Fortgeschrittene Techniken der Kryptographie

Jürgen Fuß

Episode 7: Digitale Signaturen und diskrete Logarithmen

HAGENBERG | LINZ | STEYR | WELS

FH OBERÖSTERREICH UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Digital Signature Algorithm (DSA)

Disclaimer: DSA ist obsolet!

- Das DSA-Signaturverfahren ist seit Februar 2023 (Version FIPS 186-5) nicht mehr als standardisiertes Verfahren zur Erstellung von Signaturen zugelassen.
- Allerdings dürfen mit diesem Verfahren erstellte Signaturen weiterhin überprüft werden.
- Bis Februar 2024 ist DSA nach dem bis dahin noch gültigen FIPS 186-4 noch als Signaturverfahren zulässig.
- Wir betrachten das Verfahren dennoch genauer, weil es Basis für die neueren Signaturverfahren ECDSA und Ed25519 ist.

DSA – Domain-Parameter und Schlüsselerzeugung

Setup: Als Hashfunktion wird eine Hashfunktion H aus der SHA-x-Familie verwendet. Eine L Bit große Primzahl p wird vereinbart, so dass $p-1$ einen N Bit großen Primfaktor ω besitzt, weiterhin ein Element g der Ordnung ω in der Gruppe $\mathbb{Z}_p^*$.

Schlüsselerzeugung: Alice wählt zufällig eine Zahl $\alpha \in \mathbb{Z}_\omega$. Sie berechnet

Pub Priv.

$$\underset{\text{Pub}}{A} := g^{\overset{\text{Priv.}}{\alpha}} \bmod p$$

und veröffentlicht ihren Public Key A. Den Private Key α hält sie geheim.

DSA – Signieren und verifizieren

Signieren: Um zu signieren, wählt Alice zufällig eine Zahl $k \in \mathbb{Z}_\omega$. Dann berechnet sie die Signatur (r, s) der Nachricht m als

$$r := (g^k \bmod p) \bmod \omega,$$
$$s := k^{-1}(H(m) + \alpha r) \bmod \omega.$$

Verifizieren: Will Bob die Signatur überprüfen, so führt er die folgenden Schritte durch:

1. Er prüft: Ist $1 \le r < \omega$ und $1 \le s < \omega$?
2. Er berechnet $x := s^{-1} \cdot H(m) \bmod \omega$ und $y := s^{-1} \cdot r \bmod \omega$.
3. Er prüft: Ist $r = (g^x \cdot A^y \bmod p) \bmod \omega$?

$g^k \bmod \omega$

Korrektheit des Verfahrens

Eine gültige Unterschrift besteht alle Tests:

1. $1 \leq r, s < \omega$:
   Für r und s wurde modulo ω gerechnet, r und s sind also kleiner als ω. Die Werte r und/oder s könnten (theoretisch) gleich 0 sein.

   $s = 0$: In Schritt 2 würde ein Fehler beim Berechnen von s^{-1} auftreten.

   $r = 0$: Dann würde sich $y = 0$ ergeben und damit die Verifikation vom Public Key A unabhängig werden, ein Sicherheitsproblem. Um diese Probleme zu vermeiden, wird in diesen Fällen die Signatur gleich ungültig. Zufällig tritt dieser Fall praktisch nicht auf, weil die Wahrscheinlichkeit, dass r oder s den Wert 0 annehmen $1/\omega$, also vernachlässigbar klein ist.

Eine gültige Unterschrift besteht alle Tests:

3.

$$
\begin{aligned}
g^x \cdot A^y &= g^{s^{-1} \cdot H(m)} \cdot (g^{\alpha})^{s^{-1} r} \\
&= g^{s^{-1}(H(m) + \alpha r)} \\
&= g^k \pmod{p}.
\end{aligned}
$$

$s^{-1}(H(m) + \alpha r) = k \quad | \cdot s$

$H(m) + \alpha r = ks \quad | \cdot k^{-1}$

$k^{-1}(H(m) + \alpha r) = s$

DSA – Parameterwahl (FIPS 186-4)

- Die Wahl eines L Bit langen p mit großem Primfaktor ω von $p-1$ verhindert Index-Calculus-Angriffe.
- Die Wahl eines N Bit langen ω verhindert das Berechnen von diskreten Logarithmen mittels Durchprobieren (z. B. Baby-Step-Giant-Step).
- Die Primalität von ω verhindert schließlich das Berechnen von diskreten Logarithmen mit der Pohlig-Hellman-Methode.
- Im Standard erlaubte Werte für (L, N) sind u.a. $(3072, 256)$ für 128 Bit Sicherheit und $(15360, 512)$ für 256 Bit Sicherheit. Entsprechende Hashfunktionen sind dann zu verwenden: SHA-256 für 128 und SHA-512 für 256 Bit Sicherheit.

