

# Kryptologie

Johannes Köbler



Institut für Informatik  
Humboldt-Universität zu Berlin

WS 2020/21

## Eigenschaften von handschriftlichen Signaturen

- Die durch die Unterschrift gekennzeichnete Person hat überprüfbar die Unterschrift geleistet
- Die Unterschrift ist nicht auf ein anderes Dokument übertragbar, ohne ihre Gültigkeit zu verlieren
- Das signierte Dokument kann nachträglich nicht unbemerkt verändert werden

Eine direkte Übertragung dieser Eigenschaften in die digitale Welt ist nicht möglich

## Lösung:

Die digitale Signatur wird nicht physikalisch, sondern logisch (inhaltlich) an ein elektronisches Dokument bzw. Text gebunden und die Fähigkeit, einen individuellen Schriftzug auszuführen, wird durch geheimes Wissen ersetzt

## Definition

Ein **digitales Signaturverfahren** besteht aus

- einer Menge  $X$  von **Texten**
- einer endlichen Menge  $Y$  von **Signaturen**
- einem **Schlüsselraum**  $K$
- einer Menge  $S \subseteq K \times K$  von Schlüsselpaaren  $(\hat{k}, k)$ , bestehend aus einem **Signierschlüssel**  $\hat{k}$  und einem **Verifikationsschlüssel**  $k$
- einem **Signieralgorithmus**  $sig : K \times X \rightarrow Y$  und
- einem **Verifikationsalgorithmus**  $ver : K \times X \times Y \rightarrow \{0, 1\}$ , so dass  $ver(k, x, y) = 1$  für alle Paare  $(\hat{k}, k) \in S$  und  $(x, y) \in X \times Y$  mit  $y = sig(\hat{k}, x)$  gilt

Im Fall  $ver(k, x, y) = 1$  heißt  $y$  **gültige** Signatur für den Text  $x$  (unter  $k$ ), andernfalls **ungültig**

## Digitale Signaturverfahren

- Ein wichtiger Unterschied zu MACs besteht darin, dass digitale Signaturverfahren asymmetrisch sind
- Aufgrund dieser Asymmetrie kann Bob nämlich auch einem Dritten gegenüber nachweisen, dass eine von Alice erzeugte Signatur  $y$  tatsächlich von Alice stammt
- Bei Verwendung eines MACs zur Authentifikation einer Nachricht  $x$  könnte Bob die Nachricht manipuliert und den MAC-Wert auch selbst erzeugt haben, weshalb Alice ihre Urheberschaft von  $x$  erfolgreich abstreiten kann
- Ein weiterer Vorteil von digitalen Signaturen gegenüber MACs ist, dass eine von Alice geleistete Signatur von allen verifizierbar ist, sofern sie den öffentlichen Verifikationsschlüssel von Alice kennen
- Um bspw. die Authentizität eines Software-Updates  $x$  zu gewährleisten, kann eine SW-Firma  $x$  zusammen mit ihrer Signatur  $y$  für  $x$  verschicken
- Bei Verwendung eines MACs müsste die SW-Firma dagegen mit jedem einzelnen Kunden  $K_i$  einen symmetrischen Schlüssel  $k_i$  vereinbaren und den zugehörigen MAC-Wert  $y_i = h_{k_i}(x)$  versenden

**Angriff bei bekanntem Verifikationsschlüssel (key-only attack)**

Dem Angreifer ist nur der öffentliche Verifikationsschlüssel  $k$  bekannt und er versucht, ein Paar  $(x, y)$  mit  $ver(k, x, y) = 1$  zu finden. Jedes solche Paar, das nicht von Alice unter Verwendung des geheimen Signierschlüssels erzeugt wurde, wird als **Fälschung** bezeichnet

**Angriff bei bekannter Signatur (known signature attack)**

Der Angreifer kennt neben  $k$  die Signaturen  $y_i = sig(\hat{k}, x_i)$  für eine Reihe von Texten  $x_1, \dots, x_q$ , auf deren Auswahl er keinen Einfluss hat, und versucht, eine Fälschung  $(x, y)$  mit  $x \notin \{x_1, \dots, x_q\}$  zu finden

**Angriff bei frei wählbaren Texten (chosen document attack)**

Der Angreifer kann die Texte  $x_1, \dots, x_q$  selbst wählen, erhält die Signaturen aber erst, nachdem er alle Texte vorgelegt hat

