Kryptologie

Johannes Köbler

Institut für Informatik
Humboldt-Universität zu Berlin

WS 2020/21

Digitale Signaturverfahren

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

Digitale Signaturverfahren

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 $sig : K \times X \to Y$ und
- einem Verifikationsalgorithmus $ver : K \times X \times Y \to \{0, 1\}$, 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

Klassifikation von Angriffen gegen Signaturverfahren

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

Erfolgskriterien für die Fälschung digitaler Signaturen

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 $x_1, x_2, \ldots$ von Zahlen $x_i \in \mathbb{Z}_n$ dargestellt, die einzeln wie folgt ver- und entschlüsselt werden:
  - $\mathrm{RSA}((e,n),x) = x^e \bmod n$
  - $\mathrm{RSA}^{-1}((d,n),y) = y^d \bmod n$
- 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 $\mathrm{RSA}_{(e,n)}$ und $\mathrm{RSA}^{-1}_{(d,n)}$ sind durch Wiederholtes Quadrieren und Multiplizieren effizient berechenbar

Das RSA-Kryptosystem

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

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

$\square$

Das RSA-Signaturverfahren

Definition

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

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

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

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

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

Satz

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

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

Der Beweis folgt direkt aus der Korrektheit des RSA-Kryptosystems

Das RSA-Signaturverfahren

- 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 $ver(k, x_i, y_i) = 1$ 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 $ver(k, x_1 x_2 \bmod n, y_1 y_2 \bmod n) = 1$
- 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 $x'' = xx'^{-1} \bmod n$ beschaffen, so kann er daraus die Signatur $y = y'y'' \bmod n$ für den Text x berechnen

Das RSA-Signaturverfahren

- 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

Das RSA-Signaturverfahren

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 Kollisonspaar (x, x') für h finden, sich den (unverdächtigen) Text x signieren lassen und die erhaltene Signatur y für den Text x' verwenden

Diskrete Logarithmen

- 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 $\mathbb{Z}_{ord(a)} = \{0, 1, \ldots, ord(a) - 1\}$ 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 $x \in \mathbb{Z}_{\mathrm{ord}(a)}$ 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 $\log_{\mathbb{Z}_m^*,a}(b)$

Diskrete Logarithmen

- 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 $\mathrm{ord}_G(g) = m$ 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 ElGamal-Signaturverfahren

- 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 α ein Erzeuger von $\mathbb{Z}_p^*$ (p und α 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

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

  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 $ver(k, x, (\gamma, \delta)) = 1$, falls $\beta^\gamma \gamma^\delta \equiv_p \alpha^x$ ist

Das ElGamal-Signaturverfahren

Lemma

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

Beweis.

- Wegen $\gamma \equiv \alpha^z \bmod p$ ist z durch γ (und γ 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 α 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 $\gamma = \alpha^z \bmod p$ eine Zufallszahl $z \in \mathbb{Z}_{p-1}^*$ wählt, hat jedes von *sig* erzeugte γ die Ordnung $\mathrm{ord}(\gamma) = \mathrm{ord}(\alpha^z) = \mathrm{ord}(\alpha) / \operatorname{ggT}(\mathrm{ord}(\alpha), z) = \mathrm{ord}(\alpha) = p - 1$

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

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

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

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

◁

Zur Sicherheit des ElGamal-Systems

- Falls der Angreifer in der Gruppe $\mathbb{Z}_p^*$ den diskreten Logarithmus von β zur Basis α 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 γ und versucht, ein passendes δ zu finden:
  - Mit $\alpha^x \equiv \beta^\gamma \gamma^\delta \bmod p$ folgt $\delta = \log_\gamma(\alpha^x \beta^{-\gamma})$
  - D.h. die Bestimmung von δ ist eine Instanz des diskreten Logarithmus Problems (kurz: DLP)
- Der Angreifer wählt zu einem gegebenen Text x zuerst δ und versucht dann ein γ mit $\alpha^x \equiv \beta^\gamma \gamma^\delta \bmod p$ zu finden
  - Hierfür ist kein effizientes Verfahren bekannt

Zur Sicherheit des ElGamal-Systems

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

Zur Sicherheit des ElGamal-Systems

- 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 (γ, δ) 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 $\mathrm{ggT}(v, p-1) = 1$ 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

Zur Sicherheit des ElGamal-Systems

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 $\mathrm{ggT}(\gamma, p-1) = 1$ die geheime Zahl

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

  als eindeutige Lösung der Kongruenz

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

  berechnen

Zur Sicherheit des ElGamal-Systems

- 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

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

  berechnen
- Ist allgemeiner $ggT(\gamma, p-1) = g \geq 1$, so ist g ein Teiler von γ 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 $\mu a \equiv_{(p-1)/g} \lambda$ $(**)$, aus der sich wegen $ggT(\mu, (p-1)/g) = 1$ folgende g Kandidaten a_i für a gewinnen lassen:

  $$a_0 := \mu^{-1}\lambda \bmod (p-1)/g \text{ und } a_i := a_0 + i(p-1)/g \text{ für } i = 1, \ldots, g-1$$
- Unter $a_0, \ldots, a_{g-1}$ lässt sich a durch Prüfen der Bedingung $\alpha^{a_i} \equiv_p \beta$ eindeutig bestimmen

Zur Sicherheit des ElGamal-Systems

- 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 $i \in \{1, 2\}$,

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

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

Das Schnorr-Signaturverfahren

- 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 α 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 α 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 $ord_p(\alpha) = q$
  - da $ord(g^i) = \frac{ord(g)}{\mathrm{ggT}(i, ord(g))} = \frac{p-1}{\mathrm{ggT}((p-1)/q, p-1)} = q$ ist (s. Übungen)
- Weiter sei $h : \{0,1\}^* \to \mathbb{Z}_q$ 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: $k = (p, \alpha, \beta)$, $\beta = \alpha^a \bmod p$

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 $z \in \mathbb{Z}_q^*$ (ElGamal: $z \in \mathbb{Z}_{p-1}^*$) und berechnet die Signatur

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

  wobei $\gamma = h(xbin(\alpha^z \bmod p))$ und $\delta = (z + a\gamma) \bmod q$
  (ElGamal: $\gamma = \alpha^z \bmod p$ und $\delta = (x - a\gamma)z^{-1} \bmod p - 1$) ist
- Der Signaturraum ist also $Y := \mathbb{Z}_q \times \mathbb{Z}_q$
- Verifikation
  Es gilt $ver(k, x, \gamma, \delta) = 1$, falls $h(xbin(\alpha^\delta \beta^{-\gamma} \bmod p)) = \gamma$
  (ElGamal: $\beta^\gamma \gamma^\delta \equiv_p \alpha^x$) ist

Das Schnorr-Signaturverfahren

Beispiel

- Seien $q = 101$, $p = 78q + 1 = 7879$ und $g = 3$
- Dann ergibt sich α zu $\alpha = g^{(p-1)/q} = 3^{78} \bmod p = 170$
- Für $a = 75$ ergibt sich β zu $\beta = \alpha^a \bmod p = 170^{75} \bmod 7879 = 4567$
- Um einen Text $x \in \{0,1\}^*$ mit dem Signierschlüssel $\hat{k} = (p, q, \alpha, a) = (7879, 101, 170, 75)$ zu signieren,
  - wählt Alice die geheime Zufallszahl $z = 50 \in \mathbb{Z}_q^*$ und
  - berechnet den Wert $\alpha^z \bmod p = 170^{50} \bmod 7879 = 2518$
  - Dies führt auf den Hashwert $\gamma = h(x\mathit{bin}(2518)) \in \mathbb{Z}_q$
  - Unter der Annahme, dass $h(x\mathit{bin}(2518)) = 96$ ist, erhält Alice wegen

    $$\delta = 50 + 75 \cdot 96 \bmod 101 = 79$$

    die Signatur $\mathit{sig}(\hat{k}, x, z) = (96, 79)$

Beispiel (Fortsetzung)

- Um die Gültigkeit der Signatur $sig(\hat{k}, x, z) = (96, 79)$ für den Text x mit dem Verifikationsschlüssel $k = (p, \alpha, \beta) = (7879, 170, 4567)$ zu prüfen,
  - berechnet Bob die Zahl

    $$\alpha^{\delta}\beta^{-\gamma} \equiv_p 170^{79} 4567^{-96} \equiv_p 2518$$

  - und verifiziert die Gleichheit $h(x\,bin(2518)) = 96$ ◁

Der Digital Signature Algorithm (DSA)

- Der DSA wurde im August 1991 vom National Institute of Standards and Technology (NIST) für die Verwendung im Digital Signature Standard (DSS) empfohlen
- Der DSS enthält neben dem DSA (ursprünglich der einzige im DSS definierte Algorithmus) als weitere Algorithmen die RSA-Signatur und ECDSA (siehe unten)
- Der DSA lässt sich durch eine Reihe von Modifikationen aus dem ElGamal-Verfahren erhalten, das wie folgt arbeitet

Der Digital Signature Algorithm (DSA)

- ElGamal-Verfahren:
  - 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
  $$sig(\hat{k}, x, z) = (\gamma, \delta) \in Y$$
  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 $ver(k, x, (\gamma, \delta)) = 1$, falls $\beta^\gamma \gamma^\delta \equiv_p \alpha^x$ ist
- Folge der Modifikationen für den Übergang zu DSA:
  - δ als Lösung von $z\delta - a\gamma \equiv_{p-1} x$ (d.h. $\delta = (x + a\gamma)z^{-1}$)
  - Dies führt auf die Verifikationsbedingung $\alpha^x \beta^\gamma \equiv_p \gamma^\delta$ $(\alpha^x \alpha^{a\gamma} \equiv_p \alpha^{z(x+a\gamma)z^{-1}})$
  - Ist $x + a\gamma \in \mathbb{Z}_{p-1}^*$, dann existiert $\delta^{-1} = (x + a\gamma)^{-1} z \bmod p - 1$
  - Dies führt auf die Verifikationsbedingung $\alpha^{x\delta^{-1}} \beta^{\gamma\delta^{-1}} \equiv_p \gamma$

