Kryptologie

Johannes Köbler

Institut für Informatik
Humboldt-Universität zu Berlin

WS 2022/23

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

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

Eine Signatur y heißt gültig für den Text x (unter k), falls $ver(k, x, y) = 1$ ist; andernfalls ist sie 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 MAC 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 MAC 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 Gegner 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 Gegner kennt neben k die Signaturen $y_i = sig(\hat{k}, x_i)$ für eine Reihe von Texten $x_1, \ldots, x_q$, auf deren Auswahl er keinen Einfluss hat, und versucht, eine Fälschung (x, y) mit $x \notin \{x_1, \ldots, x_q\}$ zu finden

Angriff bei frei wählbaren Texten (chosen document attack)
Der Gegner kann die Texte $x_1, \ldots, x_q$ selbst wählen, erhält die Signaturen aber erst, nachdem er alle Texte vorgelegt hat

Angriff bei adaptiv wählbaren Texten
Der Gegner kann die Wahl des Textes x_{i+1} von den Signaturen $y_1, \ldots, y_i$ abhängig machen

Erfolgskriterien für die Fälschung digitaler Signaturen

uneingeschränktes Fälschungsvermögen (total break)
: Der Gegner hat einen Weg gefunden, die Funktion $x \mapsto sig(\hat{k}, x)$ bei Kenntnis von k effizient zu berechnen

selektives Fälschungsvermögen (selective forgery)
: Der Gegner kann für beliebig vorgegebene Texte gültige Signaturen bestimmen (eventuell mit Hilfe des legalen Unterzeichners)

nichtselektives (existentielles) Fälschungsvermögen
: Der Gegner kann nur für bestimmte Texte x 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 geht, ob eine gegebene Zahl n prim ist, muss beim Faktorisierungsproblem im Fall, dass n zusammengesetzt ist, zudem ein nicht-trivialer Faktor gefunden werden
- Für die Sicherheit des RSA-Verfahrens ist es notwendig, 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

  $$K = \{(c,n) \mid \text{es gibt Primzahlen } p < q \text{ mit } n = pq \text{ und } c \in \mathbb{Z}^*_{\varphi(n)}\}$$

  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

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 $ed = z\varphi(n) + 1$
- Es reicht, die Kongruenz $x^{ed} \equiv_p x$ zu zeigen (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

- Wie beim RSA-Kryptosystem ist beim RSA-Signaturverfahren

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

  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

$$ver(e, n, x, y) = 1 \Leftrightarrow sig(d, n, x) = y$$

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 Gegner 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 Gegner 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 Gegner bereits die Signatur y' für einen beliebigen Text $x' \in \mathbb{Z}_n^*$ und kann er sich für den Text $x'' = xx'^{-1} \bmod n$ die Signatur y'' beschaffen,
  - so kann er 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 (z.B. kann man x durch den Text xx ersetzen und nur Texte dieser Form zulassen)
- Um auch längere Texte effizient signieren zu können, wird jedoch besser eine geeignete Hashfunktion h benutzt und nicht der gesamte Text x, sondern nur der Hashwert $h(x)$ signiert
  - Signaturerstellung: $sig_h(\hat{k}, x) := sig(\hat{k}, h(x))$
  - Verifikation: $ver_h(k, x, y) = 1 \Leftrightarrow ver(k, h(x)) = 1$

Das RSA-Signaturverfahren

Bei der Signaturerstellung benötigte Eigenschaften einer Hashfunktion h

- Die verwendete Hashfunktion h sollte die Einwegeigenschaft haben, da sonst der Gegner 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 Gegner kennt bereits ein Paar (x, y) mit $ver(k, h(x), y) = 1$
- Dann sollte h zumindest schwach kollisionsresistent sein, da sonst der Gegner ein x' mit $h(x') = h(x)$ berechnen und die Fälschung (x', y) generieren könnte
- Falls sich der Gegner für bestimmte von ihm selbst gewählte Texte x die zugehörige Signatur y beschaffen kann, so sollte h sogar kollisionsresistent sein
- Andernfalls könnte der Gegner 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
- Dieser wird mit $\log_{G,a}(b)$ bezeichnet
- Im Fall $G = \mathbb{Z}_m^*$ bezeichnen wir ihn auch kurz mit $\log_{m,a}(b)$

