z_i^{\bar{z}-\bar{e}n} \quad \text{fuer } i=1,\ldots,k.$$

[1] $\alpha^\sigma = \Gamma^\tau$, $z_i^{\sigma-n\tau} = 1 \mod n$, $\sigma \in ]-A,A[$ und $\tau \in ]0,B[$.

Setze $\sigma := (y-\bar{y})\text{sig}(e-\bar{e})$, $\tau := (e-\bar{e})\text{sig}(e-\bar{e})$. □

PROPOSITION 6.3.2. *Falls* $\sqrt{\frac{4}{3}}2AB < N$ *erhält TTP aus* $\Gamma$ *den geheimen Schlüssel* $x$ *in* $O(\sqrt{B})$ *Schritten sofern es eine gemeinsame GPS/PSG-Lösung gibt.*

BEWEIS. TTP berechnet wie im Beweis zu Satz 6.2.2 auf Seite 80

$$(\sigma', \tau') := (\sigma/d, \tau/d) \quad \text{mit } d = \text{ggT}(\sigma, \tau), d \in [0, B[.$$

Dann berechnet TTP ein $\alpha$ mit $z_i^{d(\sigma'-n\tau')} = 1$ für $i = 1, \ldots, k$.

Geht nach Pollard's Kangeruh-Methode in $O(\sqrt{d}) = O(\sqrt{B})$ Schritten. □

| | |
|---|---|
| $n = pq$ | $p, q$ jeweils mit 512 Bits $n - \varphi(n)$ 513-Bits. |
| $B = 2^{40}$ | damit TTP $d$ in $2^{20}$ Schritten rekonstruieren kann. |
| $k = 2$ | Sicherheitsniveau gegen $\tilde{P}$ ist $b^k = 2^{80}$. |

**Parametergrößen für die Verifikation $\Gamma = \alpha^x$ für $x = n - \varphi(n)$.** $A : A < n, A < (n-\varphi(n))kB$. Damit $y := r + ex < A$.

z.B. $A$ 680 Bits.

$N = PQ$, $N > 2\sqrt{\frac{4}{3}}AB$, $P, Q$ mit je 512 Bits.

## 6.4. Nicht-interaktive Version der Protokolle

Die Zufallsbits von $V$: $\underline{e} = (e_1, \ldots, e_k)$, $\underline{z} := (z_1, \ldots, z_k)$ werden mittels öffentlicher Hashfunktion $H$ erzeugt

$$\begin{aligned} z_j &:= H(N, \alpha, n, \Gamma, j) \\ e_j &:= H(N, \alpha, n, \Gamma, \underline{\gamma}, \underline{z}, j). \end{aligned}$$

Sicherheit der nicht-interaktiven Protokolle zur Verifikation von $\Gamma \stackrel{?}{=} \alpha^x$ im ROM.

Hat $\tilde{P}$ Erfolgsw. $\varepsilon_{\tilde{P}} \leq 2l/B - l$ ist die Anzahl der Aufrufe des $H$-Orakels – dann kann man mit einem black-box $\tilde{P}$ eine gemeinsame GPS-Lösung $(\sigma, \tau)$ bzw. eine GPS/PSG-Lösung $(\sigma, \tau)$ extrahieren. Die Sätze 6.2.2 und 6.3.2 bleiben bestehen.

### 6.4.1. Anwendung Key Recovery.

(1) Es sollen geheime Schlüssel gegen Verlust geschützt werden.
(2) Geheime Schlüssel $x$ einer Gesellschaft werden bei ihrer TTP in der Form $\Gamma = \alpha^x$ hinterlegt.

**6.4.2. Vorteil der Paillier Verschlüsselung.** Bei Hinterlegung von $\Gamma = \alpha^x$ mit nicht-interaktiven Protokollen braucht TTP nicht einzugreifen. TTP erhält $\Gamma$ mit dem nicht-interaktiven Verifikationsprotokoll. TTP prüft die Korrektheit dieses Protokolls.

**6.4.3. Elektronische Wahlen.** Werden die Stimmen mittels Paillier verschlüsselt, kann man die Stimmauszählung lokal mittels der Kodierungen $E_\alpha(st, r)$ machen (Homomorphie). Die lokale Auszählung kann öffentlich erfolgen. Es muss nachgewiesen werden, dass nur korrekte Stimmen $st \in \{0, 1\}$ kodiert sind. Der Wähler liefert einen Beweis, dass $E_\alpha(st, r)$ zu $st \in \{0, 1\}$ gehört.

**6.4.4. Verwandte Probleme.** Seien $\langle g\rangle, \langle \bar{g}\rangle = G$, $|G| = q$ und $h, \bar{h}$ öffentlich.

(1) Beweise in honest verifier zero-knowledge: $\log_g h = \log_{\bar{g}} \bar{h}$.
GPS-Protokoll mit $y := r + ex \mod q$.
(2) Beweise in honest verifier zero-knowledge:

$$\log_g h - \log_{\bar{g}} \bar{h} \in \{0,1\}.$$

## 6.5. CDS-Protokoll

Cramer, Damgard, Schoenmakers, Crypto '94, LNCS 839.

| $(P,V)_{CDS}$ | $P, x, b$ | | $V, h = g^x, \bar{h} = \bar{g}^{x+b}$ |
|---|---|---|---|
| 1. | Wähle<br>$r, y_{1-b}, e_{1-b} \in_R \mathbb{Z}_q$,<br>$\gamma_b := g^r$, $\bar{\gamma}_b := \bar{g}^{\gamma}$,<br>$\gamma_{1-b} := g^{y_{1-b}} h^{-e_{1-b}}$,<br>$\bar{\gamma}_{1-b} := g^{-\gamma_{1-b}} (\bar{h}/\bar{g}^{1-b})^{-e_{1-b}}$ | $\gamma_0, \gamma_1, \bar{\gamma}_0, \bar{\gamma}_1 \rightarrow$ | |
| 2. | | $\leftarrow e$ | wählt $e \in_R \mathbb{Z}_q$ |
| 3. | Berechnet<br>$e_b := e - e_{1-b} \mod q$ und<br>$y_b := r + x e_b \mod q$. | $e_0, y_0, e_1 y_1 \rightarrow$ | $e \stackrel{?}{=} e_0 + e_1 \mod q$,<br>$\gamma_0 \stackrel{?}{=} g^{y_0} h^{-e_0}$,<br>$\bar{\gamma}_0 = \bar{g}^{y_0} \bar{h}^{-e'_0}$,<br>$\gamma_1 = g^{\gamma_1} h^{-e_1}$,<br>$\bar{\gamma}_1 = \bar{g}^{y_1} (\bar{h}/\bar{g})^{-e_1}$ |

Wobei

$$\log_g l = \log_{\bar{g}} \bar{h} \left\{ \begin{array}{lcl} \gamma_0 & \stackrel{?}{=} & g^{y_0} h^{-e_0} \\ \bar{\gamma}_0 & = & \bar{g}^{y_0} \bar{h}^{-e'_0} \end{array} \right. ,$$

und

$$\log_g h = \log_{\bar{g}} (\bar{l}/\bar{g}) \left\{ \begin{array}{lcl} \gamma_1 & = & g^{y_1} h^{-e_1} \\ \bar{\gamma}_1 & = & \bar{g}^{y_1} (\bar{h}/\bar{g})^{-e_1} \end{array} \right. .$$

Das Protokoll zur DL-Identifikation – Proof of Knowledge of $x$ – läuft 3-fach ab:

Für die Generatoren $g, \bar{g}$ und die Fragen $e_0, e_1$. Die Frage $e_b, e_{1-b}$ entsprechen dem aktuellen Bit $b$ und seiner Negation $1-b$.

**Korrektheit:** $P$ teilt $e$ aud in $e = e_0 + e_1 \mod q$. Für das korrekte $b$ sind $r, e_b, y_b$ die Daten der DL-Identifikation – Proof of Knowledge $x = \log_g h$, $x = \log_{\bar{g}} \bar{h}$. $r, e_{1-b}, y_{1-b}$ sind die Daten der honest verifier zero-knowledge Simulation für den Proof of Knowledge $x = \log_g h$, $x = \log_{\bar{g}} (\bar{h}/\bar{g})$.

LEMMA 6.5.1. *Das Protokoll $(P,V)_{CDS}$ ist honest verifier zero-knowledge.*

BEWEIS. Es gibt pol. Zeit Simulator $\varphi$ der zu geheimem $e$ die Daten $(\gamma_0, \gamma_1, \bar{\gamma}_0, \bar{\gamma}_1, e_0, y_0, e_1, y_1)$ mit gleicher Verteilung erzeugt, $\varphi$ arbeitet für $e_b$ und $e_{1-b}$ wie der zero-knowledge Simulator mit Kenntnis der Fragen $e_b, e_{1-b}$. □

PROPOSITION 6.5.2. *Das Protokoll $(P,V)_{CDS}$ ist* ***witness indistinguishable****. Die Verteilungen $(\gamma_0, \ldots, \gamma_i)$ sind für $b = 0$ und $b = 1$ gleich.*

BEWEIS. Der Zweig des Protokolls für das korrekte $b$ wird gemäß DL-Identifikation abgewickelt, der andere Zweig folgt der zero-knowledge Simulation. Beim Übergang $b=0$ nach $b=1$ werden die beiden Zweige vertauscht. Die Verteilung der übertragenen Daten ändert sich dabei nicht. Denn die zero-knowledge Simulation simuliert verteilungsgleich. □

PROPOSITION 6.5.3. *Das Protokoll $(P,V)_{CDS}$ ist ein Proof of Knowledge für $x$ und $b \in \{0,1\}$.*

BEWEIS. Ang. $\tilde{P}$ kann zu $(\gamma_0, \overline{\gamma_0}, \gamma_1, \overline{\gamma_1})$ zwei Fragen $e, \overline{e}$ erfolgreich mit $(e_0, y_0, e_1, y_1)$ und $(\overline{e_0}, \overline{y_0}, \overline{e_1}, \overline{y_1})$ beantworten.

Es gibt entweder $e_0 \neq \overline{e_0}$ oder $e_1 \neq \overline{e_1}$, aber nur eines ist möglich.

**Fall:** $e_0 \neq \overline{e_0}$
Dann $\log_g h = \frac{y_0 - \overline{y_0}}{e_0 - \overline{e_0}}$, $\log_{\overline{g}} \overline{h} = \frac{y_1 - \overline{y_0}}{e_0 - \overline{e_0}} \Rightarrow b = 0$.

**Fall:** $e_1 \neq \overline{e_1}$
Dann $\log_g h = \frac{y_1 - \overline{y_1}}{e_1 - \overline{e_1}}$, $\log_{\overline{g}} \overline{h} = \frac{y_1 - \overline{y_1}}{e_1 - \overline{e_1}} \Rightarrow b = 1$.