Der Digital Signature Algorithm (DSA)

- Sei nun wie bei Schnorr $p = mq + 1$ mit $q \approx 2^{160}$ prim und sei $\alpha \in \mathbb{Z}_p^*$ mit $ord_p(\alpha) = q$
- Dann kann bei der Verifikation von $\alpha^{x\delta^{-1}}\beta^{\gamma\delta^{-1}} \equiv_p \gamma$ auf der Exponentenebene *modulo* q gerechnet werden
- Da γ jedoch rechts nicht als Exponent, sondern als Basiszahl, vorkommt, muss auch die linke Seite *modulo* q reduziert werden
- Beim DSA hat der Signierschlüssel also die Form $\hat{k} = (p, q, \alpha, a)$, wobei $a \in \mathbb{Z}_q^*$ ist
- Der zugehörige Verifikationsschlüssel ist $k = (p, q, \alpha, \beta)$ mit $\beta = \alpha^a \bmod p$
- Zudem gilt $X = \mathbb{Z}_q$ und $Y = \mathbb{Z}_q \times \mathbb{Z}_q^*$
- Zu gegebenem $x \in X$ wird zufällig eine geheime Zahl $z \in \mathbb{Z}_q^*$ gewählt

$$sig(\hat{k}, z, x) = (\gamma, \delta), \text{ wobei } \begin{cases} \gamma = (\alpha^z \bmod p) \bmod q \\ \delta = (x + a\gamma)z^{-1} \bmod q \in \mathbb{Z}_q^* \end{cases}$$

- Im Fall $\gamma = 0$ oder $\delta = 0$ muss ein neues z gewählt werden

- Die Verifikationsbedingung ist

$$ver(k, x, \gamma, \delta) = \begin{cases} 1, & (\alpha^e \beta^d \bmod p) \bmod q = \gamma, \\ 0, & sonst, \end{cases}$$

wobei $e = x\delta^{-1} \bmod q$ und $d = \gamma\delta^{-1} \bmod q$ ist

- Die Korrektheit ergibt sich wie folgt:
  - Im Fall $sig(\hat{k}, z, x) = (\gamma, \delta)$ ist

$$\alpha^e \beta^d \equiv_p \alpha^{x\delta^{-1}} \alpha^{a\gamma\delta^{-1}} \equiv_p \alpha^{\delta^{-1}(x+a\gamma)} \equiv_p \alpha^{(x+a\gamma)^{-1}z(x+a\gamma)} \equiv_p \alpha^z$$

woraus sich

$$(\alpha^e \beta^d \bmod p) \bmod q = (\alpha^z \bmod p) \bmod q = \gamma$$

ergibt

Der Digital Signature Algorithm (DSA)

Beispiel

- Seien $q = 101$, $p = 78q + 1 = 7879$, $g = 3$ $\quad (ord_p(3) = p - 1)$

  $$\rightsquigarrow \alpha = 3^{78} \bmod p = 170 \text{ hat Ordnung } q$$

- Wir wählen $a = 75 \in \mathbb{Z}_q^*$, d.h. $\beta = \alpha^a \bmod p = 170^{75} \bmod p = 4567$
- Um den Text $x = 22 \in \mathbb{Z}_q$ zu signieren, wählen wir die geheime Zufallszahl $z = 50 \in \mathbb{Z}_q^*$ ($\rightsquigarrow z^{-1} = 99$) und erhalten dann

  $$\begin{aligned} \gamma &= (170^{50} \bmod 7879) \bmod 101 \\ &= 2518 \bmod 101 \\ &= 94 \\ \delta &= (22 + 75 \cdot 94) \cdot 99 \bmod 101 \\ &= 97 \ (\rightsquigarrow \delta^{-1} = 25) \end{aligned}$$

  d.h. $sig(p, q, \alpha, z, x) = (94, 97)$, wobei $\hat{k} = (p, q, \alpha, a)$

Beispiel (Fortsetzung)

- Um diese Signatur zu prüfen berechnen wir:

$$\begin{aligned} e &= x\delta^{-1} \bmod q \\ &= 22 \cdot 25 \bmod 101 \\ &= 45 \\ d &= \gamma\delta^{-1} \bmod q \\ &= 94 \cdot 25 \bmod 101 \\ &= 27 \end{aligned}$$

$\rightsquigarrow (\alpha^e \beta^d \bmod p) \bmod q = (170^{45} 4547^{27} \bmod 7879) \bmod 101 = 94 \quad \triangleleft$

Der ECDSA (Elliptic Curve DSA)

- Der ECDSA wurde im Jahr 2000 als FIPS (Federal Information Processing Standard) 186-2 Standard deklariert
- Sei E eine elliptische Kurve über einem endlichen Körper $\mathbb{F}_{p^n}$
- Sei $A \in E$ ein Punkt der Ordnung q (q prim), so dass das Diskrete-Logarithmus-Problem zur Basis A in E schwierig ist
- Zudem sei $h\colon \{0,1\}^* \to \mathbb{Z}_q$ eine kryptografische Hashfunktion
- Der ECDSA besteht aus folgenden Komponenten:

  Textraum: $X = \{0,1\}^*$
  Signaturraum: $Y = \mathbb{Z}_q^* \times \mathbb{Z}_q^*$
  Signierschlüssel: $\hat{k} = (E, q, A, m)$, $m \in \mathbb{Z}_q^*$
  Verifikationsschlüssel: $k = (E, q, A, B)$, wobei $B = m \cdot A$ ist

Der ECDSA (Elliptic Curve DSA)

- Signaturerstellung: Um einen Text $x \in X$ zu signieren,
  - wählt der Signierer zufällig eine geheime Zahl $z \in \mathbb{Z}_q^*$ und
  - berechnet $sig(\hat{k}, x, z) = (\gamma, \delta)$ mit

$$\begin{aligned}(u, v) &:= zA \\ \gamma &:= u \bmod q \\ \delta &:= (h(x) + m\gamma)z^{-1} \bmod q\end{aligned}$$

- Hierbei wird u als eine Zahl in $\{0, \ldots, p^n - 1\}$ interpretiert
- Falls $\gamma = 0$ oder $\delta = 0$ ist, muss eine neue Zufallszahl z gewählt und der Vorgang wiederholt werden

Der ECDSA (Elliptic Curve DSA)

- Verifikation: $ver(k, x, \gamma, \delta) = 1$, falls $u \bmod q = \gamma$ ist, wobei

$$\begin{aligned} e &:= h(x)\delta^{-1} \bmod q \\ d &:= \gamma\delta^{-1} \bmod q \\ (u, v) &:= eA + dB \end{aligned}$$

- Korrektheit der Verifikation beim ECDSA:

$$\begin{aligned} (u, v) &= eA + dB \\ &= (h(x)\delta^{-1})A + (\gamma\delta^{-1})mA \\ &= (h(x) + m\gamma)\delta^{-1}A \\ &= zA \text{ (da } (h(x) + m\gamma)\delta^{-1} \equiv_q z) \end{aligned}$$

Der ECDSA (Elliptic Curve DSA)

Beispiel

- Sei E über $\mathbb{Z}_{11}$ definiert durch $y^2 = x^3 + x + 6$
- Wir wählen $A = (2,7)$, $m = 7 \to p = 11, q = 13, B = 7A = (7,2)$
- Um einen Text x mit dem Hashwert $h(x) = 4$ unter Verwendung des Signierschlüssels $\hat{k} = (E, q, A, m)$ und der Zufallszahl $z = 3$ signieren,
  - berechnet Alice

$$\begin{aligned}(u,v) &:= zA = 3 \cdot (2,7) = (8,3)\\ \gamma &:= u \bmod q = 8\\ \delta &:= (4 + 7 \cdot 8)3^{-1} \bmod 13 = 7\end{aligned}$$

  - und erhält die Signatur $sig(\hat{k}, z, x) = (8,7)$

Beispiel (Fortsetzung)

- Um diese Signatur mit dem Verifikationsschlüssel $k = (E, q, A, B)$ zu überprüfen,
  - berechnet Bob

$$
\begin{aligned}
e &:= h(x)\delta^{-1} \bmod q = 4 \cdot 7^{-1} \bmod 13 = 4 \cdot 2 \bmod 13 = 8 \\
d &:= \gamma\delta^{-1} \bmod q = 8 \cdot 2 \bmod 13 = 3 \\
(u, v) &:= eA + dB = 8 \cdot (2,7) + 3 \cdot (7,2) = (8,3)
\end{aligned}
$$

  - und testet die Kongruenz $u \equiv_q \gamma$ ◁

Die One-time-Signatur von Lamport

- Leslie Lamport konnte 1979 zeigen, dass sich digitale Signaturen auf der Basis einer Einwegfunktion f konstruieren lassen
- Damit die Signatur allerdings sicher ist, muss für jeden Text ein neues Schlüsselpaar $(\hat{k}, k)$ generiert werden
- Ein Signierschlüssel $\hat{k}$ darf also nur zum Signieren eines einzelnen Textes verwendet werden
- Seien U und V endliche Mengen und sei $f : U \to V$ eine Funktion
- Zudem sei $\ell \geq 1$ die vorgegebene Textlänge, d.h. der Textraum ist $X = \{0, 1\}^\ell$
- Der Signaturraum ist dann $Y = U^\ell$
- Um ein Schlüsselpaar $(\hat{k}, k)$ zu generieren, wird zufällig eine Folge von 2ℓ Elementen $u_{i,b}$ für $i = 1, \ldots, \ell$ und $b = 0, 1$ aus U gewählt und der Signierschlüssel $\hat{k} = \binom{u_{1,0} \ldots u_{\ell,0}}{u_{1,1} \ldots u_{\ell,1}}$ gebildet
- Der zugehörige Verifikationsschlüssel ist dann $k = \binom{v_{1,0} \ldots v_{\ell,0}}{v_{1,1} \ldots v_{\ell,1}}$ mit $v_{i,b} = f(u_{i,b})$ für alle $i = 1, \ldots, \ell$ und $b = 0, 1$