Diskrete Logarithmen

- Die Funktion $\exp_{m,a} : x \mapsto a^x \bmod m$ ist effizient berechenbar
- 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 $\mathrm{ord}_{11}(2) = |G| = 10$
- 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)

Die Gruppe $\mathbb{Z}_m^*$ ist genau für $m \in \{1, 2, 4, p^k, 2p^k \mid 2 < p \text{ prim}\}$ 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 $a \in \mathbb{Z}_{p-1} = \{0, \ldots, p-2\}$ und gibt $\beta = \alpha^a \bmod p$ öffentlich bekannt:
  Signierschlüssel: $\hat{k} = (p, \alpha, a)$
  Verifikationsschlüssel: $k = (p, \alpha, \beta)$
- Der Text- und Signaturenraum sind $X = \mathbb{Z}_{p-1}$ und $Y = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1} \setminus \{0\}$

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

$$sig(\hat{k}, x, z) = (\gamma, \delta) \in Y = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1} \setminus \{0\}$$

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

Das ElGamal-Signaturverfahren

Lemma

Eine Signatur (γ, δ) 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 $sig(\hat{k}, x, z) = (\gamma, \delta)$ gibt (d.h. $\gamma = \alpha^z \bmod p$ und $\delta = (x - a\gamma)z^{-1} \bmod p - 1$)

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 $\square$

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 $\text{ord}(\gamma) = \text{ord}(\alpha^z) = \text{ord}(\alpha)/\,\text{ggT}(\text{ord}(\alpha), z) = \text{ord}(\alpha) = p - 1$

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 $x = 100 \in X = \mathbb{Z}_{p-1} = \mathbb{Z}_{466}$ mit dem Signierschlüssel $\hat{k} = (p, \alpha, a) = (467, 2, 127)$ zu signieren,
  - wählt Alice die geheime Zufallszahl $z = 213 \in \mathbb{Z}_{p-1}^*$ ($\leadsto z^{-1} \bmod 466 = 431$) und
  - erhält wegen

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

    die Signatur $sig(\hat{k}, x, z) = (29, 51) \in Y = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1} \setminus \{0\}$
- 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

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

$\triangleleft$

Zur Sicherheit des ElGamal-Systems

- Falls der Gegner 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 Gegner 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 Gegner 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 Gegner 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 Gegner 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 Gegner 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

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 Gegner zu einer Signatur (x, γ, δ) 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 eindeutige Lösung der linearen Kongruenz

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

  berechnen, um

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

  zu bestimmen

Zur Sicherheit des ElGamal-Systems

- Kennt nämlich der Gegner zu einer Signatur (x, γ, δ) 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, γ, δ_1) und (x_2, γ, δ_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 $ord_p(\alpha) = q$:
  - Es gilt $ord(g^i) = \frac{ord(g)}{\text{ggT}(i, ord(g))} = \frac{p-1}{\text{ggT}((p-1)/q, p-1)} = q$ (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 die folgenden 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 die folgenden Schlüssel
  - Signierschlüssel: $\hat{k} = (p, q, \alpha, a)$, $a \in \mathbb{Z}_q$
  - Verifikationsschlüssel: $k = (p, \alpha, \beta)$, $\beta = \alpha^a \bmod p$
- Alice signiert einen Text $x \in X$ wie folgt
  - Signaturerstellung: Alice wählt 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 $\gamma = h(x\,bin(\alpha^z \bmod p))$ und $\delta = (z + a\gamma) \bmod q$ ist (ElGamal: $\gamma = \alpha^z \bmod p$ und $\delta = (x - a\gamma)z^{-1} \bmod p - 1$)
  - Der Signaturraum ist also $Y := \mathbb{Z}_q \times \mathbb{Z}_q$
- Bob verifiziert eine Signatur $y = (\gamma, \delta)$ für einen Text $x \in X$ wie folgt
  - Verifikation: $ver(k, x, \gamma, \delta) = 1 \Leftrightarrow h(x\,bin(\alpha^\delta \beta^{-\gamma} \bmod p)) = \gamma$ (ElGamal: $ver(k, x, \gamma, \delta) = 1 \Leftrightarrow \beta^\gamma \gamma^\delta \equiv_p \alpha^x$)
- Korrektheit der Verifikation: $\alpha^\delta \beta^{-\gamma} \equiv_p \alpha^{z+a\gamma} \alpha^{-a\gamma} \equiv_p \alpha^z \equiv_p \gamma$

