Vertiefungskurs Mathematik

Kryptographie

Kryptographie ist die Wissenschaft der Verschlüsselung von Informationen.

Kryptographie ist die Wissenschaft der Verschlüsselung von Informationen.

Diffie-Hellman Schlüsselaustausch

Kryptographie ist die Wissenschaft der Verschlüsselung von Informationen.

Diffie-Hellman Schlüsselaustausch

Alice und Bob vereinbaren eine Primzahl p und eine Generatorzahl $g \in \{1, 2, ...p-1\}$, am besten eine Primitivwurzel in $\mathbb{Z}_p$.

Kryptographie ist die Wissenschaft der Verschlüsselung von Informationen.

Diffie-Hellman Schlüsselaustausch

Alice und Bob vereinbaren eine Primzahl p und eine Generatorzahl $g \in \{1, 2, ...p-1\}$, am besten eine Primitivwurzel in $\mathbb{Z}_p$. Alice wählt geheim eine Zahl a aus, Bob geheim eine Zahl b mit $a, b \in \{1, 2, ...p-1\}$.

Kryptographie ist die Wissenschaft der Verschlüsselung von Informationen.

Diffie-Hellman Schlüsselaustausch

Alice und Bob vereinbaren eine Primzahl p und eine Generatorzahl $g \in \{1, 2, ...p - 1\}$, am besten eine Primitivwurzel in $\mathbb{Z}_p$. Alice wählt geheim eine Zahl a aus, Bob geheim eine Zahl b mit $a, b \in \{1, 2, ...p - 1\}$. Alice berechnet $A = g^a$ mod p, $B = g^b$ mod p.

Kryptographie ist die Wissenschaft der Verschlüsselung von Informationen.

Diffie-Hellman Schlüsselaustausch

Alice und Bob vereinbaren eine Primzahl p und eine Generatorzahl $g \in \{1, 2, ...p - 1\}$, am besten eine Primitivwurzel in $\mathbb{Z}_p$. Alice wählt geheim eine Zahl a aus, Bob geheim eine Zahl b mit $a, b \in \{1, 2, ...p - 1\}$. Alice berechnet $A = g^a \bmod p$, $B = g^b \bmod p$. Dann tauschen beide A und B aus.

Kryptographie ist die Wissenschaft der Verschlüsselung von Informationen.

Diffie-Hellman Schlüsselaustausch

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

Kryptographie ist die Wissenschaft der Verschlüsselung von Informationen.

Diffie-Hellman Schlüsselaustausch

Alice und Bob vereinbaren eine Primzahl p und eine Generatorzahl $g \in \{1, 2, ...p-1\}$, am besten eine Primitivwurzel in $\mathbb{Z}_p$. Alice wählt geheim eine Zahl a aus, Bob geheim eine Zahl b mit $a, b \in \{1, 2, ...p-1\}$. Alice berechnet $A = g^a \bmod p$, $B = g^b \bmod p$. 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: $B^a \equiv (g^b)^a \equiv g^{ba} \equiv K \bmod p$
Bob: $A^b \equiv (g^a)^b \equiv g^{ab} \equiv K \bmod p$

Beispiel: $p = 13, g = 2$

Beispiel: $p = 13, g = 2$
Alice: $a = 5,$

Beispiel: $p = 13, g = 2$
Alice: $a = 5, \quad A \equiv 2^5 \equiv 6 \mod 13$
Bob: $b = 8,$

Beispiel: $p = 13, g = 2$
Alice: $a = 5, \quad A \equiv 2^5 \equiv 6 \bmod 13$
Bob: $b = 8, \quad B \equiv 2^8 \equiv 9 \bmod 13$

Beispiel: $p = 13, g = 2$
Alice: $a = 5, \quad A \equiv 2^5 \equiv 6 \mod 13$
Bob: $b = 8, \quad B \equiv 2^8 \equiv 9 \mod 13$

Schlüssel berechnen:

Beispiel: $p = 13, g = 2$
Alice: $a = 5, \quad A \equiv 2^5 \equiv 6 \bmod 13$
Bob: $b = 8, \quad B \equiv 2^8 \equiv 9 \bmod 13$

Schlüssel berechnen:
Alice: $B^a \equiv 9^5 \equiv 3 \bmod 13 \Rightarrow K = 3$

Beispiel: $p = 13, g = 2$
Alice: $a = 5, \quad A \equiv 2^5 \equiv 6 \bmod 13$
Bob: $b = 8, \quad B \equiv 2^8 \equiv 9 \bmod 13$