Die One-time-Signatur von Lamport

- Signaturerstellung: Die Signatur für einen Text $x = x_1 \dots x_\ell \in X$ ist

$$sig(\hat{k}, x) = (u_{1,x_1}, \dots, u_{\ell,x_\ell})$$

- Verifikation: Für eine Signatur $y = (u_1, \dots, u_\ell)$ und einen Text $x = x_1 \dots x_\ell$ gilt

$$ver(k, x, y) = \begin{cases} 1, & f(u_i) = v_{i,x_i} \text{ für } i = 1, \dots, \ell, \\ 0, & sonst \end{cases}$$

Die One-time-Signatur von Lamport

Beispiel

- Wir wählen als Einwegfunktion eine Funktion der Form $f : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ mit $f(u) = g^u \bmod p$, wobei g ein Erzeuger von $\mathbb{Z}_p^*$ ist
- Z.B. sei $p = 7879$ und $g = 3$, also $f(u) = 3^u \bmod 7879$
- Weiter sei $\ell = 3$
- Dann erhalten wir für den zufällig gewählten Signierschlüssel $\hat{k} = \begin{pmatrix} 5831 & 4285 & 2467 \\ 803 & 735 & 6449 \end{pmatrix}$ den Verifikationsschlüssel $k = \begin{pmatrix} 2009 & 268 & 4721 \\ 4672 & 3810 & 5731 \end{pmatrix}$
- Die Signatur y für den Text $x = 110$ ist dann

$$y = sig(\hat{k}, x) = (u_{1,x_1}, u_{2,x_2}, u_{3,x_3}) = (u_{1,1}, u_{2,1}, u_{3,0}) = (803, 735, 2467)$$

- Für diese Signatur $y = (u_1, u_2, u_3)$ ist $ver(k, x, y) = 1$, da $f(u_i) = v_{i,x_i}$ für $i = 1, 2, 3$ gilt:

$$\begin{aligned} i = 1 : f(u_1) &= f(803) = 3^{803} \bmod 7879 = 4672 = v_{1,x_1} \\ i = 2 : f(u_2) &= f(735) = 3^{735} \bmod 7879 = 3810 = v_{2,x_2} \\ i = 3 : f(u_3) &= f(2467) = 3^{2467} \bmod 7879 = 4721 = v_{3,x_3} \end{aligned}$$

◁

Die One-time-Signatur von Lamport

- Ähnlich wie bei MACs können wir einen Angriff gegen ein digitales Signaturverfahren wie folgt modellieren
- Hierbei nehmen wir an, dass der Angreifer die Texte, deren Signaturen er kennt, adaptiv wählen kann
- Es handelt sich also um eine existentielle Fälschung bei adaptiv wählbaren Texten

Definition. Sei $0 \leq \varepsilon \leq 1$ und sei $q \in \mathbb{N}$

- Ein (ε, q)-***Fälscher*** für ein digitales Signaturverfahren ist ein probabilistischer Algorithmus $\mathcal{A}$, der
  - bei Eingabe eines Verifikationsschlüssels k, wobei das Schlüsselpaar $(\hat{k}, k)$ zufällig gewählt wird
  - nach den Signaturen $y_i = sig(\hat{k}, x_i)$ von q Texten $x_1, \ldots, x_q$ adaptiv fragt und
  - mit Wahrscheinlichkeit mindestens ε eine Fälschung (x, y) mit $x \notin \{x_1, \ldots, x_q\}$ und $ver(k, x, y) = 1$ ausgibt

Die One-time-Signatur von Lamport

Satz. Sei $f : U \to V$ eine Funktion

Falls für die zugehörige one-time Signatur ein $(\varepsilon, 0)$-Fälscher LAMPORT-FÄLSCHUNG(k) existiert, dann lässt sich für ein zufällig gewähltes $u \in_R U$ mit Wahrscheinlichkeit mindestens $\varepsilon/2$ ein Urbild von $v = f(u)$ bestimmen

Beweis.

Betrachte folgenden probabilistischen Algorithmus LAMPORT-URBILD(v):

Prozedur `Lamport-Urbild`(v)

```
wähle zufällig ein Indexpaar (j, a) und setze v_{j,a} := v
for all (i, b) ∈ [ℓ] × {0,1} \ {(j, a)} do
    wähle zufällig u_{i,b} ∈_R U und setze v_{i,b} := f(u_{i,b})
k := (v_{1,0}...v_{ℓ,0} / v_{1,1}...v_{ℓ,1})
(x_1 ... x_ℓ, (u_1, ..., u_ℓ)) =: LAMPORT-FÄLSCHUNG(k)
if f(u_j) = v then output(u_j) else output(?)
```

$$k := \binom{v_{1,0} \dots v_{\ell,0}}{v_{1,1} \dots v_{\ell,1}}$$

Beweis (Fortsetzung)

- Wie üblich bezeichnen wir die Zufallsvariablen, die die Wahl von v, j, a, k und $(x, y) = (x_1 \dots x_\ell, (u_1, \dots, u_\ell))$ beschreiben, mit entsprechenden Großbuchstaben
- Dann müssen wir zeigen, dass U_J mit Wahrscheinlichkeit mindestens $\varepsilon/2$ ein f-Urbild von V ist, wobei V die Wahl von $v = f(u)$ für ein zufällig gewähltes $u \in_R U$ beschreibt
- Da die Verteilung von K identisch zur Schlüsselgenerierung der Lamport-Signatur und LAMPORT-FÄLSCHUNG ein $(\varepsilon, 0)$-Fälscher ist, folgt

$$\Pr[ver(K, X, Y) = 1] \geq \varepsilon$$

- Da zudem K (und damit auch (X, Y)) unabhängig von (J, A) und auch J und A unabhängig voneinander sind, ist A von (J, K, X, Y) und damit auch von X_J unabhängig

Die One-time-Signatur von Lamport

Beweis (Schluss)

- Sei p die Erfolgswk von LAMPORT-URBILD bei Eingabe V
- Wegen

$$ver(k, x_1 \dots x_\ell, (u_1, \dots, u_\ell)) = 1 \wedge x_j = a \quad \Rightarrow \quad f(u_j) = v_{j,x_j} = v_{j,a} = v$$

folgt nun

$$\begin{aligned} p &\geq \Pr[ver(K, X, Y) = 1 \wedge X_J = A] \\ &= \underbrace{\Pr[ver(K, X, Y) = 1]}_{\geq \varepsilon} \underbrace{\Pr[X_J = A \mid ver(K, X, Y) = 1]}_{=1/2} \\ &\geq \varepsilon/2 \end{aligned}$$

□

Die One-time-Signatur von Lamport

Als nächstes untersuchen wir die Sicherheit der Lamport-Signatur, falls der Angreifer in der Lage ist, sich für einen beliebigen Text x' seiner Wahl eine gültige Signatur y' zu beschaffen

Satz. Sei $f : U \to V$ eine Funktion.

Falls für die zugehörige one-time Signatur ein $(\varepsilon, 1)$-Fälscher Lamport-Fälschung'(k) existiert, so lässt sich für ein zufällig gewähltes $u \in_R U$ mit Wahrscheinlichkeit $\geq \varepsilon/2\ell$ ein f-Urbild von $v = f(u)$ bestimmen

Für den Beweis betrachten wir folgenden probabilistischen Algorithmus `Lamport-Urbild'` und zeigen, dass er für ein zufällig gewähltes $u \in_R U$ bei Eingabe $v = f(u)$ mit Wahrscheinlichkeit $\geq \varepsilon/2\ell$ ein f-Urbild von v ausgibt

Die One-time-Signatur von Lamport

Für den Beweis betrachten wir folgenden probabilistischen Algorithmus `Lamport-Urbild'` und zeigen, dass er für ein zufällig gewähltes $u \in_R U$ bei Eingabe $v = f(u)$ mit Wahrscheinlichkeit $\geq \varepsilon/2\ell$ ein f-Urbild von v ausgibt:

Prozedur `Lamport-Urbild'`(v)

wähle zufällig ein Indexpaar (j, a) und setze $v_{j,a} := v$
for all $(i, b) \neq (j, a)$ **do**
  wähle zufällig $u_{i,b} \in_R U$ und setze $v_{i,b} := f(u_{i,b})$
$k := \binom{v_{1,0} \dots v_{\ell,0}}{v_{1,1} \dots v_{\ell,1}}$
simuliere $\textsc{Lamport-Fälschung'}(k)$ und beantworte die Frage x'
    mit $u_{1,x'_1}, \dots, u_{\ell,x'_\ell}$ (falls $x'_j = a$ ist, brich ab und gib ? aus);
    sei $(x, y) = (x_1 \dots x_\ell, (u_1, \dots, u_\ell))$ die erzeugte Ausgabe
if $f(u_j) = v$ **then output**(u_j) **else output**$(?)$

Die One-time-Signatur von Lamport

Beweis.