**Angriff bei adaptiv wählbaren Texten**

Der Angreifer kann die Wahl des Textes  $x_{i+1}$  von den Signaturen  $y_1, \dots, y_i$  abhängig machen

## ***uneingeschränktes Fälschungsvermögen (total break)***

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

## ***selektives Fälschungsvermögen (selective forgery)***

Der Angreifer kann für Texte seiner Wahl die zugehörigen Signaturen bestimmen (eventuell mit Hilfe des legalen Unterzeichners)

## ***nichtselektives (existentielles) Fälschungsvermögen***

Der Angreifer kann für bestimmte Texte  $x$ , auf deren Wahl er keinen Einfluss hat, die zugehörige digitale Signatur bestimmen

# Das RSA-Kryptosystem

- Das RSA-Kryptosystem wurde 1978 von Rivest, Shamir und Adleman veröffentlicht
- Während es beim **Primzahlproblem** nur um die Frage „Ist  $n$  prim?“ geht, muss beim **Faktorisierungsproblem** im Falle einer zusammengesetzten Zahl mindestens ein nicht-trivialer Faktor berechnet werden
- Genauer gesagt beruht das RSA-Verfahren darauf, dass die Primzahleigenschaft zwar effizient getestet werden kann, aber keine effizienten Faktorisierungsalgorithmen bekannt sind

## Schlüsselgenerierung

Für jeden Teilnehmer  $X$  werden zwei Primzahlen  $p, q$  und zwei Exponenten  $e, d$  mit  $ed \equiv_{\varphi(n)} 1$  generiert, wobei  $n = pq$  und  $\varphi(n) = (p-1)(q-1)$  ist

Öffentlicher Schlüssel:  $k_X = (e, n)$

Privater Schlüssel:  $k'_X = (d, n)$

## Ver- und Entschlüsselung

- Jede Nachricht  $x$  wird durch eine Folge  $x_1, x_2, \dots$  von Zahlen  $x_i \in \mathbb{Z}_n$  dargestellt, die einzeln wie folgt ver- und entschlüsselt werden:
  - $\text{RSA}((e, n), x) = x^e \bmod n$
  - $\text{RSA}^{-1}((d, n), y) = y^d \bmod n$
- Der Schlüsselraum ist also
$$K = \{(c, n) \mid \text{es gibt Primzahlen } p \text{ und } q \text{ mit } n = pq \text{ und } c \in \mathbb{Z}_{\varphi(n)}^*\}$$
und
$$S = \{((e, n), (d, n)) \in K \times K \mid ed \equiv_{\varphi(n)} 1\}$$
ist die Menge aller zueinander passenden Schlüsselpaare
- Die Chiffrierfunktionen  $\text{RSA}_{(e,n)}$  und  $\text{RSA}_{(d,n)}^{-1}$  sind durch **Wiederholtes Quadrieren und Multiplizieren** effizient berechenbar



## Ver- und Entschlüsselung

Der folgende Satz garantiert die Korrektheit des RSA-Systems

### Satz

Für jedes Schlüsselpaar  $((e, n), (d, n)) \in S$  und alle  $x \in \mathbb{Z}_n$  gilt

$$x^{ed} \equiv_n x$$

### Beweis.

- Sei  $n = pq$  und sei  $z$  eine natürliche Zahl mit  $ed = z\varphi(n) + 1$
- Wir zeigen  $x^{ed} \equiv_p x$ . Die Kongruenz  $x^{ed} \equiv_q x$  folgt analog und beide Kongruenzen zusammen implizieren  $x^{ed} \equiv_n x$
- Wegen  $\varphi(n) = (p-1)(q-1)$  und wegen  $x^{p-1} \equiv_p 1$  für  $x \not\equiv_p 0$  folgt

$$x^{ed} = x^{z\varphi(n)+1} = x^{z(p-1)(q-1)} x = (x^{p-1})^{z(q-1)} x \equiv_p x$$



# Das RSA-Signaturverfahren

## Definition

- Beim ***RSA-Signaturverfahren*** ist

$$K = \{(a, n) \mid n = pq \text{ für Primzahlen } p, q \text{ und } a \in \mathbb{Z}_{\varphi(n)}^*\}$$

und  $S$  die Relation  $S = \{((d, n), (e, n)) \in K \times K \mid de \equiv_{\varphi(n)} 1\}$

