Ein Mini-Zahlenbeispiel zu RSA:

Sei $n = 3 \cdot 11 = 33$, $\varphi(n) = 2 \cdot 10 = 20 = 2^2 \cdot 5$

Wähle $e = 3 \in \mathbb{Z}^*_{\varphi(n)}$. Dann ist $d = 7$ invers zu 3, denn $3 \cdot 7 = 21 = 1 + 20$ (also $3 \cdot 7 \equiv 1 \pmod{20}$)

Alle folgenden Rechnungen sind modulo 33:

$m =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$m^e =$	1	8	27	31	26	18	13	17	3	10	11	12	19	5	9	4
$m^d =$	1	29	9	16	14	30	28	2	15	10	11	12	7	20	27	25

$m =$	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
$m^e =$	29	24	28	14	21	22	23	30	16	20	15	7	2	6	25	32
$m^d =$	8	6	13	26	21	22	23	18	31	5	3	19	17	24	4	32

Beispiel: Die Nachricht/das Dokument sei $m = 17$:

$m = 17$

verschlüsseln: $\Longrightarrow m^e = 29$ entschlüsseln: $\Longrightarrow 29^d = 17$

entschlüsseln: $\Longrightarrow m^d = 8$ verschlüsseln: $\Longrightarrow 8^e = 17$

$m^d = 8$ entschlüsseln $\hat{=}$ unterschreiben

$8^e = 17$ verschlüsseln $\hat{=}$ überprüfen

Zahlenbeispiel zum Rabin-System.

Sei $n = 3 \cdot 11 = 33$. Dann ist $|\mathbb{Z}_n^*| = \varphi(n) = 2 \cdot 10 = 20$

Es dürfen nur die Elemente aus $\mathbb{Z}_n^*$ als mögliche Nachrichten m verschlüsselt werden:

$$\mathbb{Z}_{33}^* = \{1, 2, 4, 5, 7, 8, 10, 13, 14, 16, 17, 19, 20, 23, 25, 26, 28, 29, 31, 32\}$$

Es gibt genau 5 Quadratzahlen: 1, 4, 16, 25, 31
– jede davon besitzt 4 Quadratwurzeln.

mögliche Nachrichten $m \in \mathbb{Z}_{33}^*$:	1	2	4	5	8
	10	13	7	16	14
	23	20	26	17	19
	32	31	29	28	25
$m^2 \bmod 33 =$	1	4	16	25	31

Beispiel: Der Empfänger erhält die Zahl 25.
Er kann die Q.Wurzeln 5, 16, 17, 28
bestimmen und muss entscheiden, welches davon
das ursprüngliche m ist.

Zahlenbeispiel zum ElGamal-Verfahren:

$n = 31$ ist Primzahl. $n-1 = 30 = 2 \cdot 3 \cdot 5$.

Um eine Zahl a auf Primitivwurzel zu testen, muss gelten: $\underbrace{a^{\frac{30}{2}} = a^{15},\ a^{\frac{30}{3}} = a^{10},\ a^{\frac{30}{5}} = a^{6}}$ müssen alle $\not\equiv 1 \pmod{31}$ sein.

Für $a = 3$ trifft dies zu, also $\langle 3 \rangle = \mathbb{Z}_{31}^*$.

$n = 31$ und $a = 3$ sind öffentlich.

Teilnehmer B wählt die Zufallszahl $y = 7 \in \mathbb{Z}_{31}^*$ als geheimer Schlüssel und berechnet $\tilde{y} = a^y \bmod n = 3^7 \bmod 31 = 17$ als öffentlicher Schlüssel.

A besorgt sich den öffentlichen Schlüssel $\tilde{y} = 17$ von B. Er möchte die Nachricht $m = 23$ an B senden. A wählt Zufallszahl $x = 16 \in \mathbb{Z}_{31}^*$ und berechnet $\tilde{x} = a^x \bmod n = 3^{16} \bmod 31 = 28$.

Damit ergibt sich der „Schlüssel" $s = (\tilde{y})^x \bmod n = 17^{16} \bmod 31 = 14$. Er berechnet $c = s \cdot m \bmod n = 23 \cdot 14 \bmod 31 = 12$ und sendet $(\tilde{x}, c) = (28, 12)$ an B. B berechnet $s = (\tilde{x})^y \bmod n = 28^7 \bmod 31 = 14$ sowie $s^{-1} = 20$. Damit ergibt sich $m = c \cdot s^{-1} \bmod n = 12 \cdot 20 \bmod 31 = 23$.