- Sei p' die Erfolgswk von LAMPORT-URBILD' bei Eingabe V
- LAMPORT-URBILD' kann die Frage von LAMPORT-FÄLSCHUNG'(k) nach der Signatur von x' nur dann beantworten, wenn $x'_j \neq a$ ist
- Es ist klar, dass in diesem Fall u_j ein Urbild von v ist, wenn zudem $ver(k, x_1 \ldots x_\ell, (u_1, \ldots, u_\ell)) = 1 \ \wedge \ x_j = a$ gilt
- Da jedoch die Simulation von LAMPORT-FÄLSCHUNG'(k) eventuell abgebrochen wird (und die Abbruchbedingung von (j, a) abhängt), können wir nicht mehr davon ausgehen, dass diese Simulation mit Wahrscheinlichkeit ε eine Fälschung (x, y) liefert und (X, Y) unabhängig von (J, A) ist
- Durch eine einfache Modifikation von LAMPORT-URBILD'(v) erhalten wir jedoch eine Prozedur LAMPORT-URBILD* (ohne Eingabe), deren Ausgabeverhalten mit der von LAMPORT-URBILD'(V) identisch ist, und von der wir zeigen können, dass sie mit Wahrscheinlichkeit $p^* \geq \varepsilon/2\ell$ Erfolg hat (also nicht Fragezeichen ausgibt):

Die One-time-Signatur von Lamport

Beweis (Fortsetzung)

- Durch eine einfache Modifikation von Lamport-Urbild'(v) erhalten wir jedoch eine Prozedur Lamport-Urbild* (ohne Eingabe), deren Ausgabeverhalten mit der von Lamport-Urbild'(V) identisch ist, und von der wir zeigen können, dass sie mit Wahrscheinlichkeit $p^* \geq \varepsilon/2\ell$ Erfolg hat (also nicht Fragezeichen ausgibt):

Prozedur `Lamport-Urbild`*

wähle zufällig ein Indexpaar (j, a)
for all (i, b) **do** wähle zufällig $u_{i,b} \in_R U$ und setze $v_{i,b} := f(u_{i,b})$
$k := \binom{v_{1,0} \ldots v_{\ell,0}}{v_{1,1} \ldots v_{\ell,1}}$
simuliere Lamport-Fälschung'(k) und beantworte die Frage x' mit $u_{1,x'_1}, \ldots, u_{\ell,x'_\ell}$;
sei $(x, y) = (x_1 \ldots x_\ell, (u_1, \ldots, u_\ell))$ die erzeugte Ausgabe
if $f(u_j) = v_{j,a} \wedge x'_j \neq a$ **then output**(u_j) **else output**$(?)$

Die One-time-Signatur von Lamport

Beweis (Fortsetzung)

- Im Unterschied zu LAMPORT-URBILD'(v) wählt sich LAMPORT-URBILD* also die Eingabe $v = v_{j,a}$ gemäß der Verteilung von V selbst und kennt daher auch ein Urbild $u_{j,a}$ von $v_{j,a}$
- Somit kann LAMPORT-URBILD* bei der Simulation von LAMPORT-FÄLSCHUNG'(k) die Frage nach der Signatur von x' auch im Fall $x'_j = a$ beantworten
- Die Bedingung für die Ausgabe von u_j ist jedoch bei beiden Prozeduren dieselbe, d.h. die Ausgabe von LAMPORT-URBILD* hat dieselbe Verteilung wie die von LAMPORT-URBILD'(V) und somit gilt $p' = p^*$
- Der einzige Unterschied ist, dass immer wenn LAMPORT-URBILD'(V) in Zeile 5 ein Fragezeichen ausgibt, LAMPORT-URBILD* dies erst in Zeile 6 tut

Die One-time-Signatur von Lamport

Beweis (Schluss)

- Da in der Prozedur Lamport-Urbild* die ZV (J, A) unabhängig von (K, X', X, Y) ist, folgt nun

$$
\begin{aligned}
p* &= \Pr[f(U_J) = V_{J,A} \land X'_J \neq A] \\
&\geq \Pr[ver(K, X, Y) = 1 \land X_J = A \land X'_J \neq A] \\
&= \Pr[ver(K, X, Y) = 1] \underbrace{\Pr[X'_J \neq X_J = A \mid ver(K, X, Y) = 1]}_{\geq 1/2\ell} \\
&\geq \varepsilon/2\ell
\end{aligned}
$$

□

- Die Lamport-Signatur hat aus praktischer Sicht einige Nachteile, die sich jedoch teilweise beheben lassen (siehe Übungen)
- So lässt sich sowohl die Länge des privaten Signierschlüssels (mittels Pseudozufallsgeneratoren) als auch des öffentlichen Verifikationsschlüssels (mittels Hash-Listen) verringern
- Zudem können bei Verwendung von Hash-Bäumen mit demselben Schlüsselpaar auch mehrere Nachrichten signiert und verifiziert werden

Full Domain Hash (FDH) Signaturen

- Sei $\mathcal{F} = \{f_k | k \in K\}$ eine Familie von Falltür-Permutationen auf einer Menge U, d.h. es lassen sich (zufällig) Schlüsselpaare $(\hat{k}, k) \in K \times K$ generieren, so dass gilt:
  - $f_{\hat{k}}(f_k(u)) = u$ für alle $u \in U$
  - f_k ist eine Einweg-Permutation auf U, d.h. für ein zufällig gewähltes Schlüsselpaar $(\hat{k}, k) \in K \times K$ und ein zufällig gewähltes $v \in U$ ist es schwer, ohne Kenntnis von $\hat{k}$ ein Urbild u mit $f_k(u) = v$ zu finden (genauer: jedem effizienten Angreifer gelingt dies nur mit vernachlässigbarer Wahrscheinlichkeit)
- Weiter sei $h : \{0, 1\}^* \to U$ eine Funktion
- Die auf $\mathcal{F}$ und h basierende FDH-Signatur funktioniert wie folgt:

Full Domain Hash (FDH) Signaturen

- Die auf $\mathcal{F}$ und h basierende FDH-Signatur funktioniert wie folgt:
  - Zuerst wird ein Schlüsselpaar $(\hat{k}, k) \in K \times K$ generiert, wobei $\hat{k}$ als Signierschlüssel und k als Verifikationsschlüssel fungiert
  - Signaturerstellung: Die Signatur für einen Text $x \in X$ ist

$$sig(\hat{k}, x) = f_{\hat{k}}(h(x))$$

  - Verifikation: Für eine Signatur $y \in U$ und einen Text $x \in \{0,1\}^*$ gilt

$$ver(k, x, y) := \begin{cases} 1, & f_k(y) = h(x), \\ 0, & sonst \end{cases}$$

- Z.B. beruht das RSA-Signaturverfahren in Verbindung mit einer Hashfunktion auf diesem Prinzip
- Ein Problem hierbei ist allerdings, dass die benutzten RSA-Falltür-Permutationen einen Definitionsbereich der Größe 2^{1024} haben, um eine ausreichend große Sicherheit zu erreichen, wogegen die benutzten Hashfunktionen nur eine Länge von 160 Bit haben

Sicherheitsanalyse der FDH-Signatur im ZOM

- In der Praxis behilft man sich damit, dass man die 160-Bit-Hashwerte durch eine deterministische Paddingfunktion auf 1024-Bit aufbläht, was die Sicherheit allerdings beeinträchtigen kann
- Bei Verwendung einer Zufallsfunktion $G : \{0, 1\}^* \to U$ (vgl. Zufalls-Orakel-Modell, ZOM) anstelle von h lässt sich die Fälschungssicherheit der resultierenden FDH-Signatur aus der Falltüreigenschaft von $\mathcal{F}$ herleiten
- Das ZOM modelliert eine Hashfunktion mit optimalen kryptografischen Eigenschaften, d.h. die Zufallsvariablen $U_x = G(x)$ sind stochastisch unabhängig und gleichverteilt auf U
- Zudem füllt der Wertebereich von G den gesamten Definitionsbereich der Funktionen f_k aus (full domain hash)
- Wir betrachten zuerst den Fall einer existentiellen Fälschung bei bekanntem Verifikationsschlüssel, d.h. der Angreifer muss eine Fälschung (x, y) mit $ver(k, x, y) = 1$ produzieren, ohne auch nur eine Signatur y' für einen Text x' zu kennen

Sicherheitsanalyse der FDH-Signatur im ZOM

- Sei `FDH-Fälschung` ein probabilistischer Algorithmus, der für einen zufällig generierten Verifikationsschlüssel k mit Wahrscheinlichkeit ε eine existentielle Fälschung (x, y) mit $f_k(y) = G(x)$ ausgibt
- Dabei nehmen wir an, dass `FDH-Fälschung` eine Folge von q verschiedenen Fragen $x_1, \ldots, x_q$ an G stellt
- Es ist klar, dass ein solcher Angriff im Fall $x \notin \{x_1, \ldots, x_q\}$ mit der Wahrscheinlichkeit $\varepsilon = 1/\|U\|$ gelingt
- Da diese Erfolgswk durch Ausgabe eines beliebigen Paares (x, y) bereits mit $q = 0$ Fragen an G erreicht wird, können wir zudem annehmen, dass $x \in \{x_1, \ldots, x_q\}$ enthalten ist (sofern $q \geq 1$ ist)
- Betrachte folgenden Invertierungsalgorithmus für f_k:

Sicherheitsanalyse der FDH-Signatur im ZOM

Prozedur `FDH-Invert`(k, v)

wähle zufällig $j \in_R \{1, \ldots, q\}$

simuliere `FDH-Fälschung`(k) und beantworte dabei die Frage x_i im Fall $i = j$ durch $v_j = v$ und sonst durch ein zufällig gewähltes $v_i \in_R U$; sei (x, y) die erzeugte Ausgabe

if $f_k(y) = v$ **then output**(y) **else output**$(?)$

Satz

Falls `FDH-Fälschung`(k) für einen zufällig gewählten Verifikationsschlüssel k mit Wahrscheinlichkeit ε eine Fälschung (x, y) mit $f_k(y) = G(x)$ ausgibt und dabei $q \geq 1$ Fragen an G stellt, so gibt `FDH-Invert`(k, v) für einen zufälligen Verifikationsschlüssel k und ein zufälliges $v \in_R U$ mit Wahrscheinlichkeit $\geq \varepsilon/q$ ein f_k-Urbild von v aus

Da sich mit Wahrscheinlichkeit $1/\|U\|$ ein Urbild erraten lässt, ist der Satz nur im Fall $\varepsilon > q/\|U\|$ interessant

Beweis.