- Signiert wird mittels  $\text{sig}(d, n, x) := x^d \bmod n$ , wobei  $X = Y = \mathbb{Z}_n$  ist
- Die Verifikationsbedingung ist

$$\text{ver}(e, n, x, y) = \begin{cases} 1, & y^e \equiv_n x \\ 0, & \text{sonst} \end{cases}$$

## Satz

Für alle  $((d, n), (e, n)) \in S$  und  $x, y \in \mathbb{Z}_n$  gilt

$$\text{ver}(e, n, x, y) = \begin{cases} 1, & \text{sig}(d, n, x) = y, \\ 0, & \text{sonst} \end{cases}$$

Der Beweis folgt direkt aus der Korrektheit des RSA-Kryptosystems

- Wir betrachten eine Reihe von Angriffen gegen das RSA-Signaturverfahren und überlegen anschließend, durch welche Maßnahmen sich diese abwehren lassen
- Ein Angreifer kann leicht eine **existentielle Fälschung bei bekanntem Verifikationsschlüssel** erhalten, indem er zu einer beliebigen Signatur  $y \in Y$  den Text  $x = y^e \bmod n$  wählt
- Zudem ist eine **existentielle Fälschung bei bekannten Signaturen** möglich, falls der Angreifer zwei signierte Texte  $(x_1, y_1), (x_2, y_2)$  mit  $\text{ver}(k, x_i, y_i) = 1$  kennt
- Wegen  $y_i^e \equiv_n x_i$  für  $i = 1, 2$  folgt nämlich  $(y_1 y_2)^e \equiv_n y_1^e y_2^e \equiv_n x_1 x_2$  und somit  $\text{ver}(k, x_1 x_2 \bmod n, y_1 y_2 \bmod n) = 1$
- Weiterhin ist eine **selektive Fälschung bei frei wählbarem Text** möglich
- Kennt der Angreifer nämlich bereits die Signatur  $y'$  für einen beliebigen Text  $x' \in \mathbb{Z}_n^*$  und kann er sich die Signatur  $y''$  für  $x'' = x x'^{-1} \bmod n$  beschaffen, so kann er daraus die Signatur  $y = y' y'' \bmod n$  für den Text  $x$  berechnen

- Diese Angriffe kann man vereiteln, indem man den Text  $x$  mit **Redundanz** versieht (indem man z.B. anstelle von  $x$  den Text  $xx$  signiert)
- Um auch längere Texte effizient signieren zu können, wird i.a. jedoch eine geeignete Hashfunktion  $h$  benutzt und nicht der gesamte Text  $x$ , sondern nur der Hashwert  $h(x)$  signiert

Bei der Signaturerstellung benötigte Eigenschaften einer Hashfunktion  $h$

- Die verwendete Hashfunktion  $h$  sollte die **Einwegeigenschaft** haben, da sonst der Angreifer zu einem  $y \in Y$  einen passenden Text  $x$  mit  $h(x) = y$  bestimmen kann (zumindest wenn das Signaturverfahren anfällig gegen eine existentielle Fälschung ist, wie etwa RSA)
- Angenommen der Angreifer kennt bereits ein Paar  $(x, y)$  mit  $ver(k, h(x), y) = 1$
- Dann sollte  $h$  zumindest **schwach kollisionsresistent** sein, da sonst der Angreifer ein  $x'$  mit  $h(x') = h(x)$  berechnen und das Paar  $(x', y)$  bestimmen könnte
- Falls sich der Angreifer für bestimmte von ihm selbst gewählte Texte  $x$  die zugehörige Signatur  $y$  beschaffen kann, so sollte  $h$  sogar **kollisionsresistent** sein
- Andernfalls könnte der Angreifer ein Kollisionspaar  $(x, x')$  für  $h$  finden, sich den (unverdächtigen) Text  $x$  signieren lassen und die erhaltene Signatur  $y$  für den Text  $x'$  verwenden

- Für ein beliebiges Element  $a$  einer multiplikativen Gruppe  $G$  ist die **Exponentiation**  $\exp_{G,a} : x \mapsto a^x$  zur **Basis**  $a$  eine Bijektion zwischen der Menge  $\mathbb{Z}_{\text{ord}(a)} = \{0, 1, \dots, \text{ord}(a) - 1\}$  und der Untergruppe  $\langle a \rangle$
- Die zugehörige Umkehrabbildung spielt in der Kryptografie eine wichtige Rolle