Es gibt einen Extraktor $AL : (\tilde{P}, h, \overline{h}) \mapsto (x, b)$ mit $E_{w_{AL}} |AL| = O(|\tilde{P}| / \varepsilon_{\tilde{P}})$ sofern $\varepsilon_{\tilde{P}} \geq 2/q$. □

## 6.6. Geschichte des Paillier-Schemas

Goldwasser, Micali (1984)

**pk:** $n = p \cdot q$, ($p, q$ prim), $g \in (\mathbb{Z}_n^*)^2 = QR_n$.
**sk:** $\varphi(n)$.

**Verschlüsselung von $b \in \{0,1\}$.** Wähle $r \in_R \mathbb{Z}_n^*$: $cip = g^b r^2 \mod n$.

**Entschlüsselung von $cip$:**

$$b = \begin{cases} 0 & \text{falls } cip \in QR_n \\ 1 & \text{sonst.} \end{cases}.$$

**Sicherheit:** Verschlüsselung ist semantisch sicher, falls Entscheidung der quadratischen Residuosität schwierig ist.

(Benaloh, Fischer 1998, 1987)

**pk:** $n = p \cdot q$, sowie $s$ prim mit $s \mid p-1$, $s \nmid \frac{p-1}{s}$, $s \nmid q-1$. Ferner $g \in \mathbb{Z}_n^* \setminus (\mathbb{Z}_n^*)$ mit $s \mid \varphi(n)$, $\mathrm{ggT}(s, \frac{\varphi(n)}{s}) = 1$.
**sk:** $\varphi(n)$.

**Verschlüsselung von $M \in \mathbb{Z}_s$.** Wähle $r \in_R \mathbb{Z}_n^*$: $cip = g^M r^s \mod n$.

**Entschlüsselung von $cip$.** $M := \{i \in \mathbb{Z}_s$ falls $cip / g^j \in (\mathbb{Z}_n^*)^s$, d.h. falls

$$(cip/g^j)^{\varphi(n)/r} \equiv 1 \mod n\}.$$

**Sicherheit.** Verschlüsselung ist semantisch sicher, falls die Entscheidung für $x \in \mathbb{Z}_n^*$, $x \stackrel{?}{\in} (\mathbb{Z}_n^*)^s$ schwierig ist.

[**NaccacheStern1998**]

**Schlüsselerzeugung.** Sei $n = p \cdot q$. Sei $\sigma = p_1 p_2 \cdots p_k$ Teiler von $\varphi(n)$. $\mathrm{ggT}(\sigma, \frac{\varphi(n)}{\sigma}) = 1$ und $\mathrm{ggT}(p_i, p_j) = 1$ für $i \neq j$.

**pk:** $n = p \cdot q$, $\sigma$, $g \in \mathbb{Z}_n^*$ mit $\mathrm{ord}(g) = \sigma \cdot \tau$, $\tau$ groß.

**Verschlüsselung von $M \in \mathbb{Z}_\sigma$.** Wähle $r \in_R \mathbb{Z}_n^*$: $cip = g^M r^\sigma \mod n$.

**Entschlüsselung von $cip$.** $M := j \mod p_i$ falls $(cip/g^j)^{\frac{\varphi(n)}{p_i}} \mod n$. Nach CRT gilt

$$(M \mod p_1, \ldots, M \mod p_k) \mapsto M.$$

**Sicherheit.** Semantisch sicher falls das Erkennen von $p_i$-ten Potenzen schwierig ist für $i = 1, \ldots, k$.

[**OkamotoUchiyama1998**]

**Schlüsselerzeugung:** Sei $n = p^2 q$ mit $p, q$ prim. Sei $g \in \mathbb{Z}_n^*$ mit $\mathrm{ord}(g) = p(p-1)$.
**pk:** $n, g$
**sk:** $p$.

**Verschlüsselung von $M \in \mathbb{Z}_p$.** Wähle $r \in_R \mathbb{Z}_n^*$: $cip = g^M r^n \mod n$.

**Entschlüsselung von $cip$.**

$$M := \frac{cip^{p-1} \mod (p^2-1)}{p} \Big/ \frac{g^{p-1} \mod (p^2-1)}{p} \mod p.$$

KAPITEL 7

# Bit-Hinterlegung und perfekte Zufallsgeneratoren

Die beiden Hauptthemen dieses Kapitels sind die Bit-Hinterlegung und Zufallsgeneratoren. Bei der Bit-Hinterlegung handelt es sich um ein bekanntes Problem: *A* legt sich *B* gegenüber beim „commitment" auf ein Bit (1 oder 0) fest, möchte aber, dass *B* nicht weiss, was er gewählt hat. Später kann *A* seine Wahl gegenüber *B* „aufdecken", so dass *B* sich sicher sein kann, dass *A* seine Wahl plötzlich verändert hat. Hier wird vor allem das Naor-Schema besprochen.

Pseudozufallsgeneratoren sind Funktionen, die bei einer Eingabe von $n$ „echten" Zufallsbits mindestens $n+1$-viele neue Bits generieren, so dass diese alle statistischen Zufallszahlen-Tests bestehen und von dem „echt zufälligen" Ensemble $(U_{h(n)})_{n\in\mathbb{N}}$ ununterscheidbar sind. Nach Yao wird bewiesen dass die Definition von „Pseudozufall" nach der Ununterscheidbarkeit von echten Zufallsfolgen äquivalent ist zu der Unmöglichkeit Bits „vorherzusagen".

Im Detail wird der $x^2 \mod N$-Generator besprochen, der sehr einfach und zugleich sehr gut ist. Danach werden noch Verfahren beschrieben, wie man ein Geheimnis unter $n$ Personen aufteilen kann, so dass weniger als $t$ Personen keine Chance haben, das Geheimnis zu berechnen – $t$ Personen aber schon. Dieses Verfahren wird dann erweitert, so dass die Parteien auch lineare Rechenoperationen ausführen können, ohne das Gesamtergebnis erfahren zu können – und dabei können sogar $n/3$ der Personen die Korrektheit des Ergebnisses nicht beeinträchtigen!

## 7.1. Network-Modell – nicht uniformes Modell.

DEFINITION 7.1.1. Ein **Zufallsgenerator** vom Typ $h$ ist eine pol. Zeit Abb. $G : \{0,1\}^* \to \{0,1\}^*$ mit $G_{\{0,1\}^n} \subset \{0,1\}^{h(n)}$, $h(n) = n^{O(1)}$, $h(n) > n$.

**Bezeichnung:** $U_n$ sei Gleichverteilung auf $\{0,1\}^n$. $x \in_D \{0,1\}^n$ ist ein zufälliges $x$ gemäß Verteilung $D$, $G(D) : G(x)_{x \in_D \{0,1\}^n}$.

DEFINITION 7.1.2. Ein Zufallsgenerator $G$ vom Typ $h$ ist **pseudozufällig**, wenn es vom Ensemble $(U_{h(n)})_{n\in\mathbb{N}}$ pol. ununterscheidbar ist.

Damit ist der Zufallsgenerator $G$ perfekt gdw. das Ensemble $G(U_n)_{n\in\mathbb{N}}$ pseudozufällig ist.

DEFINITION 7.1.3. Ein Ensemble $(X_n)_{n\in\mathbb{N}}$ vom Typ $h$ oder $l$ ist (nach rechts) **unvorhersagbar**, wenn für jede prob. pol. Netzwerkfamilie $(C_n)_{n\in\mathbb{N}}$ und für alle $t > 0$ gilt

$$\max_{i=0,\ldots,h(n)-1} |Ws_{X_n}[C_{n,i}(x_1,\ldots,x_i) = x_{i+1}] - \frac{1}{2}| = O(n^{-t}).$$

und

$$C_{n,i}(x_1,\ldots,x_i) := C_n(x_1 \cdots x_i r_{i+1} \cdots r_{h(n)}) \quad \text{mit } r_i \in_R \{0,1\}.$$

PROPOSITION 7.1.4. *(Yao, 1982) Ein Ensemble ist genau dann (nach rechts) unvorhersagbar, wenn es pseudozufällig ist.*

Yao, FOCS 1982. Satz wurde von Yao im Vortrag vorgestellt und ist nicht im Artikel veröffentlicht.

BEWEIS. „⇐": Sei $C_{n,i}$ Netzwerkfolge zur Vorhersage von $x_{i+1}$ bei gegebenem $x_1 \cdots x_i$ mit Vorteil $n^{-t}$ bei $x \in \{0,1\}^{h(n)}$. $C_{n,i}$ liefert Netzwerkfolge $C'_{n,i}$ zur Unterscheidung von $(X_n)_{n\in\mathbb{N}}$ und $(U_{h(n)})_{n\in\mathbb{N}}$:

$$C'_{n,i}(x_1 \cdots x_{h(n)}) := C_{n,i}(x_1,\ldots,x_i) \oplus x_{i+1}.$$

Es gilt

$$\begin{aligned} & |Ws_{X_n}[C'_{n,i}(X_n) = 1] - Ws[C'_{n,i}(U_{h(n)}) = 1]| \\ = \quad & |Ws_{X_n}[C_{n,i}(x_1,\ldots,x_i) = x_{i+1}] - \frac{1}{2}|. \end{aligned}$$

Informal: Der Vorteil von $C_{n,i}$ bei der Vorhersage von $x_{i+1}$ zu gegebenem $x_1 \cdots x_i$ mit $x_1 \cdots x_{h(n)} \in_{X_n} \{0,1\}^{h(n)}$ ist gleich dem Vorteil von $C'_{n,i}$ beim Unterscheiden von $X_n$ und $U_{h(n)}$.

„⇒": Sei $C$ pol. Netzwerkfamilie, die $(X_n)_{n\in\mathbb{N}}$ und $(U_{h(n)})_{n\in\mathbb{N}}$ pol. unterscheidet. Dann gilt für ein $t > 0$ und unendlich viele $n$

$$|Ws[C_n(X_n) = 1] - Ws[C_n(U_{h(n)}) = 1]| \geq n^{-t}.$$

Sei $x_1 \cdots x_{h(n)} \in_{X_n} \{0,1\}^{h(n)}$, $r_q \cdots r_{h(n)} \in_{U_{h(n)}} \{0,1\}^{h(n)}$.

Betrachte die „hybride" Folge $x_1 \cdots x_i r_{i+1} \cdots r_{h(n)}$.

Es bezeichne

$$P_i = Ws[C_n(x_1 \cdots x_i r_{i+1} \cdots r_{h(n)}) = 1].$$

Es gilt

$$|P_0 - P_{h(n)}| = |Ws[C_n(X_n) = 1] - Ws[C_n(U_{h(n)}) = 1]|.$$

Ang. $|P_0 - P_{h(n)}| \geq n^{-t}$. Dann gibt es ein $i$ mit $|P_i - P_{i+1}| \geq n^{-t}/h(n) \geq n^{-t'}$. O.B.d.A. sei $p_{i+1} > p_i$. Das Netzwerk $C'_{n,i}$ zur Vorhersage von $x_{i+1}$ arbeite wie folgt:

$$C'_{n,i}(x_1 \cdots x_i) = \begin{cases} r_{i+1} & \text{falls } C_n(x_1 \cdots x_i r_{i+1} \cdots r_{h(n)}) = 1 \\ \neg r_{i+1} & \text{sonst.} \end{cases}$$

Im Fall $P_{i+1} > P_i$ ist diese Vorhersage gerechtfertigt.

Erfolgswahrscheinlichkeit der Vorhersage

$$Ws[C'_{n,i}(x_1 \cdots x_i) = x_{i+1} \mid x_{i+1} = r_{i+1}] = P_{i+1}$$

gilt nach Konstr. von $C'_{n,i}$ und Definition von $P_{i+1}$.

Es bezeichne

$$\begin{aligned} \bar{p} &:= Ws[C'_{n,i}(x_1 \cdots x_i) = x_{i+1} \mid x_{i+1} \neq r_{i+1}] \\ &= Ws[C_n(x_1 \cdots x_i \neg x_{i+1} r_{i+2} \cdots r_{h(n)}) = 0]. \end{aligned}$$

Es gilt

$$p_i = \frac{1}{2}(P_{i+1} + 1 - \bar{p})$$

somit $\bar{p} = 1 + p_{i+1} - 2p_i$.