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 $\gamma = h(x\,bin(2518)) \in \mathbb{Z}_q$
  - Unter der Annahme, dass $h(x\,bin(2518)) = 96$ ist, erhält Alice wegen

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

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

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 Alice 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 der Vorgang mit einer neuen Zufallszahl z wiederholt werden)
  - Verifikation: Es gilt $ver(k, x, \gamma, \delta) = 1$, falls $\beta^\gamma \gamma^\delta \equiv_p \alpha^x$ ist
- Der Übergang zu DSA basiert auf folgenden Modifikationen
  - 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 $\gamma^\delta \equiv_p \alpha^x \beta^\gamma$ (wegen $\alpha^{z(x+a\gamma)z^{-1}} \equiv_p \alpha^x \alpha^{a\gamma}$)
  - 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 $\gamma \equiv_p \alpha^{x\delta^{-1}} \beta^{\gamma\delta^{-1}}$

- Der Übergang zu DSA basiert auf folgenden Modifikationen
  - 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 $\gamma^\delta \equiv_p \alpha^x \beta^\gamma$ (wegen $\alpha^{z(x+a\gamma)z^{-1}} \equiv_p \alpha^x \alpha^{a\gamma}$)
  - 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 $\gamma \equiv_p \alpha^{x\delta^{-1}} \beta^{\gamma\delta^{-1}}$
  - 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

Der Digital Signature Algorithm (DSA)

- Beim DSA werden also die folgenden Schlüssel benutzt
  - Signierschlüssel: $\hat{k} = (p, q, \alpha, a)$, wobei $a \in \mathbb{Z}_q^*$ ist
  - Verifikationsschlüssel: $k = (p, q, \alpha, \beta)$ mit $\beta = \alpha^a \bmod p$
- Zudem gilt $X = \mathbb{Z}_q$ und $Y = \mathbb{Z}_q \times \mathbb{Z}_q^*$
  - Signaturerstellung: Um einen Text $x \in X$ zu signieren, wählt Alice zufällig eine geheime Zahl $z \in \mathbb{Z}_q^*$ und berechnet die Signatur

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

    (im Fall $\gamma = 0$ oder $\delta = 0$ muss ein neues z gewählt werden)
  - Verifikation:

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

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

Der Digital Signature Algorithm (DSA)

- Signierschlüssel: $\hat{k} = (p, q, \alpha, a)$, wobei $a \in \mathbb{Z}_q^*$ ist
- Verifikationsschlüssel: $k = (p, q, \alpha, \beta)$ mit $\beta = \alpha^a \bmod p$
- Signaturerstellung:

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

- Verifikation:

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

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

Die Korrektheit ergibt sich wegen

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

wie folgt:

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

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 $x = 22 \in \mathbb{Z}_q$ mit dem Signierschlüssel $\hat{k} = (p, q, \alpha, a)$ zu signieren, wählen wir die geheime Zufallszahl $z = 50 \in \mathbb{Z}_q^*$, d.h. $z^{-1} \bmod q = 99$, und erhalten dann