DSA – Performance (Sicherheitsniveau 128 Bit)

- Private Keys sind 256 bit lang, Public Keys 3072 bit.
- Signaturen sind nur 512 Bit lang, also (bei gleichem Sicherheitsniveau) sind RSA-Signaturen (3072 bit) sechsmal so lang.
- Der wesentliche Aufwand beim Signieren entsteht durch einmaliges Potenzieren mit einem 256 bit langen Exponenten. Im Vergleich zu RSA (3072-bit-Exponent) ist DSA damit wesentlich effizienter (Faktor 12) beim Signieren.
- Beim Prüfen einer Signatur tauchen zwei 256 bit lange Exponenten auf. Dank extrem kurzem Exponenten ist RSA beim Prüfen von Signaturen (17 Multiplikationen) dennoch bedeutend schneller (ca. Faktor 45) als DSA.

Erinnerung an GDK: RSA vs. DSA

Für ein Sicherheitsniveau von 128 Bit ergibt sich:

	RSA	**DSA**	**Verhältnis**
Domain Parameter		3000 Bit	
Größe des Public Keys	3000 Bit	3000 Bit	1:1
Größe einer Signatur	3000 Bit	512 Bit	6:1
Multiplikationen pro Signatur	$\approx 1,5 \cdot 3000$	$\approx 1,5 \cdot 256$	12:1
Multiplikationen pro Verifikation	17	$\approx 2 \cdot 1,5 \cdot 256$	1:45

Sehr deutlich sind die Unterschiede in den Aufwänden für das Signieren und das Verifizieren.

DSA – Performance (Sicherheitsniveau 256 Bit)

- Private Keys sind 512 bit lang, Public Keys 15360 bit.
- Signaturen sind nur 1024 Bit lang, also (bei gleichem Sicherheitsniveau) sind RSA-Signaturen (15360 bit) 15-mal so lang.
- Der wesentliche Aufwand beim Signieren entsteht durch einmaliges Potenzieren mit einem 512 bit langen Exponenten. Im Vergleich zu RSA (15360-bit-Exponent) ist DSA damit wesentlich effizienter (Faktor 30) beim Signieren.
- Beim Prüfen einer Signatur tauchen zwei 512 bit lange Exponenten auf. Dank extrem kurzer Exponenten ist RSA beim Prüfen von Signaturen (17 Multiplikationen) dennoch bedeutend schneller (ca. Faktor 90) als DSA.

Attacken auf DSA

Nonce Reuse (1)

Es ist nicht ratsam, sich Arbeit zu sparen, indem man beim Signieren jedes Mal die selbe Nonce k verwendet. Werden zwei verschiedene Nachrichten m_1 und m_2 mit dem selben Wert für k signiert, so lässt sich aus den beiden Signaturen[1] (r, s_1) und (r, s_2) der Private Key α berechnen.

$$\Rightarrow r = (g^k \bmod p) \bmod q$$

$$s_1 = k^{-1}(H(m_1) + \alpha \cdot r) \bmod q$$

$$s_2 = \qquad m_2$$

[1] Beachte, dass sich für das gleiche k auch zweimal der selbe Wert r ergibt. Nonce Reuse ist also auch sehr einfach zu erkennen.

Nonce Reuse (2)

Es ist dann (modulo ω)

$$
\begin{aligned}
s_1 - s_2 &= k^{-1}(H(m_1)+\alpha r) - k^{-1}(H(m_2)+\alpha r)\\
&= k^{-1}(H(m_1)+\cancel{\alpha r} - H(m_2) - \cancel{\alpha r})\\
s_1 - s_2 &= k^{-1}(H(m_1)-H(m_2)). \qquad /\cdot k \quad /:(s_1-s_2)\\
k &= (s_1-s_2)^{-1}(H(m_1)-H(m_2)) \pmod{\omega}.
\end{aligned}
$$

Nonce Reuse (2)

Es ist dann (modulo ω)

$$\begin{aligned} s_1 - s_2 &= k^{-1}(H(m_1) + \alpha r) - k^{-1}(H(m_2) + \alpha r) \\ &= k^{-1}(H(m_1) + \alpha r - H(m_2) - \alpha r) \\ &= k^{-1}(H(m_1) - H(m_2)). \\ k &= (s_1 - s_2)^{-1}(H(m_1) - H(m_2)) \pmod{\omega}. \end{aligned}$$

Kennt man aber k, so lässt sich α einfach berechnen.

$$\begin{aligned} s_1 &= k^{-1}(H(m_1) + \alpha r) \quad / \cdot k \\ k s_1 &= H(m_1) + \alpha r \\ \alpha r &= k s_1 - H(m_1) \quad / : r \\ \alpha &= \underline{r^{-1}(k s_1 - H(m_1)) \pmod{\omega}}. \end{aligned}$$

Schnorr-Signaturen

Schnorr-Signaturen – Domain-Parameter und Schlüsselerzeugung

Setup: Als Domain-Parameter werden eine Hashfunktion H, eine Gruppe $\mathbb{G}$ und ein Element $g \in \mathbb{G}$ der Ordnung ω vorbereitet.