Die Erfolgswahrscheinlichkeit der Vorhersage ist

$$\begin{aligned} \frac{1}{2}(P_{i+1} + \bar{p}) &= \frac{1}{2}(P_{i+1} + 1 + P_{i+1} - 2p_i) \\ &= \frac{1}{2} + P_{i+1} - P_i. \end{aligned}$$

□

COROLLARY 7.1.5. *Das Ensemble $(X_n)_{n\in\mathbb{N}}$ ist nach rechts unvorhersagbar gdw. es nach links unvorhersagbar ist.*

BEWEIS. $X_n^{sp}$ sei die Verteilung von $x_{h(n)} \cdots x_1$ mit $x_1 \cdots x_{h(n)} \in_{X_n} \{0,1\}^{h(n)}$. Mit $(X_n)_{n\in\mathbb{N}}$ ist auch $(X_n^{sp})_{n\in\mathbb{N}}$ pseudozufällig. □

## 7.2. Bit Hinterlegung (Bit Commitment)

**Hinterlegung:** Prover $P$ hinterlegt Zufallsbit $b \in_R \{0,1\}$ in verschlüsselter Form an Verifier $V$.

**Aufdecken:** $P$ entschlüsselt $b$ und $V$ verifiziert.

| | $P, C_n$ | | $V, C_n$ |
|---|---|---|---|
| 0. | | $\leftarrow w_V$ | |
| 1. | $c := C_n(w_V, w_P, b)$ | $c \rightarrow$ | |
| 2. | | $w_P, b \rightarrow$ | $c \stackrel{?}{=} C_n(w_V, w_P, b)$ |

$V$ will $b$ aus der Verschlüsselung erraten. $P$ will wahlweise $b = 0$ oder $b = 1$ aufdecken.

Es seien $C_n, V_n$ pol. Zeit Turing Maschinen (uniform) oder Boolesches Netzwerk der Größe $poly(n) = n^{O(1)}$ (nicht uniform).

**Comp. zero-knowledge.** (Sicherheit von $P$ gegen $V$)

$$\max_{w_V, \bar{w}_V} |Ws_{w_p}[V_n(w_V, \bar{w}_V, c_n(w_V, w_P, b) = b] - \frac{1}{2}| = O(n^{-t}) \quad \forall t > 0.$$

d.h. der Vorteil von $V$ beim Erraten von $b$ ist vernachlässigbar.

**Soundness.** (Sicherheit von $V$ gegen $P$)

$$\max_{w_P^0, w_P^1} Ws_{w_V}[c_n(w_V, w_P^0, 0) = c_n(w_V, w_P^1, 1)] = O(n^{-t}) \quad \forall t > 0.$$

Betrug von $P$, $P$ hat $w_p^0, w_p^1$ für $b = 0, 1$.

EXAMPLE 7.2.1. Ein schönes Beispiel für ein Bit-Hinterlegungsprotokoll ist in [**Salomaa1996**] gegeben:

Zunächst einigen sich $P$ und $V$ auf eine große Primzahl $p$ und den Generator $\langle g \rangle = \mathbb{Z}_p^*$. Dann wählt $V$ eine zufällige Zahl $r \in_R \mathbb{Z}_p$ und teilt diese $P$ mit. Wie jedes Element aus $\mathbb{Z}_p$,so lässt sich auch $r$ darstellen als $r \equiv g^e \mod p$. Die Berechnung von $e$ ist jedoch schwer unter der Annahme, dass das DL-Problem schwer ist.

$P$ wählt nun sein Bit $b$ und eine Zufallszahl $y \in_R \mathbb{Z}_p$ und teilt $V$ die Zahl $x \equiv r^b g^y \mod p$ mit. Möchte $P$ nun $V$ sein Bit $b$ mitteilen, so verrät es $V$ die Zahl $y$, so dass $V$ die Berechnung $x \cdot g^{-y}$ anstellen kann und so entweder $r$ (für $b = 1$) oder 1 (für $b = 0$) herausbekommt. Die Zahl $y$ ermöglicht durch seine zufälige Wahl nur das Veröffentlichen des zugehörigen $b$'s und ist nicht für andere möglich.

Wollte $P$ betrügen, also vielleicht im Nachhinein sein $b$ ändern, so müsste er zwei Zahlen $y, y'$ besitzen, für die gilt

$$g^y \mod p \equiv r g^{y'} \mod p.$$

In diesem Fall könnte er im Nachhinein durch das Verschicken von $y$ bzw. $y'$ den Wert von $b$ „beeinflussen". Wäre er aber im Besitz dieser beiden Zahlen, dann wüsste er auch den diskreten Logarithmus von $r$, da gilt

$$r \equiv g^{y-y'} \mod p,$$

was ein Widerspruch zu der Annahme oben wäre. Ein Betrug wäre also nur bei Kenntnis des diskreten Logarithmus möglich.

## 7.3. Naor-Schema

Journal of Cryptology 4 (1991)

Sei $G : \{0,1\}^n \to \{0,1\}^{3n}$ perfekter (kryptographisch sicherer) Pseudo-Zufallsgenerator (PZG) vom Typ $3n$.

**Hinterlegen:** $V$ sendet $w_V \in_R \{0,1\}^{3n}$. $P$ hinterlegt $C := G(w_p) \oplus (b \wedge w_V)$[1] zu $w_p \in_R \{0,1\}^n$. $b \wedge w_V := \begin{cases} w_v & \text{fuer } b = 1 \\ 0 & \text{fuer } b = 0 \end{cases}$.

**Aufdecken:** $P$ sendet $w_p, b$. $V$ prüft $C \stackrel{?}{=} G(w_p) \oplus (b \wedge w_V)$.

[1] Mit $\oplus$ ist das Bitweise XOR gemeint.

PROPOSITION 7.3.1. *Das Naor Schema ist sound und für perfekte PZG auch comp. zero-knowledge.*

BEWEIS. Wir werden wieder die beiden Eigenschaften soundness und completeness zeigen:

**sound:** Zum Betrug benötigt $P$ $w_P^0, w_P^1$ mit $G(w_p^0) \oplus G(w_p^1) = w_V$. Es gilt

$$Ws_{w_V}[\exists w_P^1, w_P^0 \in \{0,1\}^n : G(w_P^0) \oplus G(w_P^1) = w_V] \leq 2^{-n} = 2^{2n}/2^{3n},$$

denn

$$\#\{w_V\} = 2^{3n}, \#\{(w_P^0, w_P^1)\} = 2^{2n}.$$

**comp. zero-knowledge:**

$$\begin{aligned} P_i(w_V) &:= Ws_{w_P}[V_n(w_V, G(w_P) \oplus (i \wedge w_V)) = 1] \quad \text{fuer } i = 0,1 \\ \text{Vort}(w_V) &:= \frac{1}{2} + \frac{P_1(w_V) - P_0(w_V)}{2}. \end{aligned}$$

Ang.

$$\exists t > 0, w_V : \text{Vort}(w_V(n)) \geq \frac{1}{2} + n^{-t}$$

für unendliche viele $n$. Dann

$$P_1(w_V) - P_0(w_V) \geq 2n^{-t}.$$

Ersetze $G(w_P)$ durch $w \in_R \{0,1\}^{3n}$.

$$\begin{aligned} \bar{P_i}(w_V) &:= Ws_w[V_n(w_V, w \oplus (i \wedge w_V)) = 1] \quad \text{fuer } i = 0,1 \\ & \bar{P_1}(w_V) \quad \bar{P_0}(w_V) \\ & P_1(w_V) - P_0(w_V) = 2n^{-t}. \end{aligned}$$

Offenbar gilt $\bar{P_1}(w_V) = \bar{P_0}(w_V)$, denn $w$ und $w \oplus (i \wedge w_V)$ sind gleichverteilt über $\{0,1\}^{3n}$.
Somit gilt

$$\begin{aligned} \text{entweder} \quad & |\bar{P_1}(w_V) - P_1(w_V)| \geq n^{-t} \\ \text{oder} \quad & |\bar{P_0}(w_V) - P_0(w_V)| \geq n^{-t}. \end{aligned}$$

In beiden Fällen sind die Ensembles

$$(U_{3n})_{n \in \mathbb{N}} \quad \text{und} \quad G(U_n)_{n \in \mathbb{N}}$$

pol. Zeit unterscheidbar. Widerspruch zu $G$ perfekt.

□

## 7.4. Der $x^2 \mod N$-Generator und quadratische Residuosität

Sei $N = p \cdot q$ Blum-Zahl, $p, q$ prim, $p, q = 3 \mod 4$.

$|QR_N| = \frac{p-1}{2} \cdot \frac{q-1}{2}$ ist ungerade.

$x \mapsto x^2$ ist Permutation auf $QR_N$.

$$B_n = \{N \in \mathbb{N} \mid N = p \cdot q \text{ Blum} - \text{Zahl und } 2^{n/2-1} < p, q < 2^{n/2}\}.$$

Es bezeichne zu $x \in \mathbb{Z}_N$, $[x]_N \in [0,N[$, $[x]_N = x \mod N$ und $\chi_N : \mathbb{Z}_N^* \to \{0,1\}$ die **charakteristische Funktion** von $QR_N$, d.h.

$$\chi_N(x) = \begin{cases} 1 & x \in QR_N \\ 0 & \text{sonst.} \end{cases}$$

**7.4.1. Die quadratische Residuositätsannahme (QRA).** Für jede (prob.) polynomiale Netzwerk-Familie $(C_n)_{n\in\mathbb{N}}$ und alle $t > 0$ gilt für $N \in_R B_n$, $z \in_R \mathbb{Z}_N^*(+1)$:

$$\left| Ws_{z,N}[C_n(z,N) = \chi_N(z)] - \frac{1}{2} \right|_{\text{fuer alle } t>0} = O(n^{-t}).$$

REMARK 7.4.1. Für Blum-Zahlen $N$ gilt $|\mathbb{Z}_N^*(+1)| = 2|QR_N|$ und somit $Ws_{z\in_R\mathbb{Z}_N^*(+1)}|\chi_N(z) = 1| = \frac{1}{2}$.

Blum-Zahlen der Form $4r+3$ besitzen als hilfreichste Eigenschaft, dass $-1$ ein Quadratischer Nichtrest ist und der Wert des Jacobi-Symbols gleich $+1 \mod n$ ist.

Nach der QRA wird also kein Algorithmus die quadratische Residuosität einer Zahl mit Ws. signifikant größer als $1/2$ bestimmen.

**7.4.2. Der $x^2 \mod N$-Generator.**

**Eingabe:** $N \in B_n$, $x \in QR_N$. $x_1 := x$, $x_{i+1} := x_i^2 \mod N$ für $i = 1,\ldots,\ell(n)-1$.
**Ausgabe:** $([x_i]_N \mod 2$ für $i = 1,\ldots,\ell(n))$.

PROPOSITION 7.4.2. *(Blum, Blum, Shub) Unter der QRA ist der $x^2 \mod N$-Generator bei gegebenem $N$ nach links unvorhersagbar und somit perfekt. (Genauer: zu gegebenem $\pm x_i, x_{i+1}, N$ ist $[x_i]_N \mod 2$ nicht vorhersagbar)*

BEWEIS. Es bezeichne $b_i = [x_i]_N \mod 2$ das $i$-te Ausgabebit des $x^2 \mod N$-Generators. Wir transformieren eine beliebige Netzwerk-Familie $C_{n,i}$, die $b_i$ aus $(b_{i+1},\ldots,b_{\ell(n)},N)$ vorhersagt, in eine Netzwerk-Familie $C'_{n,i}$, welche $\chi_N(z)$ zu $z \in_R \mathbb{Z}_N^*(+1)$ und $N$ vorhersagt. Bei der Transformation bleibt die Erfolgswahrscheinlichkeit erhalten.