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

d.h. $sig(\hat{k}, z, x) = (94, 97)$ mit $\hat{k} = (p, q, \alpha, a) = (7879, 101, 170, 75)$

Beispiel (Fortsetzung)

- Um diese Signatur zu prüfen berechnen wir:

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

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

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 Alice zufällig eine 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

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:

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

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 $(8,7)$ mit dem Verifikationsschlüssel $k = (E, q, A, B)$ für den Text x mit dem Hashwert $h(x) = 4$ 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 $b = 0, 1$

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:

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

◁

Die One-time-Signatur von Lamport

- Für digitale Signaturverfahren kann man ähnlich wie bei MACs unterschiedliche Angriffsszenarien und Erfolgskriterien betrachten
- Im Folgenden nehmen wir an, dass der Gegner die Texte x_i, deren Signaturen y_i bekannt werden, adaptiv wählen kann
- Zudem verlangen wir von dem Gegner nur, eine existentielle Fälschung (x, y) zu finden

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 existentielle 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 $\geq \varepsilon/2$ ein f-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 $(i, b) \in [\ell] \times \{0, 1\} \setminus \{(j, a)\}$ **do**
  wähle zufällig $u_{i,b} \in_R U$ und setze $v_{i,b} := f(u_{i,b})$
$k := \binom{v_{1,0} \ldots v_{\ell,0}}{v_{1,1} \ldots v_{\ell,1}}$
$(x_1 \ldots x_\ell, u_1, \ldots, u_\ell) =:$ `Lamport-Fälschung`(k)
if $f(u_j) = v$ **then output**(u_j) **else output**$(?)$

Die One-time-Signatur von Lamport

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 $\mathcal{V}$, $\mathcal{J}$, $\mathcal{A}$, $\mathcal{K}$ und $(\mathcal{X}, \mathcal{Y})$
- Dann müssen wir zeigen, dass $\mathcal{U}_{\mathcal{J}}$ mit Wahrscheinlichkeit mindestens $\varepsilon/2$ ein f-Urbild von $\mathcal{V}$ ist, wobei $\mathcal{V}$ die Wahl von $v = f(u)$ für ein zufällig gewähltes $u \in_R U$ beschreibt
- Da die Verteilung von $\mathcal{K}$ identisch zur Schlüsselgenerierung der Lamport-Signatur und `Lamport-Fälschung` ein $(\varepsilon, 0)$-Fälscher ist, folgt

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

- Da zudem $\mathcal{K}$ (und damit auch $(\mathcal{X}, \mathcal{Y})$) unabhängig von $(\mathcal{J}, \mathcal{A})$ und auch $\mathcal{J}$ und $\mathcal{A}$ unabhängig voneinander sind, ist $\mathcal{A}$ von $(\mathcal{J}, \mathcal{K}, \mathcal{X}, \mathcal{Y})$ und damit auch von $\mathcal{X}_{\mathcal{J}}$ unabhängig

Die One-time-Signatur von Lamport

Beweis (Schluss)

- Sei p die Erfolgswahrscheinlichkeit von `Lamport-Urbild` bei Eingabe $\mathcal{V}$
- Wegen

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

folgt nun

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

□

Die One-time-Signatur von Lamport

- Als nächstes untersuchen wir die Sicherheit der Lamport-Signatur gegen einen Angriff, bei dem sich der Gegner eine gültige Signatur y' für einen von ihm gewählten Text x' beschaffen kann (indem er z.B. den Text x' von Alice signieren lässt)
- Würde es ihm nun gelingen, für einen Text $x \neq x'$ eine gültige Signatur y zu finden, so könnte er (x', y') durch die Fälschung (x, y) ersetzen