### Definition

- Seien  $a, b \in G$  mit  $b \in \langle a \rangle$
- Dann heißt der eindeutig bestimmte Exponent  $x \in \mathbb{Z}_{\text{ord}(a)}$  mit  $a^x = b$  **Index** oder **diskreter Logarithmus von  $b$  zur Basis  $a$  in  $G$** , kurz

$$x = \log_{G,a}(b)$$

- Im Fall  $G = \mathbb{Z}_m^*$  schreiben wir auch einfach  $\log_{m,a}(b)$  anstelle von  $\log_{\mathbb{Z}_m^*,a}(b)$

- Die Funktion  $\exp_{m,a} : x \mapsto a^x$  ist effizient berechenbar (siehe unten)
- Dagegen sind bis heute keine effizienten Verfahren zur Berechnung von  $\log_{m,a}(b)$  bekannt (falls  $a$  und  $m$  geeignet gewählt werden)

### Beispiel

- Das Element  $a = 2$  hat in der Gruppe  $G = \mathbb{Z}_{11}^*$  die maximal mögliche Ordnung  $\text{ord}_{11}(2) = \|G\| = 10$
- Die folgenden Tabellen zeigen den Werteverlauf der Funktionen  $\exp_{11,2}$  und  $\log_{11,2}$

|       |   |   |   |   |   |    |   |   |   |   |
|-------|---|---|---|---|---|----|---|---|---|---|
| $x$   | 0 | 1 | 2 | 3 | 4 | 5  | 6 | 7 | 8 | 9 |
| $2^x$ | 1 | 2 | 4 | 8 | 5 | 10 | 9 | 7 | 3 | 6 |

|                  |   |   |   |   |   |   |   |   |   |    |
|------------------|---|---|---|---|---|---|---|---|---|----|
| $b$              | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $\log_{11,2}(b)$ | 0 | 1 | 8 | 2 | 4 | 9 | 7 | 3 | 6 | 5  |

# Zyklische Gruppen

Für manche Anwendungen sind Elemente  $a \in G$  nützlich, mit denen sich die gesamte Gruppe erzeugen lässt

## Definition

- Sei  $G$  eine endliche Gruppe der Ordnung  $\|G\| = m$
- Ein Element  $g \in G$  mit  $\text{ord}_G(g) = m$  heißt **Erzeuger** von  $G$
- $G$  heißt **zyklisch**, falls  $G$  mindestens einen Erzeuger besitzt

Ein Element  $a \in G$  ist also genau dann ein Erzeuger, wenn die von  $a$  erzeugte Untergruppe  $\langle a \rangle$  die gesamte Gruppe  $G$  umfasst

## Satz (Gauß)

Genau für  $m \in \{1, 2, 4, p^k, 2p^k \mid 2 < p \text{ prim}\}$  ist die Gruppe  $\mathbb{Z}_m^*$  zyklisch (ohne Beweis)



- Das **Signaturverfahren von ElGamal** (1985) ist wie das gleichnamige asymmetrische Kryptosystem probabilistisch und beruht wie dieses auf dem diskreten Logarithmus
- Sei  $p$  eine große Primzahl und  $\alpha$  ein Erzeuger von  $\mathbb{Z}_p^*$  ( $p$  und  $\alpha$  sind öffentlich)
- Jeder Teilnehmer  $B$  wählt eine geheime Zahl  $a \in \mathbb{Z}_{p-1} = \{0, \dots, p-2\}$  und gibt  $\beta = \alpha^a \bmod p$  als Teil seines öffentlichen Verifikationsschlüssels bekannt:  
Signierschlüssel:  $\hat{k} = (p, \alpha, a)$   
Verifikationsschlüssel:  $k = (p, \alpha, \beta)$
- Der **Textraum** ist  $X = \mathbb{Z}_{p-1}$  und der **Signaturenraum** ist  $Y = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1} \setminus \{0\}$

# Das ElGamal-Signaturverfahren

- **Signaturerstellung:** Um einen Text  $x \in X$  zu signieren, wählt der Signierer zufällig eine Zahl  $z \in \mathbb{Z}_{p-1}^*$  und berechnet die Signatur

$$\text{sig}(\hat{k}, x, z) = (\gamma, \delta) \in Y$$

mit  $\gamma = \alpha^z \bmod p$  und  $\delta = (x - a\gamma)z^{-1} \bmod p - 1$