Das Netzwerk $C'_{n,i}$:
Eingabe: $N \in_R B_n$, $z \in_R \mathbb{Z}_N^*(+1)$, $x_{i+1} := z^2$, $b_{i+1} := [x_{i+1}]_N \mod 2$, $x_j := x_{j-1}^2$, $b_j := [x_j]_N \mod 2$ für $j = i+2,\ldots,h(n)$.
Ausgabe: 1, wenn $C_{n,i}(b_{i+1}\cdots b_{h(n)},N) = z \mod 2$. 0, sonst.

Bez.: $x_i := \sqrt{x_{i+1}} \in QR_N$.

Es gilt $x_i \in \{\pm z\}$, $x_i = z \Leftrightarrow z \in QR_N$. Somit $z = [x_i]_N \mod 2 \Leftrightarrow z \in QR_N$. Das Bit $[x_i]_N \mod 2$ trägt (zu gegebenem $x_i = 1$) die QR-Information von $z$. $C_{n,i}, C'_{n,i}$ haben gleiche Erfolgswahrsch. Es gilt jeweils $x_{i+1} \in_R QR_N$. □

Das QR-Problem bei der Vorhersage des $x^2 \mod N$-Generators ist

$$\pm x_i, x_{i+1} \mapsto [x_i]_N \mod 2.$$

Das wirklich schwierige Problem ist die Berechnung $x_{i+1} \mapsto \pm x_i$, nämlich die Berechnung von $\sqrt{x_{i+1}} \mod n$.

LEMMA 7.4.3. *Sei $AL_N : QR_N \to \mathbb{Z}_N^*(+1)$ beliebiger SQRT-Alg. Dann gilt für $x \in_R \mathbb{Z}_N^*$: Mit $Ws_x = \frac{1}{2}$ ist*

$$ggT(AL_N(x^2) \pm x, N) = \{p, q\}.$$

BEWEIS. $x^2 \mod N \in QR_N$ hat 4 Quadrat-Wurzeln in $\mathbb{Z}_N^*$, jeweils zwei $\mod p$, $\mod q$, $\pm\sqrt{x} \mod p$.

$$\begin{aligned} x \neq \pm AL_N(x^2) \quad &\Leftrightarrow \quad x = \pm AL_N(x) \mod p \\ &\qquad\quad x = \mp AL_N(x) \mod q \\ \Leftrightarrow \quad \mathrm{ggT}(AL_N(x^2) \pm x, N) \quad &= \quad \{p, q\}. \end{aligned}$$

Wegen $x \in_R \mathbb{Z}_N^*$ tritt der Fall $x \neq \pm AL_N(x^2)$ genau mit $Ws_x = \frac{1}{2}$ ein. □

Alexi, Chor, Goldreich, Schnorr (1988), Siam, J. Comp. 17,2

PROPOSITION 7.4.4. *Der $x^2 \mod N$-Generator mit $E_N(x) =_{def} \min([x^2]_N, N - [x^2]_N)$ statt $x^2 \mod N$ ist perfekt, falls die Zerlegung von N schwierig ist.*

$E_N : [0, N/2[ \to [0, N/2[$ ist Permutation. Weil $-1 \in \mathbb{Z}_N^*(+1) \setminus QR_N$ ist genau einer von $[x^2]_N, N - [x^2]_N$ in $QR_N$.

**RSA-Funktion:** $\mathrm{ggT}(e, \varphi(N)) = 1$, zu gegebenem $e, N$.

$$\begin{aligned} E_N^{RSA} : \mathbb{Z}_N^* \quad &\to \quad \mathbb{Z}_N^* \\ x \quad &\mapsto \quad [x^e]_N \end{aligned}$$

mit $x_{i+1} := E_N^{\overset{u, uc}{RSA}}(x_i)$. Ausgabe $[x_{i+1}]_N \mod 2^k$ falls $k$-Bit prob Runde.
**Rabin-Funktion:** (nicht zentriert) $E_N^u : x \mapsto [x^2]_N$.
**Rabin-Funktion:** (zentriert) $E_N^c : x \mapsto x^2 \mod N \quad \in ]-N/2, N/2[$
**Rabin-Funktion:** (absolute) $E_N^a : x \mapsto \min([x^2]_N, N - [x^2]_N)$
**Fakt:**

(1) $E_N^u$ permutiert $QR_N \cap ]0, N[$
(2) $E_N^c$ permutiert $QR_N \cap ]-N/2, N/2[$
(3) $E_N^q$ permutiert $\mathbb{Z}_N^*(+1) \cap ]0, N/2[$ falls $(-\frac{1}{N}) = 1$.

Zu gegebenem $\pm x_i$, $x_{i+1} := E_N^c(\pm x_i)$ trägt $x_i \mod 2$ die $QR_N$-Information $x_i \in QR_N$.

PROPOSITION 7.4.5. *Zu gegebenem $N \in_R B_n$, $x_{i+1} := E_N^{\overset{u}{a}}(x_i)$ mit $x_i \in_R \mathbb{Z}_N^*(+1)$ sind die $\log_2 \log_2 N$ niedrigsten Bits von $x_i$ (simultan) pseudozufällig, sofern N schwer zerlegbar ist. $N = 2^{1024}$ asymptotisch $\log_2 \log_2 N$.*

Konkret Korollar 6 [FS]: $N \approx 2^{5000}$: 8 niedrigsten Bit sind simultan sicher.

PROPOSITION 7.4.6. *Für die RSA-Funktion sind zu gegebenem N, $x_{i+1} := [x_i^e]_N$ mit $x_i \in_R \mathbb{Z}_N^*$ die $\log_2 \log_2 N$ niedrigsten Bits von $x_i$ (simultan) pseudozufällig sofern $E_N^{RSA}$ schwer invertierbar ist. [FS]Th2, P.236 par. 2.*

CONCLUSION 7.4.7. Beim $x^2 \mod N$-/RSA-Generator kann man pro Runde die $\log_2 \log_2 N$ niedrigsten (für $N = 2^{5000}$, 8) Bits von $E_N^{RSA}(x_i)$ bzw. $E_N^a(x_i)$ ausgeben.

Der zugehörige Generator ist perfekt, sofern $E_N^{RSA}$ schwer invertierbar bzw. $N$ schwer zerlegbar ist.

PROBLEM 7.4.8. Ist die Entscheidung $x \in QR_N$ für $x \in_R \mathbb{Z}_N^*(+1)$ so schwierig wie die Zerlegung von $N$?

Beim $E_N^c$-Generator wurde das $QR_N$-Bit $x_i \mod 2$ und die Berechnung $x_{i+1} \mapsto x_i = \sqrt{\pm x_{i+1}}$ getrennt. Die beiden Sicherheitsbeweise unter *QRA* bzw. Faktorisierungsannahme liefern keine Abhängigkeit von QRA und Faktorisierungsannahme.

Für den $x^2 \mod N$-Generator mit $E_N^u, E_N^a, E_N^c$ ist $x_{i+1} \mapsto \pm x_i$ so schwer wie die Zerlegung von $N$.

Ein $QR_N$-Orakel für $\pm x_i \in QR_N$ hilft nicht: Berechnung $x_{i+1} \mapsto x_i = \sqrt{\pm x_{i+1}}$.

Wichtigste Beweistechnik:

**7.4.3. Tschebycheff'sche Ungleichung (TU).** Für jede Zufallsvariable $X$ gilt

$$\begin{aligned} Ws[|X - E[X]| \geq \varepsilon] &\leq \frac{VAR[X]}{\varepsilon^2} \\ \text{mit} \quad VAR[x] &:= E[(E[X] - X)^2] \end{aligned}$$

wobei $E[X]$ der Erwartungswert ist und $VAR[X]$ der Erwartungswert der quadratischen Abweichung.

Mit der TU kann man einen Vorteil über **Merheitsentscheidung** vergrößern. Aus $poly(n)^{-1}$-Vorteil erhält man eine fast sichere Entscheidung.

EXAMPLE 7.4.9. $QR_N$-Entscheidung für Blumzahl $N$, $\left(\frac{-1}{N}\right) = 1$.
Das Netzwerk $C_n$ habe $\varepsilon$-Vorteil bei der QR-Entscheidung. Für $x \in_R \mathbb{Z}_N^*(+1)$ gelte

$$Ws_x[C_n(x,N) = \chi'_N(x)] = \frac{1}{2} + \varepsilon$$

mit

$$\chi'_N(x) := \begin{cases} 1 & x \in QR_N \\ -1 & x \in \mathbb{Z}_N^*(+1) - QR_N. \end{cases}$$

**Mehrheitsentscheidung zu $\chi'_N(x)$.**

(1) Wähle $x_1, \ldots, x_m \in_R \mathbb{Z}_N^*(+1)$, $b_1, \ldots, b_m \in_R \{\pm 1\}$.
(2) $C_n^m(x) := \text{sig}(\sum_{i=1}^m b_i C_N(b_i x_i^2 x))$

Es gilt:

$$\begin{aligned} b_i C_N(b_i x_i^2 x) &= \chi'_N(x) \\ \Leftrightarrow \quad C_N(b_i x_i^2 x) &= \chi'_N(b_i x_i^2 x). \end{aligned}$$

$C_n^m$ fällt eine Mehrheitsentscheidung: falsch, wenn die Mehrheit der $C_N(b_i x_i^2 x)$ richtig ist.

PROPOSITION 7.4.10. *Hat $C_n$ $\varepsilon$-Vorteil bei der $QR_N$-Entscheidung, dann gilt*

$$Ws_{b_i, x_i}\left[C_N^m(x) \neq \chi'_N(x)\right] < \frac{1}{4m\varepsilon^2}$$

*für beliebige $x \in \mathbb{Z}_N^*(+1)$.*

BEWEIS. Für bel. $x \in \mathbb{Z}_N^*(+1)$ ist $b_i x_i^2 x \in_R \mathbb{Z}_N^*(+1)$. Sei

$$R_i = \begin{cases} 1 & \text{falls } b_i \in_N (b_i x_i^2 x) = \chi'_N(x) \\ 0 & \text{sonst.} \end{cases} .$$

$$\begin{aligned} E[R_i] &= \frac{1}{2} + \varepsilon \\ VAR[R_i] &= \left(\frac{1}{2} + \varepsilon\right)\left(\frac{1}{2} - \varepsilon\right) < \frac{1}{4} \\ E[\sum_{i=1}^{m} R_i] &= \frac{m}{2} + m\varepsilon \\ VAR[\sum_{i=1}^{n} R_i] &= \sum_{i=1}^{m} VAR[R_i] < \frac{m}{4} \text{ (fuer paarweise unabh. } R_i) \end{aligned}$$

Es gilt

$$\begin{aligned} Ws_{b_i,V_i}[C_N^m(x) \neq \chi'_N(x)] &\leq Ws[\sum_{i=1}^{m} R_i \leq \frac{m}{2}] \\ &= Ws[\sum_{i=1}^{m} R_i - E[\sum_{i=1}^{m} R_i] \leq -m\varepsilon] \\ &\leq Ws[|\sum R_i - E[\sum R_i]| \geq m\varepsilon] \\ &\leq^{TU} \frac{VAR[\sum R_i]}{(m\varepsilon)^2} \leq \frac{m}{4m^2\varepsilon^2} = \frac{1}{4m\varepsilon^2}. \end{aligned}$$

□

DEFINITION 7.4.11. $D(AL)$ ist der **Definitionsbereich** des partiellen Algorithmus AL.