- Seien J, K, U, V, X, $X_1, \dots, X_q$ Zufallsvariablen, die die Wahl von j, k, u, v, x, $x_1, \dots, x_q$ beschreiben
- Da die Eingabe v gleichverteilt ist, erhält `FDH-Fälschung` auf die Fragen $x_1, \dots, x_q$ an G stochastisch unabhängig unter Gleichverteilung gewählte Strings $v_1, \dots, v_q$ als Antwort, was dem ZOM entspricht
- Daher liefert die Simulation von `FDH-Fälschung`(k) für einen zufällig generierten Schlüssel k mit Wahrscheinlichkeit ε eine Fälschung (x, y) mit $f_k(y) = G(x)$:

  $$\Pr[f_K(Y) = G(X)] = \varepsilon$$

- Wir wollen zeigen, dass $\Pr[f_K(Y) = V] \geq \varepsilon/q$ ist
- Da $x \in \{x_1, \dots, x_q\}$ enthalten ist, existiert ein i mit $x = x_i$ und die Gleichheit $f_k(y) = G(x)$ impliziert $f_k(y) = G(x_i) = v_i$

Beweis (Fortsetzung)

- Folglich gilt die Implikation

$$f_k(y) = G(x) \wedge j = i \implies f_k(y) = v_j = v$$

  und es folgt

$$\Pr[f_K(Y) = V] \geq \Pr[f_K(Y) = G(X) \wedge J = I]$$

- Zudem wird $j \in \{1, \ldots, q\}$ zufällig gewählt und die Fragen $x_1, \ldots, x_q$ werden unabhängig voneinander durch zufällige $v_1, \ldots, v_q \in_R U$ beantwortet (nach Voraussetzung trifft dies auch auf $v_j = v$ zu)
- Daher erhält `FDH-Fälschung` weder durch k noch durch die Antworten $v_1, \ldots, v_q$ irgendeine Information über j

Beweis (Schluss)

- Folglich sind neben der Eingabe K auch die Ausgabe (X, Y) und somit auch die Zufallsvariable I, die den Index $i \in \{1, \ldots, q\}$ mit $x = x_i$ bestimmt, stochastisch unabhängig von J
- Daher folgt

$$\begin{aligned}
\Pr[f_K(Y) = V] &\geq \Pr[f_K(Y) = G(X) \wedge J = I] \\
&= \Pr[f_K(Y) = G(X)] \underbrace{\Pr[J = I \mid f_K(Y) = G(X)]}_{1/q} \\
&= \Pr[f_K(Y) = G(X)]/q \; = \; \varepsilon/q
\end{aligned}$$

$\square$

Sicherheitsanalyse der FDH-Signatur im ZOM

- Falls sich also f_k nur mit einer vernachlässigbaren Wahrscheinlichkeit ε' effizient invertieren lässt, so gelingt einem ähnlich effizienten Angreifer, der nicht mehr als q Hashwertberechnungen durchführt, im ZOM höchstens mit einer (ebenfalls vernachlässigbaren) Wahrscheinlichkeit $\varepsilon \leq q\varepsilon'$ eine existentielle Fälschung für die FDH-Signatur
- Als nächstes beweisen wir die Fälschungssicherheit der FDH-Signatur im ZOM gegenüber einem existentiellen Angriff mit adaptiv gewählten Texten

Sicherheitsanalyse der FDH-Signatur im ZOM

- Sei `FDH-Fälschung'` ein probabilistischer Algorithmus, der für einen zufällig generierten Verifikationsschlüssel k mit Wahrscheinlichkeit ε eine existentielle Fälschung (x, y) mit $f_k(y) = G(x)$ ausgibt und insgesamt für q Texte $x_1, \ldots, x_q$ den Wert $G(x_i)$ oder die Signatur $sig(\hat{k}, x_i) = f_{\hat{k}}(G(x_i))$ erfragt
- Dabei können wir o.B.d.A. annehmen, dass `FDH-Fälschung'` zwar nicht die Signatur von x, aber den G-Wert von x erfragt und vor jeder Frage nach der Signatur eines Textes x_i den G-Wert von x_i erfragt

Sicherheitsanalyse der FDH-Signatur im ZOM

Satz

Falls `FDH-Fälschung'`(k) für einen zufällig gewählten Verifikationsschlüssel k mit Wahrscheinlichkeit ε eine Fälschung (x, y) mit $f_k(y) = G(x)$ berechnet und dabei für q Texte x_i den Wert $G(x_i)$ sowie im Fall $x_i \neq x$ evtl. auch die Signatur $sig(\hat{k}, x_i)$ erfragt, so lässt sich für einen zufälligen Verifikationsschlüssel k und ein zufälliges $v \in_R U$ mit Wahrscheinlichkeit $\geq \varepsilon/q$ ein f_k-Urbild von v bestimmen

Für den Beweis betrachten wir folgenden probabilistischen Algorithmus

Prozedur `FDH-Invert'`(k, v)

wähle zufällig $j \in_R \{1, \ldots, q\}$
simuliere `FDH-Fälschung'`(k) und beantworte dabei jede Frage x_i an G im Fall $i = j$ durch $v_j = v$ und sonst durch $v_i = f_k(u_i)$, wobei u_i zufällig aus U gewählt wird; falls später die Signatur von x_i erfragt wird, gib u_i als Antwort (falls $i = j$ ist, brich ab und gib ? aus); sei (x, y) die erzeugte Ausgabe
if $f_k(y) = v$ **then output**(y) **else output**$(?)$

Sicherheitsanalyse der FDH-Signatur im ZOM

Prozedur `FDH-Invert'`(k, v)

---

wähle zufällig $j \in_R \{1, \ldots, q\}$
simuliere `FDH-Fälschung'`(k) und beantworte dabei jede Frage x_i an G im Fall $i = j$ durch $v_j = v$ und sonst durch $v_i = f_k(u_i)$, wobei u_i zufällig aus U gewählt wird; falls später die Signatur von x_i erfragt wird, gib u_i als Antwort (falls $i = j$ ist, brich ab und gib ? aus); sei (x, y) die erzeugte Ausgabe
if $f_k(y) = v$ **then output**(y) **else output**$(?)$

- Da die Frage nach der Signatur von x_i im Fall $i = j$ unbeantwortet bleibt, ist nicht klar, dass die Simulation von `FDH-Fälschung'`(k) mit Wahrscheinlichkeit ε eine Fälschung (x, y) mit $f_k(y) = G(x)$ findet
- Wir können aber eine Prozedur `FDH-Invert`$^*(k)$ angeben, die nur k als Eingabe erhält, so dass die Ausgaben von `FDH-Invert`$^*(K)$ und von `FDH-Invert'`(K, V) identisch verteilt sind, und

$$\Pr[\texttt{FDH-Invert'}(K, V) \neq ?] = \Pr[\texttt{FDH-Invert}^*(K) \neq ?] \geq \varepsilon / q$$

gilt (siehe Übungen)

Verbindliche Signaturen (undeniable signatures)

- In manchen Fällen ist es für den Unterzeichner eines Textes nicht wünschenswert, dass jeder dazu in der Lage ist, die Gültigkeit einer vorgelegten Signatur zu verifizieren
- Zum Beispiel könnte eine Softwarefirma (Alice) ihre Produkte mit einer Signatur versehen, die u.a. Virenfreiheit garantiert
- ***Problem:*** Neben den legalen Erwerbern der Software (Bob) können sich auch Kaufinteressenten auf dem Schwarzmarkt von der Gültigkeit einer Signatur (und damit von der Virenfreiheit des signierten Produkts) überzeugen
- ***Lösung:*** Die Gültigkeit einer Signatur lässt sich nur unter Mitwirkung von Alice verifizieren
- ***Neues Problem:*** Alice könnte versuchen, eine von ihr erzeugte gültige Signatur abzuleugnen, indem sie die Verifikation sabotiert

Verbindliche Signaturen (undeniable signatures)

- ***Neues Problem:*** Alice könnte versuchen, eine von ihr erzeugte gültige Signatur abzuleugnen, indem sie die Verifikation sabotiert
- ***Lösung:*** Es gibt zusätzlich ein Ableugnungsprotokoll (disavowal protocol), mit dem Alice die Ungültigkeit von (falschen) Signaturen nachweisen kann
- Falls Alice die Gültigkeit einer Signatur bestreitet und sich dennoch weigert, deren Gültigkeit mithilfe des Ableugnungsprotokolls zu widerlegen, kann man davon ausgehen, dass die Signatur gültig ist

Das Signaturverfahren von Chaum und van Antwerpen

- Bei dem Signaturverfahren von Chaum und van Antwerpen wird eine Primzahl $p = 2q + 1$ und ein Element $\alpha \in \mathbb{Z}_p^*$ der Ordnung q benutzt, wobei q ebenfalls prim und das Diskrete Logarithmus Problem zur Basis α hart ist
- Sei $G = \{\alpha^a | a \in \mathbb{Z}_q\}$ die von α erzeugte Untergruppe von $\mathbb{Z}_p^*$
- Der Text- und Signaturenraum ist $X = Y = G$
- Der Signierschlüssel hat die Form $\hat{k} = (p, \alpha, a)$, $a \in \mathbb{Z}_q^*$ und der zugehörige Verifikationsschlüssel ist $k = (p, \alpha, \beta)$ mit $\beta = \alpha^a \bmod p$
- Signaturerstellung: Die Signatur für einen Text $x \in G$ ist

$$sig(\hat{k}, x) = x^a \bmod p$$

- Um eine Signatur $y \in G$ von Alice für einen Text $x \in G$ zu verifizieren, führt Bob zusammen mit Alice folgendes Verifikationsprotokoll aus:

Das Signaturverfahren von Chaum und van Antwerpen

- Möchte nun Alice Bob gegenüber nachweisen, dass eine Signatur y gültig ist, so führen beide folgendes Verifikationsprotokoll aus:

1. Bob wählt zufällig $e, f \in \mathbb{Z}_q$ und sendet $c = y^e \beta^f \bmod p$ an Alice
2. Alice sendet $d = c^{a^{-1} \bmod q} \bmod p$ zurück an Bob
3. Bob akzeptiert y als gültig, falls $x^e \alpha^f \equiv_p d$ gilt

- Es ist leicht zu sehen, dass Bob eine gültige Signatur $y = x^a \bmod p$ mit Wahrscheinlichkeit 1 als gültig akzeptiert, falls sich beide an das Verifikationsprotokoll halten:

$$x^e \alpha^f \equiv_p \underbrace{(x^{ae} \alpha^{af})}_{y^e \beta^f \equiv_p c}{}^{a^{-1} \bmod q} \equiv_p c^{a^{-1} \bmod q} \equiv_p d$$

Das Signaturverfahren von Chaum und van Antwerpen

Beispiel

- Sei $q = 233$ und $p = 2q + 1 = 2 \cdot 233 + 1 = 467$
- Da $g = 2$ ein Erzeuger von $\mathbb{Z}_p^*$ ist, hat $\alpha = g^2 = 4$ die gewünschte Ordnung $q = (p-1)/2$
- Da α die Untergruppe $QR_p = \{y^2 \bmod p \mid y \in \mathbb{Z}_p^*\}$ der quadratischen Reste in $\mathbb{Z}_p^*$ erzeugt, ist $G = QR_p$
- Die Wahl von $a = 101$ führt auf den Signierschlüssel $\hat{k} = (p, \alpha, a) = (467, 4, 101)$ und den Verifikationsschlüssel $k = (p, \alpha, \beta) = (467, 4, 449)$
- Die Signatur für $x = 119 \in G$ berechnet sich wie folgt:
  - $sig(\hat{k}, x) = x^a \bmod p = 119^{101} \bmod 467 = 129 = y$
- Verifikation der Signatur $y = 129$ für den Text $x = 119$ unter k:
  - Bob wählt $e, f \in \mathbb{Z}_q$ ($e = 38, f = 164$) und sendet $c = y^e \beta^f \bmod p = 129^{38} 449^{164} \bmod 467 = 13$ an Alice
  - Alice sendet $d = c^{a^{-1} \bmod q} \bmod p = 9$ an Bob zurück
  - Bob akzeptiert, da $x^e \alpha^f \bmod p = 119^{38} 4^{164} \bmod 467 = 9 = d$ ist $\triangleleft$

Bemerkung

Die Wahl von p der Form $p = 2q + 1$ mit q prim dient folgenden Zielen:

- Die Ordnung q der Untergruppe G von $\mathbb{Z}_p^*$ ist prim (dies erlaubt die Berechnung von $a^{-1} \bmod q$ in Schritt 2 des Verifikationsprotokolls)
- G ist eine möglichst große Untergruppe von $\mathbb{Z}_p^*$ mit primer Ordnung (man beachte, dass die Ordnung von $\mathbb{Z}_p^*$ gleich $p - 1$, also zusammengesetzt ist)

Behauptung 1

Bob akzeptiert eine ungültige Signatur $y \not\equiv_p x^a$ nur mit Wahrscheinlichkeit $1/q$ (auch wenn sich Alice nicht an das Verifikationsprotokoll hält)

Beweis.

- Alice steht in Zeile 2 des Verifikationsprotokolls vor der Aufgabe, eine Zahl $d \in G$ zu finden, so dass Bob in Zeile 3 akzeptiert
- Das wäre für Alice problemlos möglich, wenn sie e und f kennen würde
- Alice hat aber nur partielles Wissen über das Paar (e, f), nämlich dass es folgende Kongruenz erfüllt:

$$c \quad \equiv_p \quad y^e \beta^f \tag{1}$$

- Da es für jedes $e \in \mathbb{Z}_q$ genau ein $f \in \mathbb{Z}_q$ gibt, so dass das Paar (e, f) die Kongruenz (1) erfüllt, gibt es genau q solche Paare in $\mathbb{Z}_q \times \mathbb{Z}_q$
- Da Alice nur c kennt, sind aus ihrer Sicht diese q Paare alle gleichwahrscheinlich
- Wir zeigen nun, dass unabhängig davon, welches $d \in G$ Alice an Bob sendet, genau eines dieser q Paare zusätzlich die Kongruenz

$$d \quad \equiv_p \quad x^e \alpha^f \tag{2}$$

erfüllt

Beweis (Fortsetzung)

- Folglich akzeptiert Bob mit Wahrscheinlichkeit $1/q$
- Seien $c', d', x', y' \in \mathbb{Z}_q$ die zu $c, d, x, y \in G$ gehörigen Exponenten, d.h. $c \equiv_p \alpha^{c'}, \ldots, y \equiv_p \alpha^{y'}$
- Dann erfüllt ein Paar (e, f) genau dann die beiden Kongruenzen (1) und (2), wenn Folgendes gilt:

$$\begin{matrix} c \equiv_p y^e \beta^f \\ d \equiv_p x^e \alpha^f \end{matrix} \Leftrightarrow \begin{matrix} \alpha^{c'} \equiv_p \alpha^{y'e} \alpha^{af} \\ \alpha^{d'} \equiv_p \alpha^{x'e} \alpha^f \end{matrix} \Leftrightarrow \begin{matrix} c' \equiv_q y'e + af \\ d' \equiv_q x'e + f \end{matrix} \Leftrightarrow \underbrace{\begin{pmatrix} y' & a \\ x' & 1 \end{pmatrix}}_{A} \begin{pmatrix} e \\ f \end{pmatrix} \equiv_q \begin{pmatrix} c' \\ d' \end{pmatrix}$$

- Wegen $\alpha^{y'} \equiv_p y \not\equiv_p x^a \equiv_p \alpha^{x'a}$ folgt $y' \not\equiv_q x'a$ und daher ist $\det A \equiv_q y' - x'a \not\equiv_q 0$ $\square$

Das Signaturverfahren von Chaum und van Antwerpen

- Möchte nun Alice Bob gegenüber nachweisen, dass eine Signatur y ungültig ist, so führen beide folgendes Ableugnungsprotokoll aus:

---

1 Bob wählt zufällig $e_1, f_1 \in \mathbb{Z}_q$ und sendet $c_1 = y^{e_1}\beta^{f_1} \bmod p$ an Alice
2 Alice sendet $d_1 = c_1^{a^{-1} \bmod q} \bmod p$ zurück
3 Bob testet, ob $d_1 \not\equiv_p x^{e_1}\alpha^{f_1}$ ist
4 Bob wählt zufällig $e_2, f_2 \in \mathbb{Z}_q$ und sendet $c_2 = y^{e_2}\beta^{f_2} \bmod p$ an Alice
5 Alice sendet $d_2 = c_2^{a^{-1} \bmod q} \bmod p$ zurück
6 Bob testet, ob $d_2 \not\equiv_p x^{e_2}\alpha^{f_2}$ ist
7 Bob erkennt y als ungültig an, falls der Test in Zeile 3 oder der Test in Zeile 6 erfolgreich war und $(d_1\alpha^{-f_1})^{e_2} \equiv_p (d_2\alpha^{-f_2})^{e_1}$ gilt

---

- Bei den Schritten 1-3 und 4-6 handelt es sich jeweils um eine fehlgeschlagene Verifikation der Signatur y (sofern der Test von Bob in Zeile 3 bzw. 6 positiv ausfällt)
- In Schritt 7 führt Bob zusätzlich einen Konsistenztest aus, um sich davon zu überzeugen, dass Alice die Zahlen d_1 und d_2 gemäß dem Protokoll gewählt hat

Beispiel

- Sei $p = 467, q = 233, \alpha = 4, a = 101, \beta = 449$
- Wir nehmen an, dass der Text $x = 286$ mit der Alice zugeschriebenen Signatur $y = 81$ unterschrieben ist und Alice Bob davon überzeugen möchte, dass y ungültig ist
- Ausführung des Ableugnungsprotokolls:
  - Bob wählt $e_1 = 45, f_1 = 4$ und sendet $c_1 = 305$ an Alice
  - Alice antwortet mit $d_1 = c_1^{a^{-1}} = 109$
  - Bob verifiziert, dass $286^{45}4^4 \equiv_p 149 \not\equiv_p 109$ gilt
  - Bob wählt $e_2 = 125, f_2 = 9$ und sendet $c_2 = 72$ an Alice
  - Alice antwortet mit $d_2 = c_2^{a^{-1}} = 68$
  - Bob verifiziert, dass $286^{125}4^9 \equiv_p 25 \not\equiv_p 68$ gilt
  - Bob erkennt y als ungültig an, da
  $$(109 \cdot 4^{-4})^{125} \equiv_p 188 \equiv_p (68 \cdot 4^{-9})^{45}$$
  ist und somit die Konsistenzbedingung erfüllt ist ◁

Es bleibt zu zeigen, dass sich Bob von der Ungültigkeit einer Signatur y im Fall $y \not\equiv_p x^a$ mit sehr hoher und im Fall $y \equiv_p x^a$ nur mit sehr kleiner Wahrscheinlichkeit überzeugen lässt (auch wenn sich im zweiten Fall Alice nicht an das Ableugnungsprotokoll hält)

Behauptung 2

Im Fall $y \not\equiv_p x^a$ erkennt Bob y mit Wahrscheinlichkeit $1 - \frac{1}{q^2}$ als ungültig an, falls sich beide an das Ableugnungsprotokoll halten

Das Signaturverfahren von Chaum und van Antwerpen

Behauptung 2

Im Fall $y \not\equiv_p x^a$ erkennt Bob y mit Wahrscheinlichkeit $1 - \frac{1}{q^2}$ als ungültig an, falls sich beide an das Ableugnungsprotokoll halten

Beweis.

