Diffie - Hellmann Schlüsselaustausch

Alice und Bob vereinbaren Primzahl p und eine Generatorzahl $g \in \{1, 2, \dots, p-1\}$, am besten eine Primitivwurzel.
Alice wählt geheim eine Zahl a aus, Bob geheim eine Zahl b, $a, b \in \{1, \dots, p-1\}$.
Alice berechnet $A = g^a \bmod p$. Bob berechnet $B = g^b \bmod p$.
Dann tauschen beide A und B aus. Das bedeutet: p, g, A, B sind öffentlich bekannt, a kennt nur Alice, b nur Bob.

Beide können nun den gemeinsamen Schlüssel K berechnen:

Alice: $B^a \equiv (g^b)^a \equiv g^{ab} \equiv K \bmod p$

Bob: $A^b \equiv (g^a)^b \equiv g^{ab} \equiv K \bmod p$

Beispiel: $p = 13$, $g = 2$

Alice: $a = 5$, $A \equiv 2^5 \equiv 6 \bmod 13$

Bob: $b = 8$ $B = 2^8 = 9 \bmod 13$

Schlüssel K berechnen:

Alice: $9^5 \equiv 3 \bmod 13 \Rightarrow K = 3$

Bob: $6^8 \equiv 3 \bmod 13 \Rightarrow K = 3$

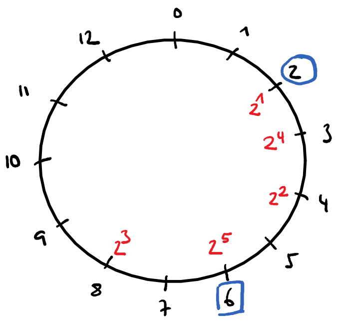
NR (mod 13)

$2^1 \equiv 2$
$2^2 \equiv 4$
$2^4 \equiv 3$
$9^1 \equiv -4$
$9^2 \equiv 16 \equiv 3$
$9^4 \equiv -4$

$6^1 \equiv 6$
$6^2 \equiv 36 \equiv -3$
$6^4 \equiv 9 \equiv -4$
$6^8 \equiv 16 \equiv 3$

Angriff auf Diffie-Hellmann

Eve versucht aus den öffentlich bekannten Zahlen den Schlüssel K zu berechnen. Eve weiß: $2^a \equiv 6 \bmod 13$ und $2^b \equiv 9 \bmod 13$.



Die Beschaffung der Exponenten a bzw. b heißt "Berechnung des diskreten Logarithmus". Bei kleinen Zahlen ist dies durch Ausprobieren möglich. g sollte Primitivwurzel sein, damit Angreifer möglichst viel ausprobieren muss.

In der Praxis ist p eine Primzahl mit ca. 300 Stellen $\approx 10^{300}$ (mehr als Atome im Weltall)
Es ist kein effektives Verfahren zur Berechnung des diskreten Logarithmus bekannt.

Unterschied zum "normalen" Logarithmus

Beim Rechnen in $\mathbb{R}$ kann man den Wert des Exponenten abschätzen.
Beispiel: Gesucht ist $a \in \mathbb{R}$ mit $2^a = 12$. Wegen $2^3 = 8$ und $2^4 = 16$ folgt: $3 < a < 4$.
Diese Folgerung ist im diskreten Fall nicht möglich. Die Potenzen der Generatorzahl springen wie zufällig in dem Restklassenring herum.

Man in the middle Angriff auf Diffie Hellman

Alice		Bob	
a	$\underbrace{g^{ab}}$	b	} gewünschtes setup
↑ geheim	gemeinsames Geheimnis	↑ geheim	

Alice		Mallory		Bob	
a	g^{am}	m	g^{bm}	b	} man-in-the-middle setup

Mallory kontrolliert das Netzwerk. Er gibt sich gegenüber Alice als Bob aus und gegenüber Bob als Alice. Mit beiden vereinbart er getrennte Schlüssel g^{am} und g^{bm}.