Ein Generator wird wie folgt definiert:

DEFINITION 7.4.12. $(G,g)$ ist **Generator** zum Ensemble $X = (X_n)_{n\in\mathbb{N}}$, wenn $X$ und $G(U_{g(n)})_{n\in\mathbb{N}}$ statistisch ununterscheidbar sind.

$$Ws_{G(U_{g(n)})}[z] := Ws_{x\in_R\{0,1\}^{g(n)}}[G(x) = z \mid x \in D(G)].$$

Desweiteren ist:

DEFINITION 7.4.13. $(G,g)$ ist **polynomial**, wenn $G, g(1^n)$ pol. Zeit sind und $2^{g(n)}/|D(G) \cap \{0,1\}^{g(n)}| = n^{O(1)}$.

Für ein Ensemble gilt folgende Definition:

DEFINITION 7.4.14. Das Ensemble $X$ ist **polynomial**, wenn es von einem pol. Generator erzeugt wird.

Seien $X_n, Y_n$ Verteilungen über der Menge $S$ mit

$$\|X_n - Y_n\|_1 = O(n^{-t}) \quad \text{fuer alle } t > 0.$$

LEMMA 7.4.15. *Sei $X$ pol. Ensemble, dann gibt es einen pol. Generator $(G',g')$ mit $2^{g'(n)}/|D(G') \cap \{0,1\}^{g'(n)}| \leq 2$.*

BEWEIS. Sei $(G,g)$ Generator zu $X$, $2^{g(n)}/|D(G)\cap\{0,1\}^{g(n)}| \leq n^t$ *mit* $t \in \mathbb{N}$. Setze $g'(n) := g(n)\cdot n^t$.

$G'(x)$, $|x| = g'(n) : x = \bar{x_1}\cdots\bar{x}_{n^t} \in (\{0,1\}^{g(n)})^{n^t}$.

$G'(x) := G(\bar{x_i})$ für das minimale $i$ mit $\bar{x_i} \in D(G)$.

$|0,1\}^{g'(n)} - D(G)| \leq 2^{g'(n)}(1-n^{-t})^{n^t} \leq 2^{g'(n)}/e$

$2^{g'(n)/|\{0,1\}^{g'(n)}\cap D(G)| \leq \frac{1}{1-1/e} < 2}$. □

PROPOSITION 7.4.16. *Sei* $P_n = \{p \in [3,2^n] \mid p \text{ prim}\}$, $U_{P_n}$ *die Gleichverteilung auf* $P_n$. *Dann ist* $(U_{P_n})_{n\in\mathbb{N}}$ *polynomial.*

BEWEIS. Der Generator $(G,g)$, $g(n) = n+2n^2$

$$G : \{0,1\}^{g(n)} \to [0,2^n),$$

Eingabe: $x = Nw_1\ldots w_n \in \{0,1\}^{g(n)} \cong [0,2^n)[0,2^{2n})^n$.

(1) Stoppe falls $N$ gerade oder $N = p^k$ mit $p$ prim, $k \geq 2$ oder $\exists i : \mathrm{ggT}(N,w_i) = 1$.
(2) **if** $[w_i^{\frac{N-1}{2}} \equiv \left(\frac{w_i}{N}\right) \mod N$ oder $w_i \equiv 0 \mod N]$ für $i = 1,\ldots,n$ **then** $G(x) := N$.

Fälle:

(1) $\|w_1 \mod N - U_{\mathbb{Z}_N}\|_1 \leq 2^{-n}$
(2) $\|X_1\times X_k - Y_1\times\ldots\times Y_k\|_1 \leq \sum_{i=1}^k \|X_i - Y_i\|_1$
(3) $\|(w_1 \mod N,\ldots,w_n \mod N) - U_{(\mathbb{Z}_n)^n}\|_1 \leq n\cdot 2^{-n}$

LEMMA 7.4.17. *Ang.* $N$ *hat ungerade Primfaktoren* $p \neq q$. *Dann gilt für* $w \in_R \mathbb{Z}_N^*$: $Ws[w^{\frac{N-1}{2}} = \left(\frac{w}{N}\right) \mod p] \leq 2^{-n}$.

(4) $\forall p \in P_n : Ws_x[G(x) = p] = 2^{-n}$
(5) $\forall N \in [0,2^n) - P_n : Ws_X[G(x) = N] \leq 2^{-n}(N^{-n} + \underbrace{2^{-n}}_{\text{Lemma 5}} + \underbrace{n2^{-n}}_{(3)})$
(6) $\|G(U_{g(n)}) - U_{P_n}\|_1 = \sum_{p\in P_n} |\frac{Ws_x G(x)=p}{Ws_x[x\in D(G)]} - 2^{-n}| + \sum_{N\notin P_n} Ws_x[G(x) = N]$ für $x \in_R \{0,1\}^{g(n)}$
(7) $\frac{|P_n|}{2^n} \leq Ws_x[x \in D(G)] \leq \frac{|P_n|}{2^n} + 2^{-n}\underbrace{(n+1)}_{(5)} + N^{-n}$

Somit

$$\|G(U_{g(n)}) - U_{P_n}\|_1 \leq 2\cdot\sum_{N\notin P_n} Ws_x[G(x) = N] \leq 2(N^{-n} + 2^{-n}(n+1)).$$

und

$$\begin{aligned} 2^{g(n)}/|D(G)\cap\{0,1\}^{g(n)}| &\leq \frac{2^n}{|P_n|} \\ (\text{Primzahlsatz}) &= O(\log 2^n) = n\log 2 + O(1). \end{aligned}$$

□

## 7.5. Verifiable Secret Sharing nach Pedersen

[**Pedersen1991**]
P. Feldman FOCS, 1987.

**Pedersen Commitment einer Nachricht in $\mathbb{Z}_q$.** Sei $G = \langle g \rangle$, $|G| = q$ prim. $h \in_R G$.

**Hinterlegen von $s \in \mathbb{Z}_q$:** Wähle $t \in_R \mathbb{Z}_q$: hinterlege $E(s,t) = g^s h^t$.
**Aufdecken:** Zeige $s,t$.
**Fakt:** Für $t \in_R \mathbb{Z}_q$ ist $E(s,t) \in_R G$ stat. unabh. von $m$.
**Soundness:** Kann $P$ die Hinterlegung $E(s,t)$ mit $m' \neq m$ aufdecken, so gilt

$$\begin{aligned} g^s h^t &= g^{s'} h^{t'} \quad \text{und somit} \\ \log_g h &= \frac{s-s'}{t'-t} \mod q. \end{aligned}$$

THEOREM 7.5.1. *Pedersen-Commitment ist sound falls DL-Problem schwer is.*

**Secret Sharing:** Dealer $D$ will Geheimnis $s \in \mathbb{Z}_q$ so an $P_1,\ldots,P_n$ aufteilen, dass je $k$ Shareholder $s$ rekonstruieren können, aber weniger als $k$ Shareholder keine Shannon-Information über $s$ erhalten ($(k,n)$ **Threshold Schema**, Shamir $(k,n)$-Threshold Schema, 1 Shamir Comm., ACM 1979):
$D$ wählt $f = s + \sum_{i=1}^{k-1} F_i x^i \in \mathbb{Z}_q[x]$ mit $f(0) = s$, und gibt $P_i$ den Anteil $f(i)$ für $i = 1,\ldots,n$.
**Rekonstruktion von $s$:** $P_{i_1},\ldots,P_{i_k}$ die **Interpolationsformel**

$$f(x) = \sum_{j=1}^{k} \prod_{l \neq j} \frac{x - i_j}{i_l - i_j} d(i_j).$$

Danach gilt

$$s = \sum_{j=1}^{k} \prod_{l \neq j} \frac{i_j}{i_j - i_l} f(i_j).$$

**Fakt:** Weniger als $k$ Funktionswerte von $f$ geben keine Information über $f(0)$.

**Forderungen an die Verifikation.**

**F1.:** Wenn $D$ das Protokoll befolgt, akzeptieren $P_1,\ldots,P_n$.
**F2.:** Akzeptieren $(P_i)_{i\in S_1}$, $(P_i)_{i\in S_2}$ ihre Anteile und $\#S_1, \#S_2 = k$ dann rekonstruieren $(P_1)_{i\in S_1}, (P_i)_{i\in S_2}$ dasselbe Geheimnis.

**Verifikations-Protokoll.**

(1) $D$ wählt $t \in_R \mathbb{Z}_q$ publiziert $E_0 := E(s,t) = g^s h^t$.
(2) $D$ wählt $F(x) = s + \sum_{j=1}^{k-1} F_j x^j \in \mathbb{Z}_q[x]$, berechnet $s_i := F(i)$ für $i = 1,\ldots,n$.
$D$ wählt $G(x) = t + \sum_{j=1}^{k-1} G_j x^j \in \mathbb{Z}_q[x]$, berechnet $t_i := G(i)$ für $i = 1,\ldots,n$ und publiziert $E_j := E(F_j, G_j)$ für $j = 1,\ldots,k-1$ (Commitment).
(3) $D$ sendet $(s_i,t_i)$ privat an $P_i$ für $i = 1,\ldots,n$.
(4) $P_i$ akzeptiert, wenn $\underbrace{E(s_i,t_i)}_{=g^{s_i}h^{t_i}} \stackrel{?}{=} \prod_{j=0}^{k-1} E_j^{i^j} = \prod_{j=0}^{k-1} \left(g^{F_j} h^{G_j}\right)^{i^j}$.

LEMMA 7.5.2. *Sei $S \subset \{1,\dots,n\}$ Menge von k Teilnehmern, die ihre Anteile akzeptieren. Diese finden $(s',t')$ mit $E_0 = E(s',t')$.*

BEWEIS. Die Teilnehmer in $S$ bestimmen die Polynome $F', G' \in \mathbb{Z}_q[x]$ vom Grad $\leq k-1$ mit

$$F'(x) = s_i \quad , \quad G'(i) = t_i \quad \text{fuer } i \in S$$

und setzen $s' = F'(0)$, $t' := G'(0)$.

Sei $h = g^d$, dann gilt für $i \in S$:

$$g^{F'(i)+dG'(x)} = E(s_i,t_i) = g^{s_i+dt_i}.$$

Damit ist $F' + dG' \in \mathbb{Z}_q[x]$ das eind. bestimmte Polynom $P$ vom Grad $\leq k-1$ mit $P(i) = s_i + dB_i$ für $i \in S$.

Sei $E_j = g^{e_j}$ für $j = 0,\dots,k-1$. Das Polynom $e(x) = \sum_{j=0}^{k-1} e_j x^j$ erfüllt $e(i) = s_i + dt_i$ für $i \in S$. Somit gilt

$$\begin{aligned} e(x) &= (F' + dG')(x) \\ \Rightarrow \quad E_0 &= g^{e_0} = g^{e(0)} = g^{F'(0)} h^{G'(0)} \\ &= g^{s'} h^{t'}. \end{aligned}$$

□

**Berechnung von $s,t$.** $s = \sum_{i\in S} a_i s_i$, $t := \sum_{i\in S} a_i t_i \mod q$ mit $a_i = \prod_{j\in S, j\neq i} \frac{i}{i-j} \mod q$.