Schlüsselerzeugung: Alice wählt zufällig eine Zahl $\alpha \in \mathbb{Z}_\omega$. Sie berechnet $A := g^\alpha$ und veröffentlicht ihren Public Key A. Den Private Key α hält sie geheim.

Schnorr-Signaturen – Signieren und verfizieren

Signieren: Um zu signieren, wählt Alice zufällig eine Zahl $k \in \mathbb{Z}_\omega$ und berechnet

$$r := g^k,$$

Challenge $\rightarrow$ $$c := H(r, m) \bmod \omega,$$

$$s := k + c\alpha \pmod{\omega}.$$

c

s

$H(m) + \alpha \cdot r$

Die Signatur ist dann das Paar (c, s).

Verifizieren: Will Bob die Signatur überprüfen, so führt er die folgenden Schritte durch:

1. Er berechnet $r := g^s \circ A^{-c}$ und
2. prüft, ob $c = H(r, m) \bmod \omega$.

$r = g^k$

$= g^{s-c\alpha} = g^s \circ ((g^{-1})^\alpha)^c$

$(A^{-1})^c$

$k = s - c\alpha$

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

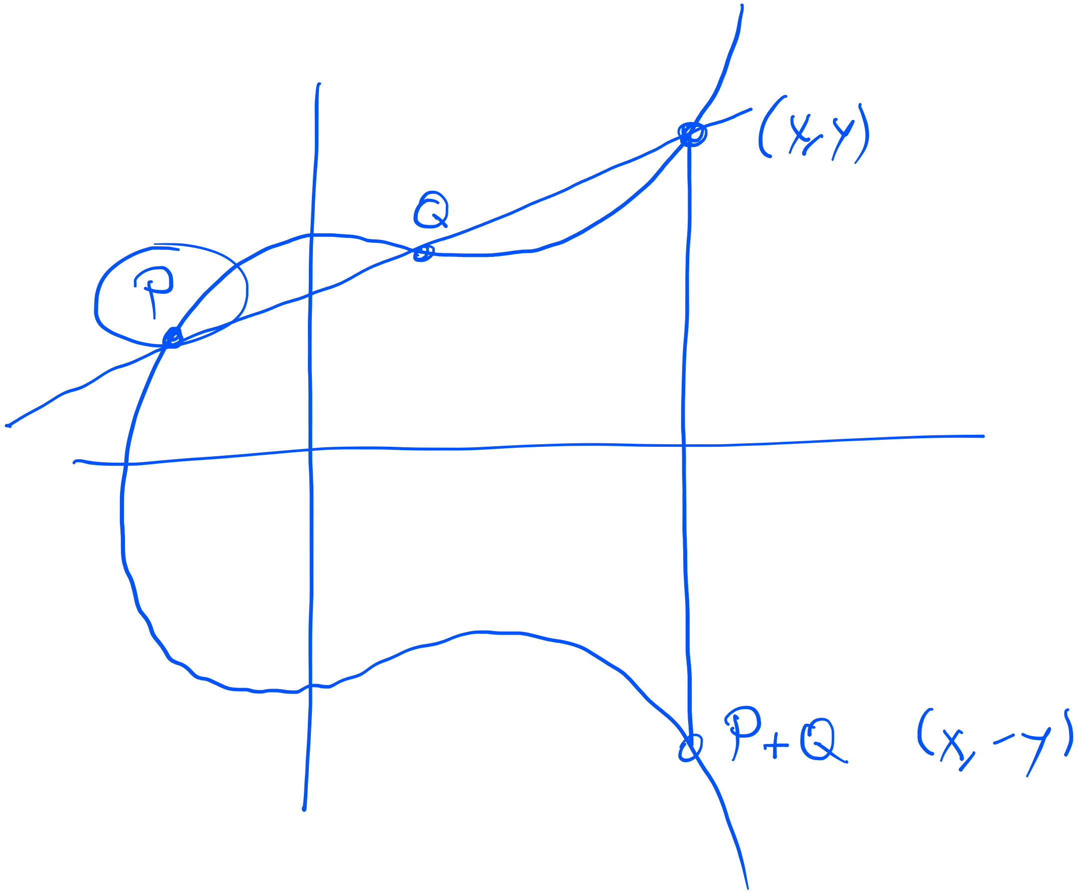
Schnorr-Signaturen – Korrektheit

Wir bemerken, dass die Überprüfung für eine korrekt erstellte Signatur erfolgreich ist. Es ergibt sich in Schritt 1. das richtige r, denn

$$
\begin{aligned}
g^s \circ A^{-c} &= g^{k+c\alpha} \circ g^{-c\alpha} \\
&= g^{k+c\alpha-c\alpha} \\
&= g^k \\
&= r.
\end{aligned}
$$

$g^7 = g^1 \circ g^2 \circ g^4$

$g^7 = g^8 \cdot g^{-1}$


P
Q
(x,y)
P+Q
(x,-y)