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

Falls für die zugehörige one-time Signatur ein $(\varepsilon, 1)$-Fälscher $\texttt{Lamport-Fälschung}'(k)$ 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 Algorithmus $\texttt{Lamport-Urbild}'$ 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 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 $(i, b) \in [\ell] \times \{0, 1\} \setminus \{(j, a)\}$ **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 `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 $p' = \Pr[\texttt{Lamport-Urbild}'(\mathcal{V}) \neq ?]$ die Erfolgswahrscheinlichkeit von $\texttt{Lamport-Urbild}'$ bei Eingabe $\mathcal{V}$
- $\texttt{Lamport-Urbild}'$ kann die Frage von $\texttt{Lamport-Fälschung}'(k)$ nach der Signatur von x' 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 $\texttt{Lamport-Fälschung}'(k)$ eventuell abgebrochen wird (und die Abbruchbedingung von (j, a) abhängt), können wir nicht mehr davon ausgehen, dass diese Simulation mit Wk ε eine Fälschung (x, y) liefert und $(\mathcal{X}, \mathcal{Y})$ unabhängig von $(\mathcal{J}, \mathcal{A})$ ist
- Durch eine einfache Modifikation von $\texttt{Lamport-Urbild}'$ erhalten wir jedoch eine Prozedur $\texttt{Lamport-Urbild}^*$ ohne Eingabe, deren Ausgabeverhalten mit der von $\texttt{Lamport-Urbild}'(\mathcal{V})$ identisch ist
- Zudem können wir zeigen, dass $\Pr[\texttt{Lamport-Urbild}^* \neq ?] \geq \varepsilon/2\ell$ ist:

Die One-time-Signatur von Lamport

Beweis (Fortsetzung)

- Durch eine einfache Modifikation von `Lamport-Urbild`$'$ erhalten wir jedoch eine Prozedur `Lamport-Urbild`* ohne Eingabe, deren Ausgabeverhalten mit der von `Lamport-Urbild`$'(\mathcal{V})$ identisch ist
- Zudem können wir zeigen, dass $\Pr[\texttt{Lamport-Urbild}^* \neq ?] \geq \varepsilon/2\ell$ ist:

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 $\texttt{Lamport-Urbild}'(v)$ wählt sich $\texttt{Lamport-Urbild}^*$ also die Eingabe $v = v_{j,a}$ gemäß der Verteilung von $\mathcal{V}$ selbst und kennt daher auch ein Urbild $u_{j,a}$ von $v_{j,a}$
- Somit kann $\texttt{Lamport-Urbild}^*$ bei der Simulation von $\texttt{Lamport-Fälschung}'(k)$ die Frage nach der Signatur von x' auch im Fall $x'_j = a$ beantworten
- Die Bedingung für die Ausgabe von u_j bzw. von ? ist jedoch bei beiden Prozeduren dieselbe, d.h. die Ausgabe von $\texttt{Lamport-Urbild}^*$ hat dieselbe Verteilung wie die von $\texttt{Lamport-Urbild}'(\mathcal{V})$ und somit gilt

$$p' = \Pr[\texttt{Lamport-Urbild}'(\mathcal{V}) \neq ?] = \Pr[\texttt{Lamport-Urbild}^* \neq ?]$$

- Der einzige Unterschied zwischen beiden Prozeduren ist also, dass $\texttt{Lamport-Urbild}^*$ die Eingabe $v = v_{j,a}$ zusammen mit dem f_k-Urbild $u_{j,a}$ selbst zufällig wählt und immer wenn $\texttt{Lamport-Urbild}'$ in Zeile 6 ein Fragezeichen ausgibt, $\texttt{Lamport-Urbild}^*$ dies erst in Zeile 7 tut

Die One-time-Signatur von Lamport

Beweis (Schluss)