Schlüssel berechnen:
Alice: $B^a \equiv 9^5 \equiv 3 \bmod 13 \Rightarrow K = 3$
Bob: $A^b \equiv 6^8 \equiv 3 \bmod 13 \Rightarrow K = 3$

Beispiel: $p = 13, g = 2$
Alice: $a = 5, \quad A \equiv 2^5 \equiv 6 \mod 13$
Bob: $b = 8, \quad B \equiv 2^8 \equiv 9 \mod 13$

Schlüssel berechnen:
Alice: $B^a \equiv 9^5 \equiv 3 \mod 13 \Rightarrow K = 3$
Bob: $A^b \equiv 6^8 \equiv 3 \mod 13 \Rightarrow K = 3$

Nebenrechnungen (mod 13):

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

Angriff auf Diffie-Hellman

Angriff auf Diffie-Hellman

Eve versucht aus den öffentlich bekannten Zahlen p, g, A, B den Schlüssel K zu berechnen.

Angriff auf Diffie-Hellman

Eve versucht aus den öffentlich bekannten Zahlen p, g, A, B den Schlüssel K zu berechnen. Eve weiß: $2^a \equiv 6 \mod 13$ und $2^b \equiv 9 \mod 13$.

Angriff auf Diffie-Hellman

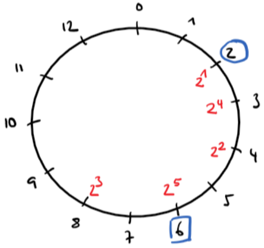
Eve versucht aus den öffentlich bekannten Zahlen p, g, A, B den Schlüssel K zu berechnen. Eve weiß: $2^a \equiv 6 \mod 13$ und $2^b \equiv 9 \mod 13$.

Die Beschaffung des Exponenten a oder b heißt ‘Berechnung des diskreten Logarithmus’.

Angriff auf Diffie-Hellman

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

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



Unterschied Logarithmus in $\mathbb{Z}_p$ und Logarithmus in $\mathbb{R}$

Unterschied Logarithmus in $\mathbb{Z}_p$ und Logarithmus in $\mathbb{R}$

Beim Rechnen in $\mathbb{R}$ kann man den Wert des Exponenten abschätzen.

Unterschied Logarithmus in $\mathbb{Z}_p$ und Logarithmus in $\mathbb{R}$

Beim Rechnen in $\mathbb{R}$ kann man den Wert des Exponenten abschätzen.

Beispiel: Gesucht ist $a \in \mathbb{R}$ mit $2^a = 12$.

Unterschied Logarithmus in $\mathbb{Z}_p$ und Logarithmus in $\mathbb{R}$

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: $3 < a < 4$.

Unterschied Logarithmus in $\mathbb{Z}_p$ und Logarithmus in $\mathbb{R}$

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.

Unterschied Logarithmus in $\mathbb{Z}_p$ und Logarithmus in $\mathbb{R}$

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

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

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

RSA-Verfahren

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

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 $\mathrm{ggT}(e, \tilde{m}) = 1$

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 $\mathrm{ggT}(e, \tilde{m}) = 1$ und berechnet Entschlüsselungsexponent d mit $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$.

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 $\mathrm{ggT}(e, \tilde{m}) = 1$ und berechnet Entschlüsselungsexponent d mit $\overline{d} = \dfrac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$.

Dann ist der öffentliche Schlüssel (m, e) und der private Schlüssel (m, d).

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 $\mathrm{ggT}(e, \tilde{m}) = 1$ und berechnet Entschlüsselungsexponent d mit $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$.

Dann ist der öffentliche Schlüssel (m, e) und der private Schlüssel (m, d).

Für die zu verschlüsselnde Nachricht muss gelten: $0 < n < m$.

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 $\mathrm{ggT}(e, \tilde{m}) = 1$ und berechnet Entschlüsselungsexponent d mit $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$.

Dann ist der öffentliche Schlüssel (m, e) und der private Schlüssel (m, d).

Für die zu verschlüsselnde Nachricht muss gelten: $0 < n < m$.

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

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 $\mathrm{ggT}(e, \tilde{m}) = 1$ und berechnet Entschlüsselungsexponent d mit $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$.

Dann ist der öffentliche Schlüssel (m, e) und der private Schlüssel (m, d).

Für die zu verschlüsselnde Nachricht muss gelten: $0 < n < m$.

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

Beispiel: $p = 7, q = 13$

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

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$,

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\mathrm{ggT}(e, \tilde{m}) = 1$.