- Nach Behauptung 1 beträgt die Wahrscheinlichkeit, dass die beiden Tests in Zeile 3 und 6 fehlschlagen genau $\frac{1}{q^2}$
- Wegen $\beta \equiv_p \alpha^a$, $c_i \equiv_p y^{e_i}\beta^{f_i}$ und $d_i \equiv_p c_i^{a^{-1} \bmod q}$ für $i \in \{1, 2\}$ folgt

$$d_i\alpha^{-f_i} \equiv_p (\underbrace{y^{e_i}\beta^{f_i}}_{c_i})^{a^{-1}}\alpha^{-f_i} \equiv_p y^{e_i a^{-1}} \underbrace{\beta^{f_i a^{-1}}}_{\alpha^{f_i}}\alpha^{-f_i} \equiv_p y^{e_i a^{-1}}$$

und somit

$$(d_1\alpha^{-f_1})^{e_2} \equiv_p y^{e_1 a^{-1} e_2} \equiv_p y^{e_2 a^{-1} e_1} \equiv_p (d_2\alpha^{-f_2})^{e_1},$$

d.h. die Konsistenzbedingung wird mit Wahrscheinlichkeit 1 erfüllt □

Behauptung 3

Im Fall $y \equiv_p x^a$ erkennt Bob y nur mit einer Wahrscheinlichkeit $\leq \frac{1}{q}$ als ungültig an, auch wenn sich Alice nicht an das Ableugnungsprotokoll hält

Beweis.

- Bob erkennt y nur dann als ungültig an, wenn Folgendes gilt:
$$(d_1 \not\equiv_p x^{e_1}\alpha^{f_1} \text{ oder } d_2 \not\equiv_p x^{e_2}\alpha^{f_2}) \text{ und } (d_1\alpha^{-f_1})^{e_2} \equiv_p (d_2\alpha^{-f_2})^{e_1}$$
- Da die beiden Fälle $d_1 \not\equiv_p x^{e_1}\alpha^{f_1}$ und $d_2 \not\equiv_p x^{e_2}\alpha^{f_2}$ symmetrisch sind, reicht es, einen davon zu betrachten
- Wir nehmen also an, dass Alice eine Zahl $d_1 \not\equiv_p x^{e_1}\alpha^{f_1}$ an Bob sendet
- Nachdem Alice die Zahl c_2 in Zeile 4 von Bob erhalten hat, weiß sie nur, dass das von Bob gewählte Paar (e_2, f_2) die Kongruenz $c_2 \equiv_p y^{e_2}\beta^{f_2}$ erfüllt
- Wie wir bereits im Beweis zu Behauptung 1 gesehen haben, trifft dies auf genau q Paare zu

Beweis.

- Wir zeigen nun, dass für jedes $d_2 \in G$ genau eines der q Paare (e_2, f_2) die Konsistenzbedingung $(d_1\alpha^{-f_1})^{e_2} \equiv_p (d_2\alpha^{-f_2})^{e_1}$ erfüllt
- Dies beweist, dass unabhängig davon, welches d_2 Alice an Bob sendet, Bob y nur mit Wahrscheinlichkeit $1/q$ als ungültig akzeptiert
- Sei $u = d_1\alpha^{-f_1}$ mod p und seien $c_2', d_2', x', u' \in \mathbb{Z}_q$ die zu c_2, d_2, x, u gehörigen Exponenten
- Dann gilt

$$\begin{array}{c} c_2 \equiv_p y^{e_2}\beta^{f_2} \\ (\underbrace{d_1\alpha^{-f_1}}_{u})^{e_2} \equiv_p (d_2\alpha^{-f_2})^{e_1} \end{array} \Leftrightarrow \begin{array}{r} c_2' \equiv_q x'ae_2 + af_2 \\ u'e_2 \equiv_q d_2'e_1 - e_1f_2 \end{array} \Leftrightarrow \underbrace{\begin{pmatrix} x'a & a \\ u' & e_1 \end{pmatrix}}_{A} \begin{pmatrix} e_2 \\ f_2 \end{pmatrix} \equiv_q \begin{pmatrix} c_2' \\ d_2'e_1 \end{pmatrix}$$

- Wegen $x^{e_1} \equiv_p \underbrace{x^{e_1}\alpha^{f_1}}_{\not\equiv_p d_1}\alpha^{-f_1} \not\equiv_p d_1\alpha^{-f_1} \equiv_p u$ folgt $x'e_1 \not\equiv_q u'$ und somit ist $\det A = x'ae_1 - u'a = a(x'e_1 - u') \not\equiv_q 0$ □

Fail-Stop-Signaturen

- Ein Nachteil aller bisher betrachteten Signaturverfahren ist, dass Alice eine vorgelegte Fälschung (x, y) nicht als solche nachweisen kann
- Dies liegt daran, dass Alice einen Dritten nicht davon überzeugen kann, dass sie die Signatur y nicht selbst erzeugt hat
- Bei so genannten Fail-Stop-Signaturen ist genau dies möglich
- Sollte es einem Angreifer gelingen, das Signaturverfahren zu brechen (“fail”) und eine Fälschung (x, y) zu generieren, so kann Alice dies mit hoher Wahrscheinlichkeit beweisen und somit ihre Signatur widerrufen (“stop”)

Das van Heyst-Pedersen Signaturverfahren

Definition

- Sei $p = 2q + 1$ prim, p, q prim und sei $\alpha \in \mathbb{Z}_p^*$ ein Element der Ordnung q
- Weiter sei $G = \{\alpha^a | a \in \mathbb{Z}_q\}$ die von α in $\mathbb{Z}_p^*$ erzeugte Untergruppe und $\beta = \alpha^a \bmod p$ für ein $a \in \mathbb{Z}_q^*$
- Die Zahlen p, q, α, β werden von einer vertrauenswürdigen Instanz generiert und bekannt gegeben, a wird jedoch vor allen Teilnehmern geheim gehalten
- Der Textraum ist $X = \mathbb{Z}_q$ und der Signaturraum ist $Y = \mathbb{Z}_q \times \mathbb{Z}_q$
- Um einen Signierschlüssel zu generieren, wählt Alice zufällig ein 4-Tupel $\hat{k} = (a_1, b_1, a_2, b_2) \in_R \mathbb{Z}_q^4$
- Der zugehörige Verifikationsschlüssel ist dann $k = (\gamma_1, \gamma_2) = (\alpha^{a_1}\beta^{b_1}, \alpha^{a_2}\beta^{b_2}) \in G^2$

Das van Heyst-Pedersen Signaturverfahren

Definition (Fortsetzung)

- Signaturerstellung: Die Signatur für einen Text $x \in \mathbb{Z}_q$ unter einem Signierschlüssel $\hat{k} = (a_1, b_1, a_2, b_2) \in \mathbb{Z}_q^4$ ist

$$sig(\hat{k}, x) = (y_1, y_2) = (a_1 + xa_2 \bmod q, b_1 + xb_2 \bmod q)$$

- Verifikation: Für einen Verifikationsschlüssel $k = (\gamma_1, \gamma_2)$, einen Text $x \in \mathbb{Z}_q$ und eine Signatur $y = (y_1, y_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$ gilt

$$ver(k, x, y) = \begin{cases} 1, & \gamma_1 \gamma_2^x \equiv_p \alpha^{y_1} \beta^{y_2}, \\ 0, & sonst \end{cases}$$

Das van Heyst-Pedersen Signaturverfahren

Beispiel

- Die vertrauenswürdige Instanz (TTP, trusted third party) generiert
  - Primzahlen p und q mit $p = 2q + 1 = 2 \cdot 1733 + 1 = 3467$, sowie
  - ein Element $\alpha = 4 \in \mathbb{Z}_p^*$ mit $ord_p(\alpha) = q$ und
  - eine geheime Zahl $a = 1567 \in \mathbb{Z}_q^*$ und
  - gibt die Zahlen p, q, α und $\beta = \alpha^a \bmod p = 4^{1567} \bmod p = 514$ bekannt, hält aber a geheim
- Wählt Alice $\hat{k} = (a_1, b_1, a_2, b_2) = (888, 1024, 786, 999)$ als Signierschlüssel, so berechnet sich der zugehörige Verifikationsschlüssel zu $k = (\gamma_1, \gamma_2)$ mit

  $$\gamma_1 = \alpha^{a_1}\beta^{b_1} = 4^{888}514^{1024} = 3405$$

  und

  $$\gamma_2 = \alpha^{a_2}\beta^{b_2} = 4^{786}514^{999} = 2281$$

Beispiel (Fortsetzung)

- Um den Text $x = 1650$ zu signieren, berechnet Alice mit dem Signierschlüssel $\hat{k} = (a_1, b_1, a_2, b_2) = (888, 1024, 786, 999)$ die Signatur $y = sig(\hat{k}, x) = (y_1, y_2)$ mit

$$\begin{aligned} y_1 &= a_1 + xa_2 \bmod q = 888 + 1650 \cdot 786 \bmod q = 1504 \text{ und} \\ y_2 &= b_1 + xb_2 \bmod q = 1024 + 1650 \cdot 999 \bmod q = 1291 \end{aligned}$$

- Um die Signatur $y = (1504, 1291)$ zu überprüfen, testet Bob mit dem Verifikationsschlüssel $k = (\gamma_1, \gamma_2) = (3405, 2281)$ die Verifikationsbedingung

$$\gamma_1 \gamma_2^x = 3405 \cdot 2281^{1650} \equiv_p 2282 \equiv_p 4^{1504} 514^{1291} = \alpha^{y_1} \beta^{y_2}$$

◁

Das van Heyst-Pedersen Signaturverfahren

- Betrachte die Menge

  $$S = \{(\hat{k}, k) \mid \hat{k} = (a_1, b_1, a_2, b_2) \in \mathbb{Z}_q^4, k = (\alpha^{a_1}\beta^{b_1}, \alpha^{a_2}\beta^{b_2}) \in G \times G\}$$

  aller möglichen Schlüsselpaare
- Für einen Verifikationsschlüssel $k \in G \times G$ sei

  $$S(k) = \{\hat{k} \in \mathbb{Z}_q^4 \mid (\hat{k}, k) \in S\}$$

  die Menge aller Signierschlüssel, die zu k passen, und
- für einen Text x und eine Signatur $y = (y_1, y_2)$ sei

  $$S(k, x, y) = \{\hat{k} \in S(k) \mid sig(\hat{k}, x) = y\}$$

  die Menge aller Signierschlüssel in $S(k)$, die für x die Signatur y berechnen

Das van Heyst-Pedersen Signaturverfahren

Lemma

Für jeden Signierschlüssel $\hat{k} \in S(k)$ und jedes Paar (x, y) mit $sig(\hat{k}, x) = y$ ist die Verifikationsbedingung $ver(k, x, y) = 1$ erfüllt

Beweis.