RSA-Verfahren

Alice wählt zwei Primzahlen p und q und berechnet $m = p \cdot q$ und $\tilde{m} = (p-1)\cdot(q-1)$.
Alice wählt Verschlüsselungsexponent e mit $1 < e < \tilde{m}$ und $ggT(e, \tilde{m}) = 1$
und berechnet Entschlüsselungsexponent d mit $\bar{d} = \frac{\bar{1}}{\bar{e}}$ in $\mathbb{Z}_{\tilde{m}}$
öffentlicher Schlüssel: (m, e) – privater Schlüssel: (m, d)
Für die zu verschlüsselnde Nachricht n muss gelten: $0 < n < m$

Bob verschlüsselt n: $N = n^e \bmod m$
Alice entschlüsselt N: $n = N^d \bmod m$

Beispiel:
$p = 7$, $q = 13$ $\Rightarrow$ $m = 91$ (RSA-Modul), $\tilde{m} = 6 \cdot 12 = 72$, $e = 11$, $ggT(e, \tilde{m}) = 1$.
Berechnung von d: $\bar{d} = \frac{\bar{1}}{\bar{e}}$ in $\mathbb{Z}_{\tilde{m}}$, d.h. $d \cdot e \equiv 1 \bmod \tilde{m}$, also $d \cdot e - 1 = k \cdot m$
Wir lösen die diophantische Gleichung $d \cdot 11 - y \cdot 72 = 1$

a	b	q	r	d	y
11	72	0	11	-13	2
72	11	6	6	2	-13
11	6	1	5	-1	2
6	5	1	1	1	-1
5	1	5	0	0	1

$d \equiv -13 \equiv 59 \bmod 72$

öffentlicher Schlüssel (91, 11)
privater Schlüssel (91, 59)

Bob verschlüsselt $n = 10$:

$N = 10^{11} \bmod 91 = 82$

NR:
$10^1 \equiv 10 \bmod 91$
$10^2 \equiv 9$
$10^4 \equiv -10$
$10^8 \equiv 100 \equiv 9$
$10^{11} \equiv 9 \cdot 9 \cdot 10 \equiv -100 \equiv 82$

Alice entschlüsselt $N = 82$

$n = 82^{59} \bmod 91 = 10$

$$82^1 \equiv -9 \bmod 91$$
$$82^2 \equiv 81 \equiv -10$$
$$82^4 \equiv 9$$
$$82^8 \equiv -10$$
$$82^{16} \equiv 9$$
$$82^{32} \equiv -10$$
$$82^{59} = 82^{32+16+8+2+1} \equiv \underbrace{-10\cdot 9}_{1} \cdot (-10) \cdot \underbrace{(-10)\cdot(-9)}_{-1} \equiv 10$$

Beweis, dass Entschlüsselung funktioniert:

Wir zeigen $n = N^d \bmod m$, indem wir zeigen: $(n^e)^d \equiv n \bmod m$

Wir zeigen zunächst: $(n^e)^d \equiv n \bmod p$ (1)

und $(n^e)^d \equiv n \bmod q$ (2)

$\bar{d} = \frac{1}{\bar{e}}$ in $\mathbb{Z}_{\tilde{m}} \Rightarrow d\cdot e \equiv 1 \bmod \tilde{m} \Rightarrow de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$ (3)

Beweis (1):

Fall 1: $p \mid n \Rightarrow n \equiv 0 \bmod p \Rightarrow (n^e)^d \equiv 0 \bmod p$ ✓

Fall 2: $p \nmid n \Rightarrow (n^e)^d = n^{e\cdot d} \overset{(3)}{=} n^{1+k(p-1)(q-1)} = n \cdot \underbrace{(n^{p-1})}_{\equiv 1}{}^{k(q-1)} \equiv n \bmod p$

Beweis (2) analog.

$(1),(2) \Rightarrow (n^e)^d - n = \underbrace{k_1 \cdot p = k_2 \cdot q}$ $= k_3 \cdot p \cdot q \Rightarrow (n^e)^d \equiv n \bmod p\cdot q = m$

p, q Primzahlen $\Rightarrow$
in k_1 steckt q und
in k_2 steckt p

q.e.d.