Alternative: Anstatt die Nachricht m mit s zu multiplizieren (verschlüsseln) und beim Empfänger mit s^{-1} zu multiplizieren (entschlüsseln), kann man die Zahlen $\in \mathbb{Z}_{31}^*$ als 5-Bitstrings betrachten. Dann ist die Nachricht $m = 23 = (10111)_2$ und der Schlüssel $s = 14 = (01110)_2$. Nun könnte man auch mittels XOR verschlüsseln: $c = m \oplus s = (11001)_2$. Also sendet A an B: $(\tilde{x}, c) = (28, 11001)$. Der Empfänger B kann ebenfalls XOR verwenden: $m = c \oplus s = (10111)_2$.

Aufgabe: Sei $n = p \cdot q$ für 2 Primzahlen, ungefähr von 500 Bit Länge. Wie groß ist etwa die Wahrscheinlichkeit, bei zufälliger Wahl von $x < n$ ein $x \notin \mathbb{Z}_n^*$ zu erhalten?

Antwort: Wir rechnen mit $p \approx 2^{500}$, $q \approx 2^{500}$, $n \approx 2^{1000}$. Die gesuchte W'keit ist $1 - \frac{\varphi(n)}{n-1} \approx 1 - \frac{(p-1)(q-1)}{n-1}$

$$\approx 1 - \frac{2^{1000} - 2 \cdot 2^{500}}{2^{1000}} = \underline{\underline{2^{-499}}}.$$

Unterschreiben in einem El Gamal Public Key System

Wir nehmen an, Teilnehmer B soll ein Dokument m unterschreiben.

Wie bereits beschrieben, besitzt B einen geheimen Schlüssel $y < n$ (zufällig gewählt), sowie einen öffentlichen Schlüssel $\tilde{y} = a^y$. (Alle Rechnungen sind modulo n, wenn nicht anders gesagt.)

Hier ist der Vorgang des Unterschreibens, der wiederum einen Aspekt der Randomisierung enthält: B wählt eine Zufallszahl $z \in \mathbb{Z}_{n-1}^*$ und berechnet $\tilde{z} = a^z$. Nun löst B die folgende Gleichung nach x auf (wobei die Rechnung nun modulo $n-1$ ist):

$$m \equiv y \cdot \tilde{z} + z \cdot x \pmod{n-1}$$

Dies gelingt dadurch, dass man z^{-1} (modulo $n-1$) bestimmt (mittels Extended Euklid) und mit z^{-1} multipliziert:

$$z^{-1} \cdot m \equiv z^{-1} \cdot y \cdot \tilde{z} + x \pmod{n-1}$$

$$x \equiv z^{-1} \cdot (m - y \cdot \tilde{z}) \pmod{n-1}$$

Die Unterschrift besteht aus $(\tilde{z}, x)$. Insgesamt also, Dokument + Unterschrift: $(m; (\tilde{z}, x))$.

Um die Unterschrift zu verifizieren, also dass diese von B stammt und mit m zusammenhängt, sollte der Verifizierende nur Informationen verwenden, die öffentlich bekannt sind, nämlich:

$$\underbrace{n, a}_{\text{allgem. Initialisierung}}, \underbrace{\tilde{y}}_{\text{öffentl. Schlüssel von B}}, \underbrace{\tilde{z}, x}_{\text{die Unterschrift}}, \underbrace{m}_{\text{das Dokument}}$$

Der Test lautet:

$$a^m \stackrel{?}{=} (\tilde{y})^{\tilde{z}} \cdot (\tilde{z})^x$$

Bestätigung des Tests: Da $m \equiv y \cdot \tilde{z} + z \cdot x \pmod{n-1}$ folgt:

$$\begin{aligned} a^m &\equiv a^{y \cdot \tilde{z} + z \cdot x} \\ &\equiv a^{y \cdot \tilde{z}} \cdot a^{z \cdot x} \\ &\equiv (\tilde{y})^{\tilde{z}} \cdot (\tilde{z})^x \qquad \checkmark \end{aligned}$$

Signieren mit Hashfunktionen

Unterschrift ist (bisher) solang wie Dokument selbst, bei ElGamal sogar doppelt so lang.