- Falls  $\delta = 0$  ist, muss eine neue Zufallszahl  $z$  gewählt und der Vorgang wiederholt werden
- **Verifikation:** Es gilt  $\text{ver}(k, x, (\gamma, \delta)) = 1$ , falls  $\beta^\gamma \gamma^\delta \equiv_p \alpha^x$  ist

# Das ElGamal-Signaturverfahren

## Lemma

Eine Signatur  $(\gamma, \delta)$  mit  $\text{ord}(\gamma) = p - 1$  erfüllt genau dann die Verifikationsbedingung  $\beta^\gamma \gamma^\delta \equiv_p \alpha^x$ , wenn es ein  $z \in \mathbb{Z}_{p-1}^*$  mit  $\text{sig}(\hat{k}, x, z) = (\gamma, \delta)$  gibt

## Beweis.

- Wegen  $\gamma \equiv \alpha^z \pmod{p}$  ist  $z$  durch  $\gamma$  (und  $\gamma$  durch  $z$ ) eindeutig bestimmt
- Weiter ist  $\beta^\gamma \gamma^\delta \equiv_p \alpha^{a\gamma} \alpha^{z\delta} \equiv_p \alpha^{a\gamma+z\delta}$
- Da  $\alpha$  ein Erzeuger von  $\mathbb{Z}_p^*$  ist, gilt die Kongruenz  $\alpha^{a\gamma+z\delta} \equiv_p \alpha^x$  genau dann, wenn  $a\gamma + z\delta \equiv_{p-1} x$  ist, was wiederum mit  $\delta \equiv_{p-1} (x - a\gamma)z^{-1}$  äquivalent ist □

## Bemerkung

Da der Signieralgorithmus für die Berechnung von  $\gamma = \alpha^z \pmod{p}$  eine Zufallszahl  $z \in \mathbb{Z}_{p-1}^*$  wählt, hat jedes von  $\text{sig}$  erzeugte  $\gamma$  die Ordnung  $\text{ord}(\gamma) = \text{ord}(\alpha^z) = \text{ord}(\alpha) / \text{ggT}(\text{ord}(\alpha), z) = \text{ord}(\alpha) = p - 1$

# Das ElGamal-Signaturverfahren

## Beispiel

- Sei  $p = 467$ ,  $\alpha = 2$ ,  $a = 127$  und  $\beta = \alpha^a \bmod p = 2^{127} \bmod 467 = 132$
- Um den Text  $x = 100 \in \mathbb{Z}_{p-1} = \mathbb{Z}_{466}$  mit dem Signierschlüssel  $\hat{k} = (p, \alpha, a) = (467, 2, 127)$  zu signieren,

- wählt Alice die geheime Zufallszahl  $z = 213 \in \mathbb{Z}_{p-1}^*$   
( $\rightsquigarrow z^{-1} \bmod 466 = 431$ ) und
- erhält

$$\gamma = 2^{213} \bmod 467 = 29 \text{ und } \delta = (100 - 127 \cdot 29) 431 \bmod 466 = 51,$$

$$\text{d.h. } \text{sig}(\hat{k}, x, z) = (29, 51)$$

- Um die Gültigkeit dieser Signatur für den Text  $x = 100$  mit dem Verifikationsschlüssel  $k = (p, \alpha, \beta) = (467, 2, 132)$  zu prüfen,
  - verifiziert Bob die Kongruenz

$$\beta^\gamma \gamma^\delta \equiv_p 132^{29} 29^{51} \equiv_p 189 \equiv_p 2^{100} \equiv_p \alpha^x$$

- Falls der Angreifer in der Gruppe  $\mathbb{Z}_p^*$  den diskreten Logarithmus von  $\beta$  zur Basis  $\alpha$  bestimmen kann, so kann er den geheimen Schlüssel  $a = \log_\alpha \beta$  berechnen
- Als nächstes betrachten wir verschiedene Szenarien für einen **selektiven Angriff** bei bekanntem Verifikationsschlüssel
- Der Angreifer wählt zu einem gegebenen Text  $x$  zuerst  $\gamma$  und versucht, ein passendes  $\delta$  zu finden:
  - Mit  $\alpha^x \equiv \beta^\gamma \gamma^\delta \pmod{p}$  folgt  $\delta = \log_\gamma(\alpha^x \beta^{-\gamma})$
  - D.h. die Bestimmung von  $\delta$  ist eine Instanz des **diskreten Logarithmus Problems** (kurz: **DLP**)
- Der Angreifer wählt zu einem gegebenen Text  $x$  zuerst  $\delta$  und versucht dann ein  $\gamma$  mit  $\alpha^x \equiv \beta^\gamma \gamma^\delta \pmod{p}$  zu finden
  - Hierfür ist kein effizientes Verfahren bekannt