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\mathrm{ggT}(e, \tilde{m}) = 1$.
Berechnung von d: $\overline{d} = \dfrac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\text{ggT}(e, \tilde{m}) = 1$.
Berechnung von d: $\overline{d} = \dfrac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, d.h. $d \cdot e \equiv 1 \mod \tilde{m}$

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\mathrm{ggT}(e, \tilde{m}) = 1$.

Berechnung von d: $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, d.h. $d \cdot e \equiv 1 \mod \tilde{m}$, also
$d \cdot e - 1 = k \cdot \tilde{m}$ für ein $k \in \mathbb{Z}$.

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\mathrm{ggT}(e, \tilde{m}) = 1$.

Berechnung von d: $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, d.h. $d \cdot e \equiv 1 \mod \tilde{m}$, also
$d \cdot e - 1 = k \cdot \tilde{m}$ für ein $k \in \mathbb{Z}$.

Wir lösen die diophantische Gleichung $d \cdot 11 - k \cdot 72 = 1$

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\mathrm{ggT}(e, \tilde{m}) = 1$.
Berechnung von d: $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, d.h. $d \cdot e \equiv 1 \mod \tilde{m}$, also
$d \cdot e - 1 = k \cdot \tilde{m}$ für ein $k \in \mathbb{Z}$.
Wir lösen die diophantische Gleichung $d \cdot 11 - k \cdot 72 = 1$ mit dem Erweiterten Euklidschen Algorithmus und erhalten $d = 59$.

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\mathrm{ggT}(e, \tilde{m}) = 1$.
Berechnung von d: $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, d.h. $d \cdot e \equiv 1 \mod \tilde{m}$, also
$d \cdot e - 1 = k \cdot \tilde{m}$ für ein $k \in \mathbb{Z}$.
Wir lösen die diophantische Gleichung $d \cdot 11 - k \cdot 72 = 1$ mit dem Erweiterten Euklidschen Algorithmus und erhalten $d = 59$. Der öffentliche Schlüssel ist $(91, 11)$, der private Schlüssel ist $(91, 59)$.

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\mathrm{ggT}(e, \tilde{m}) = 1$.

Berechnung von d: $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, d.h. $d \cdot e \equiv 1 \bmod \tilde{m}$, also
$d \cdot e - 1 = k \cdot \tilde{m}$ für ein $k \in \mathbb{Z}$.
Wir lösen die diophantische Gleichung $d \cdot 11 - k \cdot 72 = 1$ mit dem Erweiterten Euklidschen Algorithmus und erhalten $d = 59$. Der öffentliche Schlüssel ist $(91, 11)$, der private Schlüssel ist $(91, 59)$.

Bob verschlüsselt $n = 10$:

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\mathrm{ggT}(e, \tilde{m}) = 1$.
Berechnung von d: $\overline{d} = \dfrac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, d.h. $d \cdot e \equiv 1 \bmod \tilde{m}$, also $d \cdot e - 1 = k \cdot \tilde{m}$ für ein $k \in \mathbb{Z}$.
Wir lösen die diophantische Gleichung $d \cdot 11 - k \cdot 72 = 1$ mit dem Erweiterten Euklidschen Algorithmus und erhalten $d = 59$. Der öffentliche Schlüssel ist $(91, 11)$, der private Schlüssel ist $(91, 59)$.

Bob verschlüsselt $n = 10$: $\quad N = 10^{11} \bmod 91 = 82$

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\mathrm{ggT}(e, \tilde{m}) = 1$.

Berechnung von d: $\overline{d} = \dfrac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, d.h. $d \cdot e \equiv 1 \bmod \tilde{m}$, also
$d \cdot e - 1 = k \cdot \tilde{m}$ für ein $k \in \mathbb{Z}$.
Wir lösen die diophantische Gleichung $d \cdot 11 - k \cdot 72 = 1$ mit dem Erweiterten Euklidschen Algorithmus und erhalten $d = 59$. Der öffentliche Schlüssel ist $(91, 11)$, der private Schlüssel ist $(91, 59)$.

Bob verschlüsselt $n = 10$: $\quad N = 10^{11} \bmod 91 = 82$
Alice entschlüsselt $N = 82$:

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\mathrm{ggT}(e, \tilde{m}) = 1$.