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

$$\begin{aligned} p' &= \Pr[f(\mathcal{U}_{\mathcal{J}}) = \mathcal{V}_{\mathcal{J},\mathcal{A}} \wedge \mathcal{X}'_{\mathcal{J}} \neq \mathcal{A}] \\ &\geq \Pr[ver(\mathcal{K}, \mathcal{X}, \mathcal{Y}) = 1 \wedge \mathcal{X}_{\mathcal{J}} = \mathcal{A} \wedge \mathcal{X}'_{\mathcal{J}} \neq \mathcal{A}] \\ &= \underbrace{\Pr[ver(\mathcal{K}, \mathcal{X}, \mathcal{Y}) = 1]}_{\varepsilon} \underbrace{\Pr[\mathcal{X}'_{\mathcal{J}} \neq \mathcal{X}_{\mathcal{J}} = \mathcal{A} \mid ver(\mathcal{K}, \mathcal{X}, \mathcal{Y}) = 1]}_{1/2\ell} \\ &= \varepsilon/2\ell \end{aligned}$$

□

- 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 $\hat{k}$ das Urbild u mit $f_k(u) = v$ zu finden (genauer: jedem effizienten Gegner 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 $\geq 2^{1024}$ haben, um eine ausreichend große Sicherheit zu erreichen, wogegen die benutzten Hashfunktionen meist 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
- Die Fälschungssicherheit der FDH-Signatur lässt sich aus der Falltüreigenschaft von $\mathcal{F}$ herleiten, sofern für h eine Zufallsfunktion $G : \{0,1\}^* \to U$ benutzt wird (Zufalls-Orakel-Modell, ZOM)
- Wie wir gesehen haben, modelliert G eine Hashfunktion mit optimalen kryptografischen Eigenschaften, da die Zufallsvariablen $\mathcal{U}_x = G(x)$, $x \in \{0,1\}^*$, auf U gleichverteilt und stochastisch unabhängig sind
- Zudem füllt der Wertebereich von G den gesamten Definitionsbereich der Funktionen f_k aus (Full Domain Hash, FDH)
- Wir betrachten zuerst den Fall einer existentiellen Fälschung bei bekanntem Verifikationsschlüssel, d.h. der Gegner versucht, bei Kenntnis von k eine Fälschung (x, y) mit $ver(k, x, y) = 1$ zu finden ohne eine Signatur für einen Text 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 $q \geq 0$ verschiedenen Fragen $x_1, \ldots, x_q$ an G stellt
- Es ist klar, dass ein solcher Angriff im Fall $x \notin \{x_1, \ldots, x_q\}$ eine Erfolgswahrscheinlichkeit von exakt $\varepsilon = 1/|U|$ hat
- Die Erfolgswahrscheinlichkeit $\varepsilon = 1/|U|$ kann der Gegner trivialerweise bereits mit $q = 0$ Fragen an G erreichen, indem er ein beliebiges Paar $(x, y) \in \{0, 1\}^* \times U$ ausgibt
- Interessant ist demnach nur der Fall, dass der Gegner ein Paar (x, y) mit $ver(k, x, y) = 1$ mit einer Erfolgswahrscheinlichkeit $\varepsilon > 1/|U|$ findet, was $x \in \{x_1, \ldots, x_q\}$ und somit $q \geq 1$ impliziert
- In diesem Fall betrachten wir 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 durch
$v_i = v$, falls $i = j$ ist, 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 $\geq \varepsilon/q$ das f_k-Urbild von v aus

Da sich das f_k-Urbild von v mit Wahrscheinlichkeit $1/|U|$ erraten lässt, ist die Aussage des Satzes nur im Fall $\varepsilon > q/|U|$ interessant

Beweis.