- Der Angreifer versucht, zu einem gegebenen Text  $x$  gleichzeitig passende Zahlen  $\gamma$  und  $\delta$  mit  $\alpha^x \equiv \beta^\gamma \gamma^\delta \pmod{p}$  zu finden
  - Auch hierfür ist kein effizientes Verfahren bekannt
- Versucht der Angreifer bei einem **nichtselektiven Angriff**, zuerst  $\gamma$  und  $\delta$  zu wählen und dazu einen passenden Text  $x$  zu finden, so muss er den diskreten Logarithmus  $x = \log_\alpha \beta^\gamma \gamma^\delta$  bestimmen

- Eine **existentielle Fälschung** lässt sich jedoch wie folgt durchführen (falls keine Hashfunktion benutzt wird)
  - Der Angreifer wählt beliebige Zahlen  $u \in \mathbb{Z}_{p-1}$ ,  $v \in \mathbb{Z}_{p-1}^*$  und berechnet  $\gamma = \alpha^u \beta^v \bmod p$
  - Dann ist  $(\gamma, \delta)$  genau dann eine gültige Signatur für einen Text  $x$ , wenn  $\alpha^x \equiv_p \beta^\gamma (\alpha^u \beta^v)^\delta$  ist
  - Dies ist wiederum äquivalent zur Kongruenz  $\alpha^{x-u\delta} \equiv_p \beta^{\gamma+v\delta}$ , die sich im Fall  $\text{ggT}(v, p-1) = 1$  für den Text  $x = u\delta \bmod p-1$  mittels  $\delta = -\gamma v^{-1} \bmod p-1$  erfüllen lässt
  - Bei Wahl von  $v = 1$  erhalten wir z.B. die gültige Signatur  $(\gamma, \delta) = (\alpha^u \beta \bmod p, -\alpha^u \beta \bmod p-1)$  für den Text  $x = u\delta \bmod p-1$ , wobei  $u \in \mathbb{Z}_{p-1}$  beliebig gewählt werden kann

## Bemerkung

Bei der Benutzung des ElGamal-Signaturverfahrens sind folgende Punkte zu beachten

- Die Zufallszahl  $z$  muss geheim gehalten werden
- Zufallszahlen dürfen nicht mehrfach verwendet werden
- Kennt nämlich der Angreifer zu einer Signatur  $(x, (\gamma, \delta))$  die Zufallszahl  $z$ , so kann er wegen  $\delta \equiv_{p-1} (x - a\gamma)z^{-1}$  im Fall  $\text{ggT}(\gamma, p-1) = 1$  die geheime Zahl

$$a = (x - z\delta)\gamma^{-1} \text{ mod } (p-1)$$

als eindeutige Lösung der Kongruenz

$$\gamma a \equiv_{p-1} x - z\delta \quad (*)$$

berechnen



- Kennt nämlich der Angreifer zu einer Signatur  $(x, (\gamma, \delta))$  die Zufallszahl  $z$ , so kann er die geheime Zahl  $a$  als eindeutige Lösung der Kongruenz

$$\gamma a \equiv_{p-1} x - z\delta \quad (*)$$

berechnen

- Ist allgemeiner  $ggT(\gamma, p-1) = g \geq 1$ , so ist  $g$  ein Teiler von  $\gamma$  und von  $p-1$  sowie wegen  $(*)$  auch von  $x - z\delta$
- Setzen wir  $\mu := \gamma/g$  und  $\lambda := (x - z\delta)/g$ , so führt  $(*)$  auf die Kongruenz  $\mu a \equiv_{(p-1)/g} \lambda \quad (**)$ , aus der sich wegen  $ggT(\mu, (p-1)/g) = 1$  folgende  $g$  Kandidaten  $a_i$  für  $a$  gewinnen lassen:  
$$a_0 := \mu^{-1} \lambda \bmod (p-1)/g \text{ und } a_i := a_0 + i(p-1)/g \text{ für } i = 1, \dots, g-1$$
- Unter  $a_0, \dots, a_{g-1}$  lässt sich  $a$  durch Prüfen der Bedingung  $\alpha^{a_i} \equiv_p \beta$  eindeutig bestimmen