Berechnung von d: $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, d.h. $d \cdot e \equiv 1 \mod \tilde{m}$, also
$d \cdot e - 1 = k \cdot \tilde{m}$ für ein $k \in \mathbb{Z}$.
Wir lösen die diophantische Gleichung $d \cdot 11 - k \cdot 72 = 1$ mit dem Erweiterten Euklidschen Algorithmus und erhalten $d = 59$. Der öffentliche Schlüssel ist $(91, 11)$, der private Schlüssel ist $(91, 59)$.

Bob verschlüsselt $n = 10$: $\quad N = 10^{11} \mod 91 = 82$
Alice entschlüsselt $N = 82$: $\quad n = 82^{59} \mod 91 = 10$

Beispiel: $p = 7, q = 13 \Rightarrow m = 91$ (RSA-Modul), $\tilde{m} = 72$
$e = 11$, $\mathrm{ggT}(e, \tilde{m}) = 1$.

Berechnung von d: $\overline{d} = \dfrac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, d.h. $d \cdot e \equiv 1 \bmod \tilde{m}$, also
$d \cdot e - 1 = k \cdot \tilde{m}$ für ein $k \in \mathbb{Z}$.

Wir lösen die diophantische Gleichung $d \cdot 11 - k \cdot 72 = 1$ mit dem Erweiterten Euklidschen Algorithmus und erhalten $d = 59$. Der öffentliche Schlüssel ist $(91, 11)$, der private Schlüssel ist $(91, 59)$.

Bob verschlüsselt $n = 10$: $\quad N = 10^{11} \bmod 91 = 82$
Alice entschlüsselt $N = 82$: $\quad n = 82^{59} \bmod 91 = 10$

Nebenrechnungen (mod 91)	$82^1 \equiv -9$
$10^1 \equiv 10$	$82^2 \equiv 81 \equiv -10$
$10^2 \equiv 9$	$82^4 \equiv 9$
$10^4 \equiv -10$	$82^8 \equiv -10$
$10^8 \equiv 100 \equiv 9$	$82^{16} \equiv 9$
$10^{11} \equiv 9 \cdot 9 \cdot 10 \equiv -100 \equiv 82$	$82^{32} \equiv -10$
	$82^{59} \equiv 82^{32+16+8+2+1} \equiv -10 \cdot 9 \cdot -10 \cdot -10 \cdot -9 \equiv 10$

Beweis, dass RSA-Entschlüsselung funktioniert

Beweis, dass RSA-Entschlüsselung funktioniert

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

Beweis, dass RSA-Entschlüsselung funktioniert

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

$(1) : (n^e)^d \equiv n \bmod p \quad$ und $(2) : (n^e)^d \equiv n \bmod q \quad (2)$

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \bmod m$, indem wir zeigen: $(n^e)^d \equiv n \bmod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \bmod p \quad$ und $(2) : (n^e)^d \equiv n \bmod q \quad (2)$

Es ist $\overline{d} = \dfrac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \mod m$, indem wir zeigen: $(n^e)^d \equiv n \mod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \mod p \quad$ und $(2) : (n^e)^d \equiv n \mod q \quad (2)$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$.

Beweis, dass RSA-Entschlüsselung funktioniert

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

$$(1) : (n^e)^d \equiv n \bmod p \quad \text{und } (2) : (n^e)^d \equiv n \bmod q \quad (2)$$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt: $de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$.

Beweis, dass RSA-Entschlüsselung funktioniert

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

$$(1) : (n^e)^d \equiv n \bmod p \quad \text{und } (2) : (n^e)^d \equiv n \bmod q \qquad (2)$$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt: $de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):

Beweis, dass RSA-Entschlüsselung funktioniert

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

$$(1) : (n^e)^d \equiv n \bmod p \quad \text{und } (2) : (n^e)^d \equiv n \bmod q \quad (2)$$

Es ist $\overline{d} = \dfrac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n$

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \bmod m$, indem wir zeigen: $(n^e)^d \equiv n \bmod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \bmod p \quad$ und $(2) : (n^e)^d \equiv n \bmod q \quad (2)$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \bmod p$

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \bmod m$, indem wir zeigen: $(n^e)^d \equiv n \bmod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \bmod p \quad \text{und } (2) : (n^e)^d \equiv n \bmod q \quad (2)$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \bmod p \Rightarrow (n^e)^d \equiv 0 \bmod p$.