FACT 7.5.3. *Wenn D dem Protokoll folgt, dann akzeptieren $P_1,\dots,P_n$.*

BEWEIS.

$$\begin{aligned} E(s_i,t_i) &\stackrel{?}{=} \prod_{j=0}^{k-1} E_j^{ij} \\ g^{s_i} h^{t_i} &= \prod_{j=0}^{k-1} (g^{F(j)} h^{G(j)})^{i_j} \quad \text{mit } s_i = F(i), t_i = G(i) \\ &= g^{\sum_{j=0}^{e-1} F_j \cdot i^j} h^{\sum_{j=0}^{k-1} G_j \cdot i^j} = g^{s_i} h^{t_i}, \text{ weil } F(0) = s, G(0) = t. \end{aligned}$$

□

THEOREM 7.5.4. *Es gilt F2, es sei denn, dass D $\log_g h$ berechnen kann.*

BEWEIS. $D$ kann prüfen, dass die $(s_i,t_i)$ für $i = 1,\dots,n$ konsistent sind, oder dass es Polynome

$$F = \sum_{j=0}^{k-1} F_j x^j \quad , \quad G = \sum_{j=0}^{k-1} G_j x^j \in \mathbb{Z}_q[x]$$

gibt mit $F(i) = s_i$, $G(i) = t_i$ für $i = 1,\dots,n$.

Im Fall der Inkonsistenz fordert er zwei Teilmengen $S, S' \subset \{1,\dots,n\}$ mit $k$ Teilnehmern, deren Shares $(s_i,t_i)$ mit $i \in S \cup S'$ inkonsistent sind. Haben die Teilnehmer in $S \cup S'$ korrekt akzeptiert, gilt

$$E(s',t') = E_0 = E(s,t)$$

mit $(s,t) \neq (s',t')$. $D$ berechnet $(s,t)$ und $(s',t')$ wie $(P_i)_{i\in S}$ und $(P_i)_{i\in S'}$ und erhält $\log_g h = \frac{s-s'}{t'-t} \mod q$. □

## 7.6. Rechnen mit verteilten Daten

[**Pedersen1991, Ben-Or et al. 1988**]

$G = \langle g \rangle$, $|G| = q$ prim, $h \in_R G$.

**Hinterlegen von** $s \in \mathbb{Z}_q$**:** Wähle $t \in_R \mathbb{Z}_q$: $E(s,t) = g^s h^t$.

**Shamir's** $(k,m)$**-Schema.** Dealer $D$ wählt $F = s + \sum_{i=1}^{k-1} F_i x^i \in \mathbb{Z}_q[x]$ zufällig mit $f(0) = s$ und gibt shareholder $P_i$ den Anteil $f(x_i)$ für $i = 1,\ldots,n$.

**Verifikations-Protokoll.**

(1) $D$ publiziert $E_0 := E(s,t)$,
(2) wählt $F = s + \sum_{i=1}^{h-1} F_i x^i \in \mathbb{Z}_q[x]$, $G = t + \sum_{i=1}^{h-1} G_i x^i \in \mathbb{Z}_q[x]$ zufällig.
publiziert $E_j := g^{F_j} h^{G_j}$ für $j = 1,\ldots,k-1$.
(3) $D$ sendet $(s_i,t_i)$ privat an $P_i$ für $i = 1,\ldots,n$.

**Fakt:** Jede Gruppe $(P_i)_{i\in S}$ mit $S \subset \{1,\ldots,n\}$, $\#S = k$ kann $s,t$ rekonstruieren gemäß

$$\begin{aligned} s &= \sum_{i\in S} a_i s_i \\ t &= \sum_{i\in S} a_i t_i \\ a_i &= \prod_{j\in S, j\neq i} \frac{i}{i-j} \mod q. \end{aligned}$$

Schema ist symmetrisch in $(s,g,F,s_i)(t,h,G,t_i)$.

**7.6.1. Lineares Rechnen mit Geheimnissen.** Seien $s',s'' \in \mathbb{Z}_q$ und verteilt mit

$$(E'_1,\ldots,E'_n) \quad , \quad (E''_1,\ldots,E''_n)$$

bzgl.

$$\begin{aligned} F' &= s' + \sum_{j=1}^{k-1} F'_j x^j, F'' = s'' + \sum_{j=1}^{k-1} F''_j x^j \\ G' &= t' + \sum_{j=1}^{k-1} G'_j x^j, G'' = t'' + \sum_{j=1}^{k-1} G''_j x^j \end{aligned}$$

aus $\mathbb{Z}_q[x]$. $P_1,\ldots,P_n$ können linear verteilt rechnen.

(1) $s := s' + s''$ gemäß $s_i := s'_i + s''_i$ für $i = 1,\ldots,n$ und $E_j := E'_j \cdot E''_j$ für $j = 0,\ldots,k-1$.
(2) $s := as'$ für $a \in \mathbb{Z}_q$ gemäß $s_i := as'_i$ für $i = 1,\ldots,n$ und $E_j := (E'_j)^{\underline{a}}$ für $j = 0,\ldots,k-1$.

$P_1,\ldots,P_n$ können nach jedem Rechenschritt die Konsistenz ihrer Anteile prüfen.

$$P_i \text{ prueft } E(s_i,t_i) \stackrel{?}{=} \prod_{j=0}^{k-1} E_j^{(i^j)} \quad i=1,\ldots,n.$$

Nach Theorem 7.5.4 auf Seite 98 konstruiert jede Gruppe $(P_i)_{i\in S}$ mit $S\subset\{1,\ldots,n\}$, $\#S=k$ dasselbe $s,t$.

## 7.7. Anonymous Secret Sharing

**Aufgabe.** $P_1,\ldots,P_n$ wollen ein Geheimnis erzeugen und verifizierbar aufteilen. $P_i$ habe digitale Signatur, $(P_i,P_j)$ haben privaten Kanal.

Für Dealer $D$ – im Broadcast Modell – sind Signaturen nicht erforderlich. Bisher hat immer $D$ das Geheimnis verteilt – jetzt wird $D$ eliminiert.

**Protokoll für $P_i$.**

(1) Wähle $s_{i0} \in_R \mathbb{Z}_q$.
(2) Verteile $s_{i0}$ verifizierbar an $P_1,\ldots,P_n$.
$P_i$ sendet Signatur mit den Anteilen

$$s_{i,i'}\,,\, t_{i,i'} \quad i'=1,\ldots,n \quad E_{i,j}\; j=0,\ldots,k-1.$$

(3) Jeder Teilnehmer $P_{i'}$ prüft die erhaltenen Anteile $s_{i,i'}\,,\, t_{i,i'}$ gegen $E_{i,0},\ldots,E_{i,k-1}$.
(4) $s := s_{i0}+\ldots+s_{n0}\,,\, t := t_{i0}+\ldots+t_{n,0}$ mit Anteilen

$$s_i = \sum_{i'} s_{i,i'} \quad , \quad t_i = \sum_{i'} t_{i,i'}$$

und zur Verifikation $E_i = \prod_{i'} E_{i,i'}$ .

THEOREM 7.7.1. *Wenn $s_{i,0} \in_R \mathbb{Z}_q$ zufällig ist und nicht mehr als $k-1$ der anderen Teilnehmer kooperieren, ist $s$ zufällig in $\mathbb{Z}_q$.*

BEWEIS. $k-1$ Teilnehmer erhalten keine Information über $s_{i,0}$. □

## 7.8. Allgmeines verteiltes Rechnen

[**Ben-Or et al. 1988**]

**Non-Cryptographic.** Es gibt sichere private Kanäle zwischen je zwei Teilnehmern $P_1,\ldots,P_n$. Z.B. jeder Teilnehmer hat Public-Key-Kryptosystem und Signatur.

**Synchrones Rechenmodell.** Pro Runde sendet jeder Teilnehmer $P_i$ an alle übrigen, sichert die empfangenen Daten und rechnet mit ihnen. Die Teilnehmer müssen dabei alle wissen, wann eine Runde zuende ist.

**Eingabe:** Jeder Teilnehmer $P_i$ hat $x_i \in \mathbb{Z}_q$.
Rechenschritte $+, \cdot$ in $\mathbb{Z}_q$.
**Ausgabe:** $f(x_1, \ldots, x_n) \in \mathbb{Z}_q$.

DEFINITION 7.8.1. Berechnung ist **$t$-privat**, wenn jede Gruppe $(P_i)_{i \in S}$ von $t$ Teilnehmern nach der Berechnung nur Informationen über $(x_i)_{i \in S}$ und $f(x_1, \ldots, x_n)$ hat, sowie Zufallsdaten.

Desweiteren:

DEFINITION 7.8.2. Eine Berechnung ist **$t$-resistent** (**resistent**, **widerstandsfähig**), wenn je $t$ Teilnehmer die Korrektheit der Rechnung nicht verfälschen können. Die Teilnehmer geben ihre Eingaben $x_i$ korrekt oder $f$ wird entsprechend undefiniert.

THEOREM 7.8.3. *Für jede prob. Funktion $f(x_1, \ldots, x_n) : \mathbb{Z}_q^n \to \mathbb{Z}_q$ und $t < n/2$ gibt es ein $t$-privates Berechnungsprotokoll.*

Sei $E$ endlicher Körper, z.B. $E = \mathbb{Z}_q$. $\alpha_0, \ldots, \alpha_{n-1} \in E$ seien verschieden und öffentlich; bei Pederson $\{\alpha_0, \ldots, \alpha_{n-1}\} = \{1, 2, \ldots, n\}$.

**Input Stage.** Jeder Teilnehmer sendet von seiner Eingabe $s$ Anteile an $P_1, \ldots, P_n$. Er wählt $f(x) = s + a_1 x + \ldots + a_t x^t \in \mathbb{Z}_q[x]$ zufällig und sendet $s_j := f(\alpha_j)$ an $P_j$, $j = 1, \ldots, n$. $P_i$ sendet $f_i(\alpha_j)$ an $P_0$ mit $\alpha_i = f_i(0)$.

**Berechnungsschritte.**

**Linear:** $+$, skalare Multiplikation wie bei Pederson.
Jeder Teilnehmer $P_i$ rechnet privat mit seinem Anteil $f(\alpha_i)$ ohne Kommunikation.
**Muliplikationsschritt:** $a, b \mapsto a \cdot b$.
Sei $n \geq 2t + 1$

$$\begin{aligned} f(x) &= a + \sum_{i=1}^{t} f_i x^i \quad \in \mathbb{Z}_q[x] \\ g(x) &= b + \sum_{i=1}^{t} g_i x^i \quad \in \mathbb{Z}_q[x]. \end{aligned}$$

Für $h = f \cdot g$ gilt

$$h(x) = a \cdot b + \sum_{i=1}^{2t} h_i x^i.$$

Wegen $n \geq 2t + 1$ ist $h$ eind. bestimmt durch

$$h(\alpha_i) := f(\alpha_i) \cdot g(\alpha_i) \quad i = 1, \ldots, n.$$

Jeder Teilnehmer $P_i$ multipliziert seine Anteile.
**Problem:** $\text{grad}(h) \leq 2t$, $h$ ist nicht zufällig.

**Grad-Reduktion:** $h(x) \mapsto k(x) := h_0 + h_1 x + \ldots + h_t x^t$.
Die Abbildung $(h(\alpha_1), \ldots, h(\alpha_n)) \mapsto (k(\alpha_1), \ldots, k(\alpha_n))$ ist linear und hängt nur von $\alpha_1, \ldots, \alpha_n$ ab. Die Berechnung der Abbildung $h(x) \mapsto k(x)$ geschieht verteilt, es gehen alle Anteile $h(\alpha_i)$, $i = 1, \ldots, n$ ein. $h$ ist nicht zufällig und muss durch Randomisieren geschützt werden.

