

Kryptologie

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2020/21

Eigenschaften von handschriftlichen Signaturen

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

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

Lösung:

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

Definition

Ein **digitales Signaturverfahren** besteht aus

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

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

Digitale Signaturverfahren

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

Angriff bei bekanntem Verifikationsschlüssel (key-only attack)

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

Angriff bei bekannter Signatur (known signature attack)

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

Angriff bei frei wählbaren Texten (chosen document attack)

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

Angriff bei adaptiv wählbaren Texten

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

uneingeschränktes Fälschungsvermögen (total break)

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

selektives Fälschungsvermögen (selective forgery)

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

nichtselektives (existentielles) Fälschungsvermögen

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

Das RSA-Kryptosystem

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

Schlüsselgenerierung

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

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

Privater Schlüssel: $k'_X = (d, n)$

Ver- und Entschlüsselung

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

Ver- und Entschlüsselung

Der folgende Satz garantiert die Korrektheit des RSA-Systems

Satz

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

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

Beweis.

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

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



Das RSA-Signaturverfahren

Definition

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

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

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

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

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

Satz

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

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

Der Beweis folgt direkt aus der Korrektheit des RSA-Kryptosystems

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

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

Bei der Signaturerstellung benötigte Eigenschaften einer Hashfunktion h

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

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

Definition

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

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

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

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

Beispiel

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

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

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

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

Definition

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

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

Satz (Gauß)

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

- Das **Signaturverfahren von ElGamal** (1985) ist wie das gleichnamige asymmetrische Kryptosystem probabilistisch und beruht wie dieses auf dem diskreten Logarithmus
- Sei p eine große Primzahl und 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

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

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

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

Das ElGamal-Signaturverfahren

Lemma

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

Beweis.

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

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

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

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

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

- Falls der Angreifer in der Gruppe \mathbb{Z}_p^* den diskreten Logarithmus von 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 $\alpha^x \equiv \beta^{\gamma} \gamma^{\delta} \pmod{p}$ 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 $\alpha^x \equiv \beta^{\gamma} \gamma^{\delta} \pmod{p}$ zu finden
 - Hierfür ist kein effizientes Verfahren bekannt

- Der Angreifer versucht, zu einem gegebenen Text x gleichzeitig passende Zahlen γ und δ mit $\alpha^x \equiv \beta^\gamma \gamma^\delta \pmod{p}$ zu finden
 - Auch hierfür ist kein effizientes Verfahren bekannt
- Versucht der Angreifer bei einem **nichtselektiven Angriff**, zuerst 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

- 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 $\text{ggT}(v, p-1) = 1$ für den Text $x = u\delta \bmod p-1$ mittels $\delta = -\gamma v^{-1} \bmod p-1$ erfüllen lässt
 - Bei Wahl von $v = 1$ erhalten wir z.B. die gültige Signatur $(\gamma, \delta) = (\alpha^u \beta \bmod p, -\alpha^u \beta \bmod p-1)$ für den Text $x = u\delta \bmod p-1$, wobei $u \in \mathbb{Z}_{p-1}$ beliebig gewählt werden kann

Bemerkung

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

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

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

als eindeutige Lösung der Kongruenz

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

berechnen

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

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

berechnen

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

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

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

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

- Da die Primzahl p beim ElGamal-Signaturverfahren mindestens eine 512-Bit-Zahl (besser 1024-Bit-Zahl) sein sollte, beträgt die Signaturlänge 1024 bzw 2048 Bit
- Folgende Variante des ElGamal-Signaturverfahrens, die als eine Vorstufe zum DSA betrachtet werden kann, wurde von Schnorr vorgeschlagen
- Die zugrunde liegende Idee ist folgende:
 - Indem wir für 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 $\text{ord}_p(\alpha) = q$
 - da $\text{ord}(g^i) = \frac{\text{ord}(g)}{\text{ggT}(i, \text{ord}(g))} = \frac{p-1}{\text{ggT}((p-1)/q, p-1)} = q$ ist (s. Übungen)
- Weiter sei $h : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ eine Hashfunktion, die jedem Text $x \in X = \{0, 1\}^*$ einen Hashwert in \mathbb{Z}_q zuordnet
- Das Schnorr-Verfahren benutzt folgende Schlüssel:
 - Signierschlüssel: $\hat{k} = (p, q, \alpha, a), a \in \mathbb{Z}_q$
 - Verifikationsschlüssel: $k = (p, \alpha, \beta), \beta = \alpha^a \text{ mod } p$

Das Schnorr-Signaturverfahren

- Das Schnorr-Verfahren benutzt folgende Schlüssel:

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

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

- Signaturerstellung**

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

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

wobei $\gamma = h(x \text{bin}(\alpha^z \bmod p))$ 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 $\text{ver}(k, x, \gamma, \delta) = 1$, falls $h(x \text{bin}(\alpha^\delta \beta^{-\gamma} \bmod p)) = \gamma$

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

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

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

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 $h(xbin(2518)) = 96$



- 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

- 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

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

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

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

- 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 $\text{ord}_p(\alpha) = q$
- 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