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \bmod m$, indem wir zeigen: $(n^e)^d \equiv n \bmod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \bmod p \quad$ und $(2) : (n^e)^d \equiv n \bmod q \quad (2)$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \bmod p \Rightarrow (n^e)^d \equiv 0 \bmod p$.
Fall 2: $p \nmid n$

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \mod m$, indem wir zeigen: $(n^e)^d \equiv n \mod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \mod p \quad$ und $(2) : (n^e)^d \equiv n \mod q \quad (2)$

Es ist $\overline{d} = \dfrac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \mod p \Rightarrow (n^e)^d \equiv 0 \mod p$.
Fall 2: $p \nmid n$
$\Rightarrow (n^e)^d = n^{ed} = n^{1+k(p-1)(q-1)}$

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \bmod m$, indem wir zeigen: $(n^e)^d \equiv n \bmod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \bmod p \quad$ und $(2) : (n^e)^d \equiv n \bmod q \quad (2)$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \bmod p \Rightarrow (n^e)^d \equiv 0 \bmod p$.
Fall 2: $p \nmid n$
$\Rightarrow (n^e)^d = n^{ed} = n^{1+k(p-1)(q-1)} = n \cdot \left(n^{p-1}\right)^{k(q-1)}$

Beweis, dass RSA-Entschlüsselung funktioniert

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

$(1) : (n^e)^d \equiv n \mod p \quad$ und $(2) : (n^e)^d \equiv n \mod q \quad (2)$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \mod p \Rightarrow (n^e)^d \equiv 0 \mod p$.
Fall 2: $p \nmid n$
$\Rightarrow (n^e)^d = n^{ed} = n^{1+k(p-1)(q-1)} = n \cdot (n^{p-1})^{k(q-1)} \equiv n \mod p$

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \bmod m$, indem wir zeigen: $(n^e)^d \equiv n \bmod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \bmod p \quad$ und $(2) : (n^e)^d \equiv n \bmod q \quad (2)$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \bmod p \Rightarrow (n^e)^d \equiv 0 \bmod p$.
Fall 2: $p \nmid n$
$\Rightarrow (n^e)^d = n^{ed} = n^{1+k(p-1)(q-1)} = n \cdot (n^{p-1})^{k(q-1)} \equiv n \bmod p$, da $n^{p-1} \equiv 1 \bmod p$ nach dem kleinen Satz von Fermat.

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \bmod m$, indem wir zeigen: $(n^e)^d \equiv n \bmod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \bmod p \quad$ und $(2) : (n^e)^d \equiv n \bmod q \quad (2)$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \bmod p \Rightarrow (n^e)^d \equiv 0 \bmod p$.
Fall 2: $p \nmid n$
$\Rightarrow (n^e)^d = n^{ed} = n^{1+k(p-1)(q-1)} = n \cdot (n^{p-1})^{k(q-1)} \equiv n \bmod p$, da $n^{p-1} \equiv 1 \bmod p$ nach dem kleinen Satz von Fermat.
Beweis von (2) analog.

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \mod m$, indem wir zeigen: $(n^e)^d \equiv n \mod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \mod p \quad$ und $(2) : (n^e)^d \equiv n \mod q \quad (2)$

Es ist $\overline{d} = \dfrac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \mod p \Rightarrow (n^e)^d \equiv 0 \mod p$.
Fall 2: $p \nmid n$
$\Rightarrow (n^e)^d = n^{ed} = n^{1+k(p-1)(q-1)} = n \cdot (n^{p-1})^{k(q-1)} \equiv n \mod p$, da $n^{p-1} \equiv 1 \mod p$ nach dem kleinen Satz von Fermat.
Beweis von (2) analog.
Aus (1) und (2) folgt: $(n^e)^d - n = k_1 \cdot p = k_2 \cdot q$. mit geeigneten $k_1, k_2 \in \mathbb{Z}$.

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \bmod m$, indem wir zeigen: $(n^e)^d \equiv n \bmod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \bmod p \quad$ und $(2) : (n^e)^d \equiv n \bmod q \quad (2)$

Es ist $\overline{d} = \dfrac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \bmod p \Rightarrow (n^e)^d \equiv 0 \bmod p$.
Fall 2: $p \nmid n$
$\Rightarrow (n^e)^d = n^{ed} = n^{1+k(p-1)(q-1)} = n \cdot (n^{p-1})^{k(q-1)} \equiv n \bmod p$, da $n^{p-1} \equiv 1 \bmod p$ nach dem kleinen Satz von Fermat.
Beweis von (2) analog.
Aus (1) und (2) folgt: $(n^e)^d - n = k_1 \cdot p = k_2 \cdot q$. mit geeigneten $k_1, k_2 \in \mathbb{Z}$. Da p, q Primzahlen, steckt k_1 in q und k_2 in p.