**Die Abbildung $h(x) \mapsto k(x)$.**

**Gegeben:** $S = (s_0, \ldots, s_{n-1})$ mit $s_i = h(\alpha_i)$.
**Berechne:** $R = (r_0, \ldots, r_{n-1})$ mit $r_i = k(\alpha_i)$.

CLAIM 7.8.4. Es gibt konstante $A \in M_{n,n}(\mathbb{Z}_q) : R = SA$.

BEWEIS.

$$\begin{aligned} H &:= (h_0, h_1, \ldots, h_{2t}, 0, \ldots, 0) \in \mathbb{Z}_q^n \quad n \geq 2t+1 \\ K &:= (h_0, h_1, \ldots, h_t, 0, \ldots, 0) \in \mathbb{Z}_q^n \end{aligned}$$

Sei $B = [\alpha_i^j]_{0 \leq i,j < n}$ die Vandermonde Matrix.

Sei $P$ die lineare Projektion $P : (x_0, \ldots, x_{n-1}) \mapsto (x_0, \ldots, x_t, 0, \ldots, 0)$ als Matrix

$$P = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \in M_{n,n}(\mathbb{Z}_q).$$

Es gilt

$$\begin{aligned} H \cdot B = S \qquad & H \cdot P = K \\ K \cdot B = R \quad \Rightarrow \quad & B^{-1} P B = R. \end{aligned}$$

Verteilen von $h$ bzw. $k$ bedeutet Multiplikation mit Vandermonde Matrix $B$. □

**Randomisieren.** Jeder Teilnehmer $P_i$ wählt $q_i(x) \in \mathbb{Z}_q[x]$ vom Grad $\leq 2t$ zufällig mit $q_i(0) = 0$ und gibt $q_i(\alpha_j)$ an $P_j$. Sei

$$\tilde{h}(x) = h(x) + \sum_{j=0}^{n-1} q_j(x) \quad , \quad \tilde{h}(0) = h(0).$$

Es wird nicht $h$ sondern $\tilde{h}$ Grad-reduziert $\tilde{h}(x) \mapsto \tilde{k}(x)$. $P_j$ hat den $\tilde{h}$-Anteil

$$\tilde{h}(\alpha_j) = h(\alpha_j) + \sum_{i=1}^{n} q_i(\alpha_j).$$

Dies beweist Theorem 7.8.3.

Die Bedingung $t < n/2$ benötigt man für Multiplikationen. Lineares Rechnen geht $t$-privat mit $t < n$.

Die Anzahl der Runden der verteilten $d$-privaten Berechnung ist die Multiplikations-Tiefe der Berechnung von $F$.

CASE 1. $t \ll n$. Grad-Reduktion ist erst erforderlich, wenn die Grad-Schranke $n-1$ für die Polynome $f(x)$ überschritten wird. Ist $F$ multivariantes Polynom vom Grad $d$ und $td < n$, dann kann man ohne Grad-Reduktion rechnen.

Die Anteile des Ergebnispolynoms $f(\alpha_i)$, $i = 1,\ldots,n$ werden ausgetauscht und jeder kann das Ergebnis $f(0)$ durch Interpolation bilden.

**Byzantinische Spieler (Betrüger).** Es muss geprüft werden, dass jedes $P_i$ ein Polynom $f$ vom Grad $\leq t$ verteilt – andernfalls ist die Eingabe $x_i' = f(0)$ von $P_i$ nicht erklärt. Die Gradprüfung erfolgt durch Interpolation.

Es muss zusätzlich geprüft werden, dass die Eingabe $f(0)$ die korrekte Form hat, z.B. $f(0) \in \{0,1\}$ bei Stimmenzählung.

**$t$-resiliente Berechnung, $t < \frac{n}{3}$.** Es wird ein $t$-fehlerkorrigierender Code benutzt, so dass bis zu $t$ fehlerhafte Anteile $f(\alpha_i)$ korrigiert werden.

Der Körper $E = \mathbb{Z}_q$ habe $n$-te primitive Einheitswurzel $w$ mit

$$w^n = 1 \quad , \quad w^j \neq 1 \text{ fuer } 1 < j < n.$$

Wähle $\alpha_i = w^i$, $i = 0,\ldots,n-1$ und sei

$$\begin{aligned} f(x) &= a_0 + a_1 x + \ldots + a_t x^t \\ A &= (a_0,\ldots,a_{n-1})^T \quad a_{t+1},\ldots,a_{n-1} := 0 \end{aligned}$$

mit Anteilen $f(w^i)$, $i = 1,\ldots,n$.

$$\begin{bmatrix} f(w^0) \\ \vdots \\ f(w^{n-1}) \end{bmatrix} = \left[w^{ij}\right]_{0\leq i,j\leq n} \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix}.$$

Die Abb. $A^T \mapsto (f(w^0),\ldots,f(w^{n-1}))^T$ ist die diskrete Fouriertransformation.

$\hat{f}(x) = s_0 + s_1 x + \ldots + s_{n-1}x^{n-1}$ mit $s_i = f(w^{-ij})$ ist die Fouriertransformierte von $f$.

$$\begin{aligned} \left[w^{ij}\right]^{-1}_{0\leq i,j<n} &= \tfrac{1}{n}\left[w^{-ij}\right]_{0\leq i,j<n} \\ a_i &= \tfrac{1}{n}\hat{f}(w^{-i}) \qquad . \end{aligned}$$

Das Zusammensetzen der Anteile $s_i = f(w^i)$ zu $A = (a_0,\ldots,a_{n-1})$ ist die inverse Fouriertransformation.

$$\begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix} = \frac{1}{n}\left[w^{-ij}\right]_{0\leq i,j<n} \begin{bmatrix} f(w^0) \\ \vdots \\ f(w^{n-1}) \end{bmatrix}$$

Die Anteile $s_i = f(w^i)$, $i = 0,\ldots,n-1$ werden mit einem fehlerkorrigierenden Code berichtigt.

DEFINITION 7.8.5. des Codes in $\mathbb{Z}_q^n$ (**Reed-Miller Code**)

(1) $a_i = \frac{1}{n}\hat{f}(w^{-i}) = 0$ für $i = t+1,\ldots,n-1$.
bzw. $\prod_{i=t+1}^{n-1}(x - w^{-i}) \mid \hat{f}$
bzw. $\sum_{r=0}^{n-1} w^{-ir}s_r = 0$ für $i = t+1,\ldots,n-1$.

Die Gleichungen (1) definieren einen zyklischen Code in $\mathbb{Z}_q^n$ auf $(s_{2,}\ldots,s_{n-1}) \in \mathbb{Z}_q^n$ mit Generatorpolynom

$$g(x) = \prod_{i=t+1}^{n-1} (x - w^{-i}).$$

FACT 7.8.6. $grad(g) = n-1-t \overset{3t<n}{\geq} 2t$

Distanz des Codes $\min\#\{i \mid s_i = 0, s \neq 0\}$.

FACT 7.8.7. *Distanz* $\geq 2t+1$.

BEWEIS. Ang. Dist.$\leq 2t$.

$\exists \hat{f}: \hat{f}(x) = \sum_{i=0}^{n-1} f(w^i)x^i$ hat $\leq 2t$ Koeff. $\neq 0$.

Also $f(w') = 0$ an mindestens $n-2t \overset{3t<n}{\geq} t+1$ Stellen $w^i \Rightarrow f \equiv 0 \Rightarrow \hat{f} \equiv 0$. Widerspruch. □

COROLLARY 7.8.8. *Bis zu t fehlerhafte Anteile* $s_i = f(w')$ *sind korrigierbar.*

Bei linearen Operationen $+$, skal. Mult. rechnet jeder Teilnehmer $P_i$ mit seinen Anteilen $f(w^i), h(w^i)$ linear ohne Kommunikation.

**Multiplikation:** $h := f \cdot g$, $c := a \cdot b$, $a = f(0)$, $b = g(0)$, $c = h(0)$.

Die Grad-Reduktion $h(x) \mapsto k(x)$ erfolgt verteilt. Bis zu $t$ fehlerhafte Anteile $h(w^i)$ werden korrigiert. Für die Grad-Reduktion müssen die Anteile $h(w^i)$ so korrigiert werden, dass sie von der Form

$$h(w^i) = g(w^i) \cdot f(w^i)$$

mit $\mathrm{grad} f$, $\mathrm{grad} g \leq t$ sind. Falls $n \geq 4t+1$, geht das mit dem Reed-Miller Code. Falls $n = 3t+1$ siehe Paper.

THEOREM 7.8.9. *Für jedes prob. Polynom* $F(x_1,\ldots,x_n): \mathbb{Z}_q^n \to \mathbb{Z}_q$ *und* $t < n/3$ *gibt es eine verteilte Berechnung, die t-privat und t-resistent ist.*

## 7.9. Pseudo random Funktionen nach Naor, Reingold

**[GoldwasserGoldreichMicali1985, NaorReingold1997]**

**GGM-Generator.** Sei $G: I_n \to I_{2n}$ perfekter PZG, $I_1 = \{0,1\}^n$, $G(k) = (G_0(k), G_1(k)) \in I_n \times I_n$.

$G_{x_1\cdots x_n}(k) := G_{x_n} G_{x_{n-1}} \cdots G_{x_1}(k)$.

**Ensemble:** $(F_n)_{n\in\mathbb{N}}$, $F_n := \{f_k : I_n \to I_n \mid k \in I_n\}$. $f_k(x) := G_x(k)$. Die Funktion $f_k \in F_n$ hat Tiefe $n$ bzgl. $G$.

**7.9.1. NR-Konstruktion.** Diffie-Hellman **Instanzen Generator (IG)** $IG: 1^n \to (P,Q,g)$.

(1) $P$ Primzahl mit $n$ Bits.
(2) $Q \mid P-1$ prim mit 160 Bits.
(3) $g \in \mathbb{Z}_p^*$ mit Ordnung $Q$
(4) $(P,Q,g)$ uniform verteilt auf diesen Tripeln.

$$F_n := \{ f_{P,Q,g,\vec{a}} \mid \vec{a} \in_R \mathbb{Z}_q^{n+1}, 1^n \underset{IG}{\to} (P,Q,g)\}.$$
$$f_{P,Q,g,\vec{a}} : I_n \to \langle g \rangle \subset \mathbb{Z}_p^*$$
$$\vec{a} = (a_0,\ldots,a_n) \in \mathbb{Z}_Q^{n+1}$$
$$f_{P,Q,g,\vec{a}}(x) =_{def} g^{a_0 \Pi_{x_i=1} a_i}$$

**Decisional Diffie Hellman Annahme (DDH).** Für jeden prob. pol. Zeit Alg. $A$ gilt

$$\begin{aligned} Adv^A(n) := |&Pr[A(P,Q,g,g^a,g^b,g^{ab}) = 1] \\ &-Pr[A(P,Q,g,g^a,g^b,g^c) = 1]| \quad = \quad O(n^{-t}), t > 1. \end{aligned}$$

mit

$$\begin{aligned} q^n \quad &\overset{IG}{\mapsto} \quad (P,Q,g) \\ &\qquad\quad (a,b,c) \in_R \mathbb{Z}_Q^3. \end{aligned}$$

THEOREM 7.9.1. *Jeden $\varepsilon = n^{-t}$ Vorteil bei der DDH-Entscheidung kann man durch Mehrheitsentscheidung in einen $1-\varepsilon$ Vorteil transformieren.*

BEWEIS. Mehrheitsentscheidung mit $m$ unabh. Versuchen liefert nach TU Fehlerwahrscheinlichkeit $\leq \frac{1}{4m\varepsilon^2}$.

setze $m := \varepsilon^{-3} = n^{3t}$. □

In Theorem 7.9.1 kann man $1-\varepsilon$ durch $1-2^{-n}$ ersetzen.