- Sind andererseits  $(x_1, (\gamma, \delta_1))$  und  $(x_2, (\gamma, \delta_2))$  mit demselben  $z$  generierte Signaturen, dann folgt wegen  $\beta^\gamma \gamma^{\delta_i} \equiv_p \alpha^{x_i}$  für  $i \in \{1, 2\}$ ,

$$\begin{aligned}\gamma^{\delta_1 - \delta_2} &\equiv_p \alpha^{x_1 - x_2} &\Rightarrow & \alpha^{z(\delta_1 - \delta_2)} \equiv_p \alpha^{x_1 - x_2} \\ & &\Rightarrow & z(\delta_1 - \delta_2) \equiv_{p-1} x_1 - x_2\end{aligned}$$

- Aus dieser Kongruenz lassen sich  $d = \text{ggT}(\delta_1 - \delta_2, p - 1)$  Kandidaten für  $z$  gewinnen und daraus wie oben  $a$  berechnen, falls  $d$  nicht zu groß ist

- Da die Primzahl  $p$  beim ElGamal-Signaturverfahren mindestens eine 512-Bit-Zahl (besser 1024-Bit-Zahl) sein sollte, beträgt die Signaturlänge 1024 bzw 2048 Bit
- Folgende Variante des ElGamal-Signaturverfahrens, die als eine Vorstufe zum DSA betrachtet werden kann, wurde von Schnorr vorgeschlagen
- Die zugrunde liegende Idee ist folgende:
  - Indem wir für  $\alpha$  ein Element der Ordnung  $q$  mit  $q \approx 2^{160}$  wählen, reduziert sich die Signaturlänge auf  $2 \cdot 160 = 320$  Bit
  - Die Berechnungen werden aber nach wie vor modulo  $p$  mit  $p \approx 2^{1024}$  ausgeführt, so dass das Problem des diskreten Logarithmus zur Basis  $\alpha$  in  $\mathbb{Z}_p^*$  hart bleibt

# Das Schnorr-Signaturverfahren

- Sei  $g$  ein Erzeuger von  $\mathbb{Z}_p^*$ , wobei  $p$  die Bauart  $p - 1 = mq$  für eine Primzahl  $q = \frac{p-1}{m} \approx 2^{160}$  hat
- Dann ist  $\alpha = g^{(p-1)/q}$  ein Element in  $\mathbb{Z}_p^*$  der Ordnung  $\text{ord}_p(\alpha) = q$ 
  - da  $\text{ord}(g^i) = \frac{\text{ord}(g)}{\text{ggT}(i, \text{ord}(g))} = \frac{p-1}{\text{ggT}((p-1)/q, p-1)} = q$  ist (s. Übungen)
- Weiter sei  $h : \{0, 1\}^* \rightarrow \mathbb{Z}_q$  eine Hashfunktion, die jedem Text  $x \in X = \{0, 1\}^*$  einen Hashwert in  $\mathbb{Z}_q$  zuordnet
- Das Schnorr-Verfahren benutzt folgende Schlüssel:
  - Signierschlüssel:  $\hat{k} = (p, q, \alpha, a), a \in \mathbb{Z}_q$
  - Verifikationsschlüssel:  $k = (p, \alpha, \beta), \beta = \alpha^a \text{ mod } p$

# Das Schnorr-Signaturverfahren

- Das Schnorr-Verfahren benutzt folgende Schlüssel:

Signierschlüssel:  $\hat{k} = (p, q, \alpha, a)$ ,  $a \in \mathbb{Z}_q$

Verifikationsschlüssel:  $k = (p, \alpha, \beta)$ ,  $\beta = \alpha^a \bmod p$

- Signaturerstellung**

Um einen Text  $x \in X$  zu signieren, wählt der Signierer zufällig eine geheime Zahl  $z \in \mathbb{Z}_q^*$  und berechnet die Signatur

$$\text{sig}(\hat{k}, x, z) = (\gamma, \delta),$$

wobei  $\gamma = h(\text{xbin}(\alpha^z \bmod p))$  und  $\delta = (z + a\gamma) \bmod q$  ist

- Der Signaturraum ist also  $Y := \mathbb{Z}_q \times \mathbb{Z}_q$

- Verifikation**

Es gilt  $\text{ver}(k, \gamma, \delta) = 1$ , falls  $h(\text{xbin}(\alpha^\delta \beta^{-\gamma} \bmod p)) = \gamma$  ist