- Sei $\hat{k} = (a_1, b_1, a_2, b_2)$ und $sig(\hat{k}, x) = y = (y_1, y_2)$
- Wegen $\hat{k} \in S(k)$ folgt $k = (\gamma_1, \gamma_2) = (\alpha^{a_1}\beta^{b_1}, \alpha^{a_2}\beta^{b_2})$ und daher gilt

$$\begin{aligned} \gamma_1\gamma_2^x &\equiv_p \alpha^{a_1}\beta^{b_1}(\alpha^{a_2}\beta^{b_2})^x \\ &\equiv_p \alpha^{a_1+xa_2}\beta^{b_1+xb_2} \\ &\equiv_p \alpha^{y_1}\beta^{y_2} \end{aligned}$$

$\square$

Anders gesagt gibt es im Fall $ver(k, x, y) = 0$ keinen Signierschlüssel $\hat{k} \in S(k)$ mit $sig(\hat{k}, x) = y$, d.h. $S(k, x, y) = \emptyset$

Das van Heyst-Pedersen Signaturverfahren

Das nächste Lemma zeigt, dass $S(k, x, y)$ im Fall $ver(k, x, y) = 1$ genau q Signierschlüssel enthält

Lemma

Zu jedem Paar (x, y) mit $ver(k, x, y) = 1$ gibt es genau q Signierschlüssel $\hat{k} \in S(k)$ mit $sig(\hat{k}, x) = y$

Beweis.

- Wir zeigen zuerst, dass $S(k)$ für jeden Verifikationsschlüssel $k = (\gamma_1, \gamma_2)$ genau q^2 Signierschlüssel enthält
- Ein Signierschlüssel $\hat{k} = (a_1, b_1, a_2, b_2)$ ist genau dann in $S(k)$, wenn er die beiden Kongruenzen

$$\alpha^{a_1}\beta^{b_1} \equiv_p \gamma_1$$
$$\alpha^{a_2}\beta^{b_2} \equiv_p \gamma_2$$

erfüllt

Beweis (Fortsetzung)

- Seien $c_1, c_2 \in \mathbb{Z}_q$ eindeutig bestimmte Exponenten mit $\gamma_1 \equiv_p \alpha^{c_1}$ und $\gamma_2 \equiv_p \alpha^{c_2}$
- Dann sind diese Kongruenzen äquivalent zu

$$\begin{aligned} a_1 + ab_1 &\equiv_q c_1 \\ a_2 + ab_2 &\equiv_q c_2 \end{aligned} \qquad \text{bzw.} \qquad \underbrace{\begin{pmatrix} 1 & a & 0 & 0 \\ 0 & 0 & 1 & a \end{pmatrix}}_{A} \begin{pmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \end{pmatrix} \equiv_q \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \qquad (*)$$

- Da A den Rang 2 hat, folgt $\|S(k)\| = q^2$ (siehe Übungen, Aufgabe 19)

Das van Heyst-Pedersen Signaturverfahren

Beweis (Fortsetzung)

- Sei nun (x, y) ein Paar mit $x \in \mathbb{Z}_q$ und $y = (y_1, y_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$
- Dann ist ein Signierschlüssel $\hat{k} = (a_1, b_1, a_2, b_2)$ genau dann in $S(k, x, y)$, wenn er die Kongruenzen

$$\begin{aligned} a_1 + ab_1 &\equiv_q c_1 \\ a_2 + ab_2 &\equiv_q c_2 \\ a_1 + xa_2 &\equiv_q y_1 \\ b_1 + xb_2 &\equiv_q y_2 \end{aligned} \quad \text{bzw.} \quad \underbrace{\begin{pmatrix} 1 & a & 0 & 0 \\ 0 & 0 & 1 & a \\ 1 & 0 & x & 0 \\ 0 & 1 & 0 & x \end{pmatrix}}_{A'} \begin{pmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \end{pmatrix} \equiv_q \underbrace{\begin{pmatrix} c_1 \\ c_2 \\ y_1 \\ y_2 \end{pmatrix}}_{s'} \quad (**)$$

  erfüllt
- Wir zeigen, dass sowohl die Matrix A' als auch die um den Vektor s' erweiterte Matrix $A's'$ den Rang $r = rang(A') = rang(A's') = 3$ haben
- Dies impliziert, dass das lineare Gleichungssystem $(**)$ genau $q^{4-r} = q$ Lösungen hat (siehe Übungen)

Das van Heyst-Pedersen Signaturverfahren

Beweis (Schluss)

- Seien $r_1, \dots, r_4$ die Zeilen von A'
- Dann gilt $rang(A') \geq 3$, da die Zeilen r_2, r_3, r_4 linear unabhängig sind, und $rang(A') \leq 3$, da $r_1 = r_3 + ar_4 - xr_2$ ist
- Damit hat $(**)$ im Falle der Lösbarkeit genau $q^{4-3} = q$ Lösungen
- Zum Nachweis der Lösbarkeit von (**) zeigen wir, dass die in A' bestehende Zeilenabhängigkeit $r_1 = r_3 + ar_4 - xr_2$ im Fall $ver(k, x, y) = 1$ auch für den Spaltenvektor s' auf der rechten Seite von (**) gilt:

$$\gamma_1 \gamma_2^x \equiv_p \alpha^{y_1} \beta^{y_2} \;\Rightarrow\; c_1 + xc_2 \equiv_q y_1 + ay_2 \;\Rightarrow\; c_1 \equiv_q y_1 + ay_2 - xc_2$$

- Da somit die Erweiterung der Matrix A' um den Spaltenvektor s' deren Rang im Fall $ver(k, x, y) = 1$ nicht erhöht, ist (**) in diesem Fall lösbar $\square$

Das van Heyst-Pedersen Signaturverfahren

Lemma

Für alle $x, x' \in \mathbb{Z}_q$ und $y = (y_1, y_2), y' = (y'_1, y'_2) \in \mathbb{Z}_q^2$ mit $x' \neq x$ gilt

$$\|S(k, x, y) \cap S(k, x', y')\| \leq 1$$

Im Fall $ver(k, x, y) = ver(k, x', y') = 1$ gilt sogar Gleichheit

Beweis.

- Die Bedingung $\hat{k} = (a_1, b_1, a_2, b_2) \in S(k, x, y) \cap S(k, x', y')$ ist äquivalent zu

$$\underbrace{\begin{pmatrix} 1 & a & 0 & 0 \\ 0 & 0 & 1 & a \\ 1 & 0 & x & 0 \\ 0 & 1 & 0 & x \\ 1 & 0 & x' & 0 \\ 0 & 1 & 0 & x' \end{pmatrix}}_{A''} \begin{pmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \end{pmatrix} = \underbrace{\begin{pmatrix} c_1 \\ c_2 \\ y_1 \\ y_2 \\ y'_1 \\ y'_2 \end{pmatrix}}_{s''} \quad (***)$$

- wobei wieder $\gamma_1 \equiv_p \alpha^{c_1}, \gamma_2 \equiv_p \alpha^{c_2}$ ist

Beweis (Fortsetzung)

- Wir zeigen, dass die Zeilen r_1, r_2, r_4, r_6 von A'' linear unabhängig sind und somit A'' den Rang $rang(A) = 4$ hat
- Daraus folgt, dass (***) höchstens eine Lösung hat
- Aus $l_1 r_1 + l_2 r_2 + l_4 r_4 + l_6 r_6 = \vec{0}$ folgt nämlich $l_1 = l_2 = 0$ und $l_4 + l_6 = 0$ sowie $l_4 x + l_6 x' = 0$, was $l_6 = -l_4$ sowie $l_4(x - x') = 0$ und somit wegen $x - x' \neq 0$ auch $l_4 = l_6 = 0$ impliziert
- Da auch die Zeilen $r_3, \ldots, r_6$ von A'' linear unabhängig sind, lässt sich $\hat{k}$ bei Kenntnis zweier Signaturen $y = sig(\hat{k}, x)$ und $y' = sig(\hat{k}, x')$ für zwei Texte $x \neq x'$ leicht bestimmen, d.h. es handelt sich um ein *One-time-Signaturverfahren*

Beweis (Schluss)

- Um die Lösbarkeit von (***) im Fall $ver(k, x, y) = ver(k, x', y') = 1$ nachzuweisen, zeigen wir, dass die in A'' bestehenden Zeilenabhängigkeiten $r_3 = r_1 + xr_2 - ar_4$ und $r_5 = r_1 + x'r_2 - ar_6$ auch für den Spaltenvektor s'' auf der rechten Seite von (***) gelten
- Aus $ver(k, x, y) = 1$ folgt

$$\gamma_1 \gamma_2^x \equiv_p \alpha^{y_1} \beta^{y_2} \ \Rightarrow \ c_1 + xc_2 \equiv_q y_1 + ay_2 \ \Rightarrow \ y_1 \equiv_q c_1 + xc_2 - ay_2$$

- Analog folgt aus $ver(k, x', y') = 1$ die Kongruenz

$$y_1' \equiv_q c_1 + x'c_2 - ay_2' \qquad \square$$