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

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

- Die Verifikationsbedingung ist

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

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

- Die Korrektheit ergibt sich wie folgt:

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

Beispiel

- Seien $q = 101$, $p = 78q + 1 = 7879$, $g = 3$ ($\text{ord}_p(3) = p - 1$)

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

- 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

$$\begin{aligned}\gamma &= (170^{50} \bmod 7879) \bmod 101 \\ &= 2518 \bmod 101 \\ &= 94\end{aligned}$$

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

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

Beispiel (Fortsetzung)

- Um diese Signatur zu prüfen berechnen wir:

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

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

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

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 $h: \{0, 1\}^* \rightarrow \mathbb{Z}_q$ 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 $\text{sig}(\hat{k}, x, z) = (\gamma, \delta)$ mit

$$(u, v) := zA$$

$$\gamma := u \bmod q$$

$$\delta := (h(x) + m\gamma)z^{-1} \bmod q$$

- Hierbei wird u als eine Zahl in $\{0, \dots, p^n - 1\}$ 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

$$e := h(x)\delta^{-1} \bmod q$$

$$d := \gamma\delta^{-1} \bmod q$$

$$(u, v) := eA + dB$$

- Korrektheit der Verifikation beim ECDSA:

$$(u, v) = eA + dB$$

$$= (h(x)\delta^{-1})A + (\gamma\delta^{-1})mA$$

$$= (h(x) + m\gamma)\delta^{-1}A$$

$$= zA \text{ (da } (h(x) + m\gamma)\delta^{-1} \equiv_q z)$$

Beispiel

- Sei E über \mathbb{Z}_{11} definiert durch $y^2 = x^3 + x + 6$
- Wir wählen $A = (2, 7)$, $m = 7 \rightarrow p = 11, q = 13, B = 7A = (7, 2)$
- 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

$$(u, v) := zA = 3 \cdot (2, 7) = (8, 3)$$

$$\gamma := u \bmod q = 8$$

$$\delta := (4 + 7 \cdot 8)3^{-1} \bmod 13 = 7$$

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

Beispiel (Fortsetzung)

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

- berechnet Bob

$$e := h(x)\delta^{-1} \bmod q = 4 \cdot 7^{-1} \bmod 13 = 4 \cdot 2 \bmod 13 = 8$$

$$d := \gamma\delta^{-1} \bmod q = 8 \cdot 2 \bmod 13 = 3$$

$$(u, v) := eA + dB = 8 \cdot (2, 7) + 3 \cdot (7, 2) = (8, 3)$$

- 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 $f : U \rightarrow V$ 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 $i = 1, \dots, \ell$ und $b = 0, 1$ aus U gewählt und der **Signierschlüssel** $\hat{k} = \begin{pmatrix} u_{1,0} \dots u_{\ell,0} \\ u_{1,1} \dots u_{\ell,1} \end{pmatrix}$ gebildet
- Der zugehörige **Verifikationsschlüssel** ist dann $k = \begin{pmatrix} v_{1,0} \dots v_{\ell,0} \\ v_{1,1} \dots v_{\ell,1} \end{pmatrix}$ mit $v_{i,b} = f(u_{i,b})$ für alle $i = 1, \dots, \ell$ 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

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

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

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

Die One-time-Signatur von Lamport

Beispiel

- Wir wählen als Einwegfunktion eine Funktion der Form $f : \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$ 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

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

- Für diese Signatur $y = (u_1, u_2, u_3)$ ist $\text{ver}(k, x, y) = 1$, 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

- 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 $y_i = \text{sig}(\hat{k}, x_i)$ von q Texten x_1, \dots, x_q fragt und
 - mit Wahrscheinlichkeit mindestens ε eine Fälschung (x, y) mit $\text{ver}(k, x, y) = 1$ und $x \notin \{x_1, \dots, x_q\}$ ausgibt

Die One-time-Signatur von Lamport

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

Falls für die zugehörige one-time Signatur ein $(\varepsilon, 0)$ -Fälscher $\text{LAMPORT-FÄLSCHUNG}(k)$ 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 $\text{LAMPORT-URBILD}(v)$:

Prozedur Lamport-Urbild(v)

- 1 wähle zufällig ein Indexpaar (j, a) und setze $v_{j,a} := v$
 - 2 **for all** $(i, b) \in [\ell] \times \{0, 1\} \setminus \{(j, a)\}$ **do**
 - 3 wähle zufällig $u_{i,b} \in_R U$ und setze $v_{i,b} := f(u_{i,b})$
 - 4 $k := \begin{pmatrix} v_{1,0} \dots v_{\ell,0} \\ v_{1,1} \dots v_{\ell,1} \end{pmatrix}$
 - 5 $(x_1 \dots x_\ell, (u_1, \dots, u_\ell)) =: \text{LAMPORT-FÄLSCHUNG}(k)$
 - 6 **if** $f(u_j) = v$ **then output**(u_j) **else output**(?)
-