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \bmod m$, indem wir zeigen: $(n^e)^d \equiv n \bmod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \bmod p \quad$ und $(2) : (n^e)^d \equiv n \bmod q \quad (2)$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \bmod p \Rightarrow (n^e)^d \equiv 0 \bmod p$.
Fall 2: $p \nmid n$
$\Rightarrow (n^e)^d = n^{ed} = n^{1+k(p-1)(q-1)} = n \cdot (n^{p-1})^{k(q-1)} \equiv n \bmod p$, da $n^{p-1} \equiv 1 \bmod p$ nach dem kleinen Satz von Fermat.
Beweis von (2) analog.
Aus (1) und (2) folgt: $(n^e)^d - n = k_1 \cdot p = k_2 \cdot q$. mit geeigneten $k_1, k_2 \in \mathbb{Z}$. Da p, q Primzahlen, steckt k_1 in q und k_2 in p. Also gilt: $(n^e)^d - n = k_3 \cdot p \cdot q$.

Beweis, dass RSA-Entschlüsselung funktioniert

Wir zeigen $n = N^d \bmod m$, indem wir zeigen: $(n^e)^d \equiv n \bmod m$. Wir zeigen zunächst:
$(1) : (n^e)^d \equiv n \bmod p \quad$ und $(2) : (n^e)^d \equiv n \bmod q \quad (2)$

Es ist $\overline{d} = \frac{\overline{1}}{\overline{e}}$ in $\mathbb{Z}_{\tilde{m}}$, also gilt: $de - 1 = k\tilde{m}$. Daraus folgt:
$de = 1 + k\tilde{m} = 1 + k(p-1)(q-1)$. Beweis von (1):
Fall 1: $p|n \Rightarrow n \equiv 0 \bmod p \Rightarrow (n^e)^d \equiv 0 \bmod p$.
Fall 2: $p \nmid n$
$\Rightarrow (n^e)^d = n^{ed} = n^{1+k(p-1)(q-1)} = n \cdot (n^{p-1})^{k(q-1)} \equiv n \bmod p$, da $n^{p-1} \equiv 1 \bmod p$ nach dem kleinen Satz von Fermat.
Beweis von (2) analog.
Aus (1) und (2) folgt: $(n^e)^d - n = k_1 \cdot p = k_2 \cdot q$. mit geeigneten $k_1, k_2 \in \mathbb{Z}$. Da p, q Primzahlen, steckt k_1 in q und k_2 in p. Also gilt: $(n^e)^d - n = k_3 \cdot p \cdot q$. Daraus folgt $(n^e)^d \equiv n \bmod m$. $\square$

Angriff auf das RSA-Verfahren

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.

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 dem öffentlichen e in $\mathbb{Z}_{\tilde{m}}$.

Aufwand für die Faktorensuche

Aufwand für die Faktorensuche

Wenn wir versuchen, die Faktoren einer Primzahl mit 300 Stellen zu finden, testen wir nur Primzahlen, die kleiner als $\sqrt{10^{300}} = 10^{150}$ sind.

Aufwand für die Faktorensuche

Wenn wir versuchen, die Faktoren einer Primzahl mit 300 Stellen zu finden, testen wir nur Primzahlen, die kleiner als $\sqrt{10^{300}} = 10^{150}$ sind. Nach der Abschätzung von Euler gilt für große n, dass es ca. $\frac{n}{\ln(n)}$ Primzahlen unterhalb von n gibt.

Aufwand für die Faktorensuche

Wenn wir versuchen, die Faktoren einer Primzahl mit 300 Stellen zu finden, testen wir nur Primzahlen, die kleiner als $\sqrt{10^{300}} = 10^{150}$ sind.
Nach der Abschätzung von Euler gilt für große n, dass es ca. $\frac{n}{\ln(n)}$ Primzahlen unterhalb von n gibt.
$\ln(10^{150})$

Aufwand für die Faktorensuche

Wenn wir versuchen, die Faktoren einer Primzahl mit 300 Stellen zu finden, testen wir nur Primzahlen, die kleiner als $\sqrt{10^{300}} = 10^{150}$ sind.
Nach der Abschätzung von Euler gilt für große n, dass es ca. $\frac{n}{\ln(n)}$ Primzahlen unterhalb von n gibt.
$\ln(10^{150}) = 150 \cdot \ln(10) \approx 150 \cdot 2.3 = 345.4$.