Angriff auf das RSA-Verfahren

Die Sicherheit des Verfahrens hängt davon ab, dass der Angreifer das öffentlich bekannte m nicht in die beiden Primfaktoren p und q zerlegen kann. Sonst könnte er $\tilde{m}$ berechnen und dann auch das Inverse zu e in $\mathbb{Z}_{\tilde{m}}$.

Aufwand für die Faktorensuche

Primzahl hat ca. 300 Stellen, das sind ca 10^{300} Kandidaten zum Testen.

Wir nehmen nur Primzahlen zum Testen, die kleiner sind als $\sqrt{10^{300}} = 10^{150}$

Abschätzung von Euler: Für große n gibt es ca. $\frac{n}{\ln(n)}$ Primzahlen unterhalb von n.

$\ln(10^{150}) = 150 \cdot \ln(10) \approx 150 \cdot 2{,}3 = 345{,}4 \Rightarrow \frac{10^{150}}{345{,}4} \approx 3 \cdot 10^{147}$ Kandidaten

Annahme: 1 Computer schafft 10^{12} Prüfungen pro Sekunde (10^{12} = 1 Million Millionen)

Im Dauerbetrieb pro Jahr: $10^{12} \cdot 60 \cdot 60 \cdot 24 \cdot 365 \simeq 3 \cdot 10^{19}$ Prüfungen.

$\frac{3\cdot 10^{147}}{3 \cdot 10^{19}} = 10^{147-19} = 10^{128}$ Jahre. (Alter des Weltalls: ca 10^{10} Jahre)

Wenn jeder Mensch einen Computer beisteuern würde und manche 2 kämen wir auf 10 Milliarden $= 10^{10}$ Computer. Wenn jeder dieser Computer 1000 mal schneller wäre, würde es immer noch $10^{128-10-3} = 10^{115}$ Jahre dauern.

Kryptographische Hashfunktionen

bilden Eingabewerte (z.B. einen Text oder eine Datei) auf einen Wert <u>fester Länge</u> ab, den Hash-Wert der Eingabe.

Beispiel: SHA-256 bildet Eingaben auf eine Bitfolge der Länge 256 ab.

Kryptografische Hashfunktionen sind kollisionsresistente Einwegfunktionen.

kollisionsresistent: es ist praktisch unmöglich, zwei Eingaben zu finden, die denselben Hash ergeben.

Einwegfunktion: es ist praktisch unmöglich, aus dem Hashwert den Eingabewert zu rekonstruieren.

Digitale Signatur einer Nachricht N

Alice m_A, e_A, d_A

N

$\downarrow$ berechnet

$Sig = hash(N)^{d_A} \bmod m_A$

Schickt an Bob $\xrightarrow{\quad}$ N, Sig

Bob $(hash(N))^{e_A} \bmod m_A \stackrel{?}{=} Sig$ ← Bob prüft

Falls Prüfung ok, ist Bob sicher, dass N von Alice stammt.

Schutz vor Man in the middle Angriff beim Diffie-Hellman Schlüsselaustausch durch Verwendung einer RSA-Signatur.

Alice m_A, e_A, d_A $\qquad$ p, g $\qquad$ Bob m_B, e_B, d_B

a $\qquad$ b

$\downarrow$ berechnet $\qquad$ berechnet $\downarrow$

$A = g^a \bmod p$ $\qquad$ $B = g^b \bmod p$

$Sig(A) = hash(A)^{d_A} \bmod m_A$ $\qquad$ $Sig(B) = hash(B)^{d_B} \bmod m_B$

$\Longleftrightarrow$

Austausch von A, B, Sig(A), Sig(B)

prüfen

$hash(B) \stackrel{?}{=} Sig(B)^{e_B} \bmod m_B$ $\qquad$ $hash(A) \stackrel{?}{=} Sig(A)^{e_A} \bmod m_A$

$$\underbrace{A^b \equiv B^a \equiv g^{ab} \bmod p}_{\text{gemeinsames Geheimnis, wenn Prüfung ok.}}$$