Kryptographie

Diffie-Hellman Schlüsselaustausch

Alice und Bob vereinbaren Primzahl p und $g \in \{1, \dots, p-1\}$ (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: $A \equiv g^a \bmod p$ Bob berechnet: $B \equiv g^b \bmod p$

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

Spalte hoch Zeile

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12
2	1	4	9	3	12	10	10	12	3	9	4	1
3	1	8	1	12	8	8	5	5	1	12	5	12
4	1	3	3	9	1	9	9	1	9	3	3	1
5	1	6	9	10	5	2	11	8	3	4	7	12
6	1	12	1	1	12	12	12	12	1	1	12	1
7	1	11	3	4	8	7	6	5	9	10	2	12
8	1	9	9	3	1	3	3	1	3	9	9	1
9	1	5	1	12	5	5	8	8	1	12	8	12
10	1	10	3	9	12	4	4	12	9	3	10	1
11	1	7	9	10	8	11	2	5	3	4	6	12
12	1	1	1	1	1	1	1	1	1	1	1	1

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

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

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

Schlüssel K berechnen:

Alice: $9^5 \equiv 3 \bmod 13$

Bob: $6^8 \equiv 3 \bmod 13$

$K = 3$

Eve versucht, aus den öffentlich bekannten Zahlen den Schlüssel zu berechnen:

Eve weiß: $2^a \equiv 6 \bmod 13, \qquad 2^b \equiv 9 \bmod 13$

Die Beschaffung des Exponenten a bzw. b heißt "Berechnung des diskreten Logarithmus".

Bei kleinen Zahlen durch Ausprobieren möglich. g sollte Primitivwurzel sein, damit Angreifer möglichst viel Ausprobieren muss.

In Wirklichkeit ist p eine Primzahl mit ca. 300 Stellen, z.B:

p = 27122499884782007313573862315617163760695312050615951156638011646208718232192774685831595242708640530635880215073339334618350781145154852252368889552227971191686849419199254815002403337429346070501642434032857100452019790443649514202341625495680324707557198157187765461211337288532274864389859236691151 $\sim 10^{300}$ (mehr als Atome im Weltall)

Ausprobieren dauert länger als das Weltall alt ist, ein effektives Verfahren zur Berechnung des diskreten Logarithmus ist nicht bekannt.

RSA-Verfahren

Alice wählt zwei Primzahlen p und q und berechnet $m = p \cdot q$ und $\tilde{m} = (p-1)(q-1)$.

Alice wählt Verschlüsselungsexponent e mit $1 < e < \tilde{m}$ und $ggT(e, \tilde{m}) = 1$.

und berechnet den Entschlüsselungsexponent d: $e \cdot d \equiv 1 \bmod \tilde{m}$

e = encryption
d = decryption

öffentlicher Schlüssel: (m, e) privater Schlüssel: (m, d)

Bob verschlüsselt die Nachricht n: $N = n^e \bmod m$

Alice entschlüsselt die Nachricht N: $n = N^d \bmod m$

Beispiel: $p = 7, \; q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$

$e = 11, \quad ggT(72, 11) = 1$ ✓

$d \cdot 11 \equiv 1 \bmod 72 \Rightarrow d \cdot 11 - 1 = k \cdot 72$ k geeignet

$d \cdot 11 - k \cdot 72 = 1$ (diophantische Gleichung)

a	b	q	r	x	y
72	11	6	6	2	$-1-12=-13$
11	6	1	5	-1	$1+1=2$
6	5	1	1	1	$0-1=-1$
5	1	5	0	0	1

$$\Rightarrow \quad d \equiv -13 \equiv 59 \mod 72$$

öffentlicher Schlüssel: $(91, 11)$

privater Schlüssel: $(91, 59)$

n = 10 soll verschlüsselt werden:

$$N = 10^{11} \mod 91$$

$$\underline{N = 82}$$

$11 = (1011)_2$

11	
5	1
2	1
1	0
0	1

$10^1 \equiv 10 \mod 91$

$10^2 \equiv 9$

$10^4 \equiv 81 = -10$

$10^8 \equiv 100 \equiv 9$

$$10^{11} = 10^8 \cdot 10^2 \cdot 10^1 \equiv 9 \cdot 9 \cdot 10 \equiv -10 \cdot 10 \equiv -9 \equiv \underline{82}$$

N = 82 soll entschlüsselt werden:

$$n = 82^{59} \mod 91$$

$$n = \underline{10}$$

$59 = (111011)_2$

59	
29	1
14	1
7	0
3	1
1	1
0	1

$82^1 \equiv -9 \mod 91$

$82^2 \equiv 81 \equiv -10$

$82^4 \equiv 100 \equiv 9$

$82^8 \equiv 81 \equiv -10$

$82^{16} \equiv 9$

$82^{32} \equiv -10$

$$81^{59} \equiv \underbrace{-10 \cdot 9}_{1} \cdot (-10) \cdot \underbrace{(-10)(-9)}_{-1} \equiv \underline{10} \mod 91$$

Beweis, dass Entschlüsselung funktioniert:

Zu zeigen: $N^d \equiv n \mod m$

Es gilt: $e \cdot d \equiv 1 \mod \tilde{m} \Rightarrow ed - 1 = k\tilde{m}$ mit einem $k \in \mathbb{Z}$ $(*)$

(1) Wir zeigen $N^d \equiv n \mod p$

Falls $p \mid n \Rightarrow n \equiv 0$ und $p \Rightarrow N^d \equiv (n^e)^d \equiv n^{ed} \equiv 0^{ed} \equiv 0 \equiv n \mod p$

Falls $p \nmid n \underset{\text{kl. Satz von Fermat}}{\Rightarrow} n^{p-1} \equiv 1 \mod p \Rightarrow N^d \equiv n^{ed} \underset{(*)}{=} n^{1+k\tilde{m}} = n \cdot n^{k(p-1)(q-1)} \equiv n \cdot 1^{k(q-1)} = n \mod p$

(2) Genauso zeigen wir: $N^d \equiv n \mod q$

$$\left.\begin{aligned}(1) &\Rightarrow N^d - n = k \cdot p \\ (2) &\Rightarrow N^d - n = k' \cdot q\end{aligned}\right\} \quad kp = k'q \Rightarrow q \mid kp \underset{p,q \text{ Prim}}{\Rightarrow} q \mid k \Rightarrow k = k''q$$

$$N^d - n = k \cdot p = k'' p \cdot q \Rightarrow N^d \equiv n \mod p \cdot q = m \quad \checkmark$$

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 $\mathbb{Z}_{\tilde{m}}$ Schlange.

Überlegung zum Aufwand der Faktorensuche

Primzahl hat ca. 300 Stellen: Es bleiben ca. 10^{300} Kandidaten zum Testen

Gerade Zahlen muss man nicht testen: $5 \cdot 10^{299}$

Wir nehmen nur Primzahlen als Testkandidaten. Nach der Abschätzung von Euler gibt es unterhalb einer großen Zahl x ca. x/ln(x) viele Primzahlen

$$\ln(10^{300}) = 300 \cdot \ln(10) \approx 300 \cdot 2{,}3 \approx 691 . \qquad \frac{10^{300}}{\ln(10^{300})} \approx 1.4 \cdot 10^{297}$$ Kandidaten zum Testen

Man braucht nicht alle Primzahlen unterhalb von 10^{300}, sondern nur die unterhalb von $\sqrt{10^{300}} = 10^{150}$.

10^{8} 10^{6}

Davon gibt es ca. $\dfrac{10^{150}}{150 \cdot \ln(10)} \approx 3 \cdot 10^{147}$ viele

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

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

^{147} $^{147-19}$ ^{128} ^{10}

Annahme: 1 Computer schafft 10 (1 Million Millionen = 1 Billion) Prüfungen pro Sekunde.

Im Dauerbetrieb pro Jahr: $10^{12} \cdot 60 \cdot 60 \cdot 24 \cdot 365 \approx 3 \cdot 10^{19}$ 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 eine Computer beisteuern würde und manche vielleicht 2 kämen wir auf 10 Milliarden Computer $10 \cdot 10^9 = 10^{10}$

Wenn jeder dieser Computer 1000 mal schneller wäre, würde es immer noch $10^{128-13} = 10^{115}$ Jahre dauern.

Digitale Signatur

Alice hat den öffentlichen Schlüssel (m, e) und privaten Schlüssel (m, d)

Alice schreibt unverschlüsselt die Nachricht N und signiert sie mit $sig \equiv N^d \bmod m$

Bob erhält die Nachricht und Signatur und berechnet $sig^e \bmod m$

Wenn $sig^e \equiv N \bmod m$ ist sich Bob sicher, dass die Nachricht wirklich von Alice stammt.

In der Praxis wird nicht die Nachricht, sondern der Fingerabdruck der Nachricht, der mit einer kryptographischen Hashfunktion erstellt wird, signiert.

Zertifizierung des öffentlichen Schlüssels

Um die Authentizität ihres öffentlichen Schlüssels zu sichern, lässt Alice den Schlüssel von einem anerkannten Vertrauensbüro (Trust-Center) zertifizieren.

1. Alice geht mit Personalausweis und (m, e) zum Trust-Center T.
2. T bildet aus dem Namen von Alice und (m, e) eine id: id_{Alice} .
3. T hat ein eigenes RSA-Schlüsselpaar (m_T ,e_T) (öffentlich) und (m_T, d_T) privat.
4. T berechnet $z_{Alice} = (id_{Alice})^{d_T} \bmod m_T$ und schickt z_{Alice} an Alice.
5. Alice veröffentlicht außer (m,e) auch z_{Alice} und einen Link zu T.
6. Bob möchte feststellen, ob (m,e) wirklich zu Alice gehört.
7. Bob geht zu T und holt sich den öffentlichen Schlüssel (m_T ,e_T).
8. Bob berechnet test = $(z_{Alice})^{e_T} \bmod m_T$.
9. Wenn Bob in test Alice Namen und ihren öffentliche Schlüssel liest, hält er diesen Schlüssel für authentisch.