Aufwand für die Faktorensuche

Wenn wir versuchen, die Faktoren einer Primzahl mit 300 Stellen zu finden, testen wir nur Primzahlen, die kleiner als $\sqrt{10^{300}} = 10^{150}$ sind. Nach der Abschätzung von Euler gilt für große n, dass es ca. $\frac{n}{\ln(n)}$ Primzahlen unterhalb von n gibt.
$\ln(10^{150}) = 150 \cdot \ln(10) \approx 150 \cdot 2.3 = 345.4$. Also müssen wir $\frac{10^{150}}{345.5} \approx 3 \cdot 10^{147}$ Kandidaten testen.

Aufwand für die Faktorensuche

Wenn wir versuchen, die Faktoren einer Primzahl mit 300 Stellen zu finden, testen wir nur Primzahlen, die kleiner als $\sqrt{10^{300}} = 10^{150}$ sind. Nach der Abschätzung von Euler gilt für große n, dass es ca. $\frac{n}{\ln(n)}$ Primzahlen unterhalb von n gibt.
$\ln(10^{150}) = 150 \cdot \ln(10) \approx 150 \cdot 2.3 = 345.4$. Also müssen wir $\frac{10^{150}}{345.5} \approx 3 \cdot 10^{147}$ Kandidaten testen.

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

Aufwand für die Faktorensuche

Wenn wir versuchen, die Faktoren einer Primzahl mit 300 Stellen zu finden, testen wir nur Primzahlen, die kleiner als $\sqrt{10^{300}} = 10^{150}$ sind. Nach der Abschätzung von Euler gilt für große n, dass es ca. $\frac{n}{\ln(n)}$ Primzahlen unterhalb von n gibt.
$\ln(10^{150}) = 150 \cdot \ln(10) \approx 150 \cdot 2.3 = 345.4$. Also müssen wir $\frac{10^{150}}{345.5} \approx 3 \cdot 10^{147}$ Kandidaten testen.

Annahme: 1 Computer schaft 10^{12} Prüfungen pro Sekunde (1 Million Millionen). Das sind im Dauerbetrieb pro Jahr:
$10^{12} \cdot 60 \cdot 60 \cdot 24 \cdot 365 \approx 3 \cdot 10^{19}$ Prüfungen.

Aufwand für die Faktorensuche

Wenn wir versuchen, die Faktoren einer Primzahl mit 300 Stellen zu finden, testen wir nur Primzahlen, die kleiner als $\sqrt{10^{300}} = 10^{150}$ sind. Nach der Abschätzung von Euler gilt für große n, dass es ca. $\frac{n}{\ln(n)}$ Primzahlen unterhalb von n gibt.
$\ln(10^{150}) = 150 \cdot \ln(10) \approx 150 \cdot 2.3 = 345.4$. Also müssen wir $\frac{10^{150}}{345.5} \approx 3 \cdot 10^{147}$ Kandidaten testen.

Annahme: 1 Computer schaft 10^{12} Prüfungen pro Sekunde (1 Million Millionen). Das sind im Dauerbetrieb pro Jahr:
$10^{12} \cdot 60 \cdot 60 \cdot 24 \cdot 365 \approx 3 \cdot 10^{19}$ Prüfungen. Das bedeutet $\frac{3 \cdot 10^{147}}{3 \cdot 10^{19}} = 10^{128}$ Jahre für alle Prüfungen (Alter des Weltalls: ca. 10^{10} Jahre).

Aufwand für die Faktorensuche

Wenn wir versuchen, die Faktoren einer Primzahl mit 300 Stellen zu finden, testen wir nur Primzahlen, die kleiner als $\sqrt{10^{300}} = 10^{150}$ sind. Nach der Abschätzung von Euler gilt für große n, dass es ca. $\frac{n}{\ln(n)}$ Primzahlen unterhalb von n gibt.
$\ln(10^{150}) = 150 \cdot \ln(10) \approx 150 \cdot 2.3 = 345.4$. Also müssen wir $\frac{10^{150}}{345.5} \approx 3 \cdot 10^{147}$ Kandidaten testen.

Annahme: 1 Computer schaft 10^{12} Prüfungen pro Sekunde (1 Million Millionen). Das sind im Dauerbetrieb pro Jahr:
$10^{12} \cdot 60 \cdot 60 \cdot 24 \cdot 365 \approx 3 \cdot 10^{19}$ Prüfungen. Das bedeutet $\frac{3 \cdot 10^{147}}{3 \cdot 10^{19}} = 10^{128}$ Jahre für alle Prüfungen (Alter des Weltalls: ca. 10^{10} Jahre).