- Da die Eingabe $v = v_j$ und die Strings v_i, $i \neq j$, unabhängig unter Gleichverteilung aus U gewählt werden, sind die Antworten auf die Fragen $x_1, \ldots, x_q$ unabhängig gleichverteilt, was dem ZOM entspricht
- Daher liefert die Simulation von $\texttt{FDH-Fälschung}(k)$ für einen zufällig generierten Verifikationsschlüssel k mit Wahrscheinlichkeit ε eine Fälschung (x, y) mit $f_k(y) = G(x)$, d.h. es gilt $\Pr[f_{\mathcal{K}}(\mathcal{Y}) = G(\mathcal{X})] = \varepsilon$, wobei wir wie üblich die Zufallsvariablen $\mathcal{J}$, $\mathcal{K}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{X}$, $\mathcal{X}_1, \ldots, \mathcal{X}_q$ benutzen, um die Wahl von j, k, u, v, x, $x_1, \ldots, x_q$ zu beschreiben
- Wir wollen zeigen, dass $\Pr[f_{\mathcal{K}}(\mathcal{Y}) = \mathcal{V}] \geq \varepsilon/q$ ist
- Wegen $x \in \{x_1, \ldots, x_q\}$ gilt $x = x_i$ für ein $i \in \{1, \ldots, q\}$ und die Verifikationsbedingung $f_k(y) = G(x)$ impliziert die Gleichheit $f_k(y) = G(x_i) = v_i$, d.h. y ist das f_k-Urbild von v_i

Beweis (Fortsetzung)

- Folglich gilt die Implikation

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

  und es folgt

$$\Pr[f_{\mathcal{K}}(\mathcal{Y}) = \mathcal{V}] \geq \Pr[f_{\mathcal{K}}(\mathcal{Y}) = G(\mathcal{X}) \wedge \mathcal{J} = \mathcal{I}]$$

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

Beweis (Schluss)

- Daher erhält `FDH-Fälschung` weder durch k noch durch die Antworten $v_1, \ldots, v_q$ irgendeine Information über j (die Zufallsvariablen $\mathcal{K}$ und $\mathcal{V}_1, \ldots, \mathcal{V}_q$ sind also stochastisch unabhängig von $\mathcal{J}$)
- Folglich ist neben der Eingabe $\mathcal{K}$ und den Antworten $\mathcal{V}_1, \ldots, \mathcal{V}_q$ auch die Ausgabe $(\mathcal{X}, \mathcal{Y})$ und somit auch die Zufallsvariable $\mathcal{I}$, die den Index $i \in \{1, \ldots, q\}$ mit $x = x_i$ bestimmt, stochastisch unabhängig von $\mathcal{J}$
- Nun folgt

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

□

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 Gegner, 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

- Sei $\texttt{FDH-Fälschung}'$ 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 $x_1, \dots, x_q$ den Wert $G(x_i)$ oder die Signatur $sig(\hat{k}, x_i) = f_{\hat{k}}(G(x_i))$ erfragt
- Dabei nehmen wir wieder an, dass $\texttt{FDH-Fälschung}'$ den G-Wert von x erfragt (aber natürlich nicht die Signatur von x)
- Zudem nehmen wir an, dass $\texttt{FDH-Fälschung}'$ 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 $\texttt{FDH-Fälschung}'(k)$ 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 $\geq \varepsilon/q$ das f_k-Urbild von v bestimmen

Für den Beweis benutzen wir folgende **Prozedur** $\texttt{FDH-Invert}'(k, v)$:

---

wähle zufällig $j \in_R \{1, \ldots, q\}$
simuliere $\texttt{FDH-Fälschung}'(k)$ und beantworte dabei jede Frage x_i an G
durch $v_i = v$, falls $i = j$ ist, 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

- Da die Frage nach der Signatur von x_i im Fall $i = j$ unbeantwortet bleibt, können wir nicht davon ausgehen, dass $\texttt{FDH-Fälschung}'(k)$ in Zeile 6 mit Wahrscheinlichkeit ε eine Fälschung (x, y) mit $f_k(y) = G(x)$ findet
- Wir können aber eine Prozedur $\texttt{FDH-Invert}^*(k)$ angeben, die nur k als Eingabe erhält, so dass die Ausgaben von $\texttt{FDH-Invert}^*(\mathcal{K})$ und von $\texttt{FDH-Invert}'(\mathcal{K}, \mathcal{V})$ identisch verteilt sind, was

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

  impliziert
- Zudem lässt sich zeigen, dass

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

  gilt (siehe Übungen)