**Hoeffdings Schranke.** Seien $X_1,\ldots,X_m$ unabh. ZV identisch verteilt über $\mathbb{R}$ $X := \frac{1}{m}\sum_{i=1}^m X_i$. Dann gilt für $(i(1),\ldots,i(n')) \in_R [1,m]^{m'}$

$$Pr[\frac{1}{m'}\sum_{h=1}^{m'} X_{i(h)} \geq X + \frac{1}{2}\varepsilon] \leq e^{-2m'(\frac{1}{2}\varepsilon)^2}.$$

Nach Hoeffding, J. Amer. Stat. Assoc. (1963); Motwani, Raghavan 1995, Exercise 4.7; Bernstein, Chernoff-Schranke.

THEOREM 7.9.2. *Unter der DDH-Annahme ist das Ensemble $(F_n)_{n\in\mathbb{N}}$ nach Theorem 7.9.1 pseudozufällig.*

REMARK 7.9.3. Den Wertebereich $\langle g \rangle$ von $F_n$ transformiert man in ein beliebiges $I_{l(n)}$ mittels einer Familie von paarweise unabhängigen Hashfunktionen $h: \langle g \rangle \to I_{l(n)}$.

Z.B. $h: [0,p[ \to [0,2^l[, x \mapsto a+bx \mod q \mod 2^l$.

BEWEIS. (Theorem 7.9.2) Wir wenden die GGM-Konstruktion auf verschiedene PZG'en $G^{(1)},\ldots,G^{(n)}$ an:

$$G_{x_1x_2\ldots x_n}(k) := G^{(n)}_{x_n}G^{(n-1)}_{x_{n-1}}\cdots G^{(1)}_{x_1}(k)$$

$$G^{(i)}(g^b) := G_{P,Q,g,\vec{a},i}(g^b) := (g^b, g^{a_i b}) = (G^{(i)}_0(g^b), G^{(i)}_1(g^b))$$

mit $(P,Q,g)$ gemäß IG, $\vec{a} \in_R \mathbb{Z}_Q^{n+1}$. □

FACT. *Es gilt:*

(1) *Unter der DH-Annahme ist $G^{(i)}$ perfekter PZG.*
(2) $G^{(n)}_{x_n}G^{(n-1)}_{x_{n-1}}\cdots G^{(1)}_{x_1}(g^{a_0}) = f_{P,Q,g,\vec{a}}(x_1\ldots x_n)$.

Damit ist

$$F_n = \left\{ f_{P,Q,g,\vec{a}} \mid \vec{a} \in_R \mathbb{Z}_Q^{n+1},\ 1^n \xrightarrow{IG} (P,Q,g) \right\}$$

pseudozufällig in $\langle g\rangle^{I_n}$.

### 7.10. Alternative Konstruktion unter CDH-Annahme / Faktor-Annahme

Sei $N = P\dot{Q} \in_R B_n$ Blum Zahlen mit $n$ Bits.

$g \in_R QR_N$, $\vec{a} = (a_{1,0}, a_{1,1}, \ldots, a_{n,0}, a_{n,1}) \in_R \mathbb{Z}_N^{2n}$ mit $r \in_R I_n$ und

$$f_{N,g,\vec{a},r}(x) =_{def} (g^{\prod_{i=1}^n a_{i,x_i}} \mod N) \odot r$$

(Skalarprodukt mod 2) und

$$F_n = \{F_{N,g,\vec{a},r} \in I_1^{I_n}\}$$

ist pseudo zufällig unter der CDH (computational DH-Annahme).

Nach Synthesizer-Konstruktion + Goldreich, Levin.

**Bihan, Boneh, Reingold Inf. Proc. 70 (1999), 83-87.** Faktorisierungsannahme für $N \in_R B_n$ impliziert CDH für $N \in_R B_n$.

**Naor, Reingold, Rosen, STOC 2000.** $F_{N,g,\vec{a},r} \in I_{l(n)}^{I_n}$ wie folgt

$f_{N,g,\vec{a},r}(x) = b_1\cdots b_{l(n)}$

$b_i := (g^{2^{i-1}\prod_{i=1}^n a_{i,x_i}} \mod N) \odot r$

$F_n = \{ f_{N,g,\vec{a},r} \in I_{l(n)}^{I_n}\}$ ist pseudozufällig unter der Faktor-Annahme.

Der $x^2 \mod N$-Generator (Blum, Blum, Schob) mit der Konstruktion 7.10 kombiniert.

**Mudde Square PZG.** $N \in_R B_n$, $r \in_R I_n$

$G^{BBS}_{N,r,e}(x) = b_1\cdots b_{l(n)}$ mit $b_i := (x^{2^{i-1}} \mod N) \odot r$.

$G^{BBs}_{N,r,l}$ ist perfekter PZG unter der Faktorisierungsannahme.

Levin, J. Symb. Logic 58 (1993), Knuth, Vol.2, 1998.

KAPITEL 8

# Pseudozufallsgeneratoren

Für dieses Kapitel sei auf [**SchnorrKrypto2.3**] verwiesen, wo eine elektronische Form dieses Kapitels heruntergeladen werden kann.

KAPITEL 9

# Pseudozufallsfunktionen

Für dieses Kapitel sei auf [**SchnorrKrypto2.3**] verwiesen, wo eine elektronische Form dieses Kapitels heruntergeladen werden kann.

# Index

# Literaturverzeichnis


[Beutelspacher et al. 2001] Beutelspacher, Schwenk, Wolfensetter: „ModerneVerfahren der Kryptographie", 4. Auflage, Vieweg-Verlag 2001.

[BellareRogaway1993] M. Bellare, P. Rogaway: „Random oracles are practical: A paradigm for designing efficient protocols", First Annual Conference on Computer and Communications Security, ACM, 1993.

[Ben-Or et al. 1988] M. Ben-Or, S. Goldwasser, A. Widgerson: „Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation", ACM 1988.

[BrickellMcCurley92] Brickell, McCurley: „An Interactive Identification Scheme Based on Discrete Logarithms and Factoring", Journal of Cryptology 5 (1992), LNCS 0473

[ChaumEvertseGraaf1998] Chaum, Evertse, Graaf: „An Improved Protocol for Demonstrating Posession of Discrete Logarithms and Some Generalizations", Eurocrypt 87, LNCS 304, 1988.

[CoppersmithOdlukoSchroeppel1985] D. Coppersmith, A. Odluko, R. Schroeppel: „Discrete Logarithms in $GF(p)$", Algorithmica 1 (1986).

[FeigeFiatShamir1988] U. Feige, A. Fiat, A. Shamir: „Zero-knowledge proofs of identity", Journal of Cryptology 1(2), 1988.

[FiatShamir1986] Fiat, Shamir: „How to Prove Yourself: Practical Solutions to Identification and Signature Problems", Crypto 1986, LNCS 263.

[Girault1991] M. Girault: „Self-Certified Public Keys", Eurocrypt 91, LNCS 547, 1991.

[GiraultPoupardStern1999] M. Girault, G. Poupard, J. Stern: „Global Payment System (GPS): un Protocole de Signature à la Volée", Trusting Electronic Trade (TET'99), 1999.

[GoldwasserGoldreichMicali1985] O. Goldreich, S. Goldwasser, S. Micali: „How to Construct Random Functions", Journal of the ACM No. 4 Vol. 33, 1986.

[Menezes1997] A. Menezes, P. Oorschot, S. Vanstone: „Handbook of applied Cryptography", ACM-Press 1997.

[NaccacheStern1998] D. Naccache, J. Stern: „A new cryptosystem based on higher residues", 5th ACM Symposium on Computer and Communications Security, 1998.

[NaorReingold1997] M. Naor, O. Reingold: „On the Construction of Pseudo-Random Permutations: Luby-Rackoff Revisited", ACM 1997.

[Nechaev1994] V. I. Nechaev: „On the Complexity of a Deterministic Algorithm for a Discrete Logarithm", Mathematical Notes 55, 1994.

[NybergRueppel1993] K. Nyberg, R. Rueppel: „A New Signature Scheme Based on the DSA Giving Message Recovery", ACM 1993.

[Okamoto1992] Okamoto: „Provable Secure and Practical Identification Schemes and Corresponding Signature Schemes", Crypto 92, LNCS 740.

[OkamotoUchiyama1998] T. Okamoto, S. Uchiyama: „An efficient public-key cryptosystem", Advances in Cryptology–EUROCRYPT 98, pages 308-318, 1998.

[OngSchnorr1990] H. Ong, C. Schnorr: „Fast Signature Generation with a Fiat Shamir-like Scheme", Eurocrypt 90, LNCS 473.

[Paillier1999] P. Paillier: „Public-Key Cryptosystems Based on Composite Degree Residuosity Classes", Advances in Cryptology - EUROCRYPT'99, LNCS 1592, 1999.

[Pedersen1991] T. Pedersen: „Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing", Crypto 91, LNCS 576, 1991.

[PointchevalStern1996] Pointcheval, Stern: „Security Profs for Signature Schemes", Advances in Cryptology – Proceedings of EUROCRYPT96, Springer-Verlag, LNCS 1070.

[PoupardStern1998] Poupard, Stern: „Security Analysis of a Practical „on the fly" Authentication and Signature Generation", Eurocrypt 1998, LNCS1403.

[PoupardStern2000] Poupard, Stern: „Fair Encryption of RSA Keys", Eurocrypt 2000.

[Salomaa1996] A. Salomaa: „Public-Key Cryptography", Second Edition, Springer Verlag, 1996.

[SchnorrKrypto1.3] C. P. Schnorr: „Faktorisierungsproblem, Protokolle", Kapitel 3 der Vorlesung „Kryptographie 1" bei Prof. Schnorr, `http://www.mi.informatik.uni-frankfurt.de`

[SchnorrKrypto2.3] C. P. Schnorr: „Pseudozufallsgeneratoren", Kapitel 3 der Vorlesung „Kryptographie 2" bei Prof. Schnorr, `http://www.mi.informatik.uni-frankfurt.de`

[SchnorrKrypto2.4] C. P. Schnorr: „Pseudozufallsfunktionen", Kapitel 4 der Vorlesung „Kryptographie 2" bei Prof. Schnorr, `http://www.mi.informatik.uni-frankfurt.de`

[Schnorr1991] C. P. Schnorr: „Efficient Signature Generation by Smart Cards", Journal of Cryptology 4(3), 1991.

[Schnorr2001] C. P. Schnorr: „Security of DL-Encryption and signatures against generic attacks – a survey", `http://www.mi.informatik.uni-frankfurt.de`

[Schwartz1980] J. T. Schwartz: „Fast probabilistic algorithms for verification of polynomial identities", J. ACM 27(4): 701-717, 1980.

[Shoup1997] V. Shoup: „Lower bounds for discrete logarithms and related problems", Eurocrypt '97, LNCS 1233, pp. 256-266, 1997.

[Shoup1999] V. Shoup: „", LNCS 1070, Journal of Cryptology 12, 1999.

[Stinson1995] D. Stinson: „Cryptogrphy – Theory and Practice", ACM 1995.