Wenn jeder Mensch einen Computer beisteuern würde und manche zwei kämen wir auf 10 Millarden $= 10^{10}$ Computer.

Aufwand für die Faktorensuche

Wenn wir versuchen, die Faktoren einer Primzahl mit 300 Stellen zu finden, testen wir nur Primzahlen, die kleiner als $\sqrt{10^{300}} = 10^{150}$ sind. Nach der Abschätzung von Euler gilt für große n, dass es ca. $\frac{n}{\ln(n)}$ Primzahlen unterhalb von n gibt.
$\ln(10^{150}) = 150 \cdot \ln(10) \approx 150 \cdot 2.3 = 345.4$. Also müssen wir $\frac{10^{150}}{345.5} \approx 3 \cdot 10^{147}$ Kandidaten testen.

Annahme: 1 Computer schaft 10^{12} Prüfungen pro Sekunde (1 Million Millionen). Das sind im Dauerbetrieb pro Jahr:
$10^{12} \cdot 60 \cdot 60 \cdot 24 \cdot 365 \approx 3 \cdot 10^{19}$ Prüfungen. Das bedeutet $\frac{3 \cdot 10^{147}}{3 \cdot 10^{19}} = 10^{128}$ Jahre für alle Prüfungen (Alter des Weltalls: ca. 10^{10} Jahre).

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

Kryptographische Hashfunktionen

Kryptographische Hashfunktionen

Hashfunktionen bilden Eingabewerte (z.B. ein Text oder eine Datei) auf einen Wert fester Länge ab, den *Hash* der Eingabe.

Kryptographische Hashfunktionen

Hashfunktionen bilden Eingabewerte (z.B. ein Text oder eine Datei) auf einen Wert fester Länge ab, den *Hash* der Eingabe. Beispiel: *SHA-256* bildet Eingaben auf eine Bitfolge der Länge 256 ab.

Kryptographische Hashfunktionen

Hashfunktionen bilden Eingabewerte (z.B. ein Text oder eine Datei) auf einen Wert fester Länge ab, den *Hash* der Eingabe. Beispiel: *SHA-256* bildet Eingaben auf eine Bitfolge der Länge 256 ab.

Kryptographische Hashfunktionen sind kollisionsresistene Einwegfunktionen.

Kryptographische Hashfunktionen

Hashfunktionen bilden Eingabewerte (z.B. ein Text oder eine Datei) auf einen Wert fester Länge ab, den *Hash* der Eingabe. Beispiel: *SHA-256* bildet Eingaben auf eine Bitfolge der Länge 256 ab.

Kryptographische Hashfunktionen sind kollisionsresistene Einwegfunktionen. Kollisionsresistent bedeutet, es ist praktisch unmöglich, zwei Eingaben zu finden, die denselben Hash ergeben.

Kryptographische Hashfunktionen

Hashfunktionen bilden Eingabewerte (z.B. ein Text oder eine Datei) auf einen Wert fester Länge ab, den *Hash* der Eingabe. Beispiel: *SHA-256* bildet Eingaben auf eine Bitfolge der Länge 256 ab.

Kryptographische Hashfunktionen sind kollisionsresistene Einwegfunktionen. Kollisionsresistent bedeutet, es ist praktisch unmöglich, zwei Eingaben zu finden, die denselben Hash ergeben. Einwegfunktion bedeutet, es ist praktisch unmöglich, aus dem Hashwert den Eingabewert zu rekonstruieren.

Digitale Signatur

Mit dem RSA-Verfahren und einer kryptographischen Hashfunktion kann eine digitale Signatur erstellt werden.

Digitale Signatur

Mit dem RSA-Verfahren und einer kryptographischen Hashfunktion kann eine digitale Signatur erstellt werden.

Alice m_A, e_A, d_A

N

$\downarrow$ berechnet

$$\left.\mathrm{Sig} = \mathrm{hash}(N)^{d_A} \bmod m_A \right\} \xrightarrow[N,\ \mathrm{Sig}]{\text{Schickt an Bob}}$$

Bob

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

Bob prüft

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

Durch eine digitale Signatur wird der Diffie-Hellman Schlüsselaustausch vor einem Man-in-the-middle-Angriff geschützt.

Durch eine digitale Signatur wird der Diffie-Hellman Schlüsselaustausch vor einem Man-in-the-middle-Angriff geschützt.

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

a $\qquad$ b

berechnet $\qquad$ berechnet

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

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

$\Longleftrightarrow$

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

prüfen

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

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