Hashing verwenden: Anstatt m selbst zu unterschreiben, welches unkontrollierbar lang sein kann – und ebenso die Unterschrift, verwende eine öffentlich bekannte Hashfunktion $h: \{0,1\}^* \to \{0,1\}^k$ und unterschreibe stattdessen $h(m)$ (statt m)

Bsp: Wenn B der Unterzeichner ist und dieser hat öffentlichen/geheimen Schlüssel k_B / k_B', so ist Dokument m mit Unterschrift von B versehen nun: $(m, \underbrace{D(k_B', h(m))}_{m'})$

Die Verifikation verläuft so: $h(m) = E(k_B, m')$

Nun ist: m ... evtl lang, viele Blöcke

$m', h(m)$ --- kurz, 1 Block

Eine kryptographisch sichere Hashfunktion sollte folgende Eigenschaften haben:

- sollte Einwegfunktion sein.
  Zu gegebenem y sollte es unmöglich *) sein, ein x zu finden mit $h(x) = y$.

- sollte (schwach) kollisionsresistent sein:
  Zu gegebenem x soll es unmöglich *) sein, ein $x' \neq x$ zu finden, so dass $h(x) = h(x')$.

- sollte evtl. sogar stark kollisionsresistent sein; es soll unmöglich *) sein, 2 verschiedene x, x' zu finden mit $h(x) = h(x')$.

*) relativ zu den Berechnungsressourcen des Angreifers.

Aufgabe: In der Algorithmik wird oft folgende Hashfunktion $h: \mathbb{N} \to \{0,1,\ldots,m-1\}$ verwendet: $h(x) = x \bmod m$.

Man zeige, dass h weder eine Einwegfunktion, noch (stark oder schwach) kollisionsresistent ist.

Antwort: Ein Urbild von $y \in \{0,1,\ldots,m-1\}$ ist z.B. y selbst. Daher ist h kein Einwegfkt.

Für beliebige x haben x und $x+m$ denselben Hashwert ($\Rightarrow$ nicht stark koll. resistent). Dementsprechend auch nicht schwach koll. resistent.

Eine (möglicherweise) kollisionsresistente Hashfunktion:

p Primzahl; $q = \frac{p-1}{2}$ auch Primzahl

a Primitivwurzel mod p.

b zufällig gewählt, $b \in \{1, 2, \ldots, p-1\}$

$h : \{0, 1, \ldots, q-1\}^2 \longrightarrow \{1, \ldots, p-1\}$

wobei $h(x_1, x_2) = a^{x_1} b^{x_2} \bmod p$.

Analyse der Bitlängen:

p habe m Bits

$\Rightarrow$ q hat $m-1$ Bits

h bildet $2(m-1)$ lange Bitstrings auf m-Bitstrings ab. (Reduzierung auf etwa die Hälfte).

Es gilt:

Diskret. Log zur Basis a berechnen $\leq_{\text{eff.}}$ Auffinden einer Kollision für obige Hashfunktion

Thematik: Hashing und das Geburtstagsproblem

Das Geburtstags-„Problem": Bei wievielen (zufällig ausgewählten) Personen ist es „wahrscheinlich", dass hierbei irgend-zwei am selben Tag im Jahr Geburtstag haben?

Je nachdem, wie man „wahrscheinlich" genau interpretiert, kommt man auf 23 ... 28 Personen.

Allgemein: m (statt 365) mögliche Hashwerte (also $m = |W(\text{Hashfunktion})|$) und n (anstatt Personenzahl) Einträge in Hashtabelle.

Der allgemeine Zusammenhang zwischen m und n bis eine „Kollision", „wahrscheinlich" wird, liegt bei $\boxed{c \cdot \sqrt{m} = n}$ $(c \in [1{,}178 \ldots 1{,}414])$

1. Rechnung: $P(\text{alle } n \text{ Hashwerte sind verschieden})$

$$= 1 \cdot \left(1 - \tfrac{1}{m}\right) \cdot \left(1 - \tfrac{2}{m}\right) \cdot \ldots \cdot \left(1 - \tfrac{n-1}{m}\right) = \prod_{i=1}^{n-1} \left(1 - \tfrac{i}{m}\right)$$

$$\approx \prod_{i=1}^{n-1} e^{-i/m} = e^{-\sum_{i=1}^{n-1} i/m} = e^{-n(n-1)/2m} \approx e^{-(n-0{,}5)^2/2m}$$

Diese W'heit auf $\frac{1}{2}$ setzen (Median), ergibt

$n = 0{,}5 + 1{,}1774 \cdot \sqrt{m}$ (bei $m = 365$ folgt $n = 23$).

2. Rechnung: Seien $a_1, \dots, a_n$ zufällig aus $\{1, \dots, m\}$ gezogen (mit Zurücklegen).

Zufallsvariable: $X_{i,j} = \begin{cases} 1, & a_i = a_j \\ 0, & \text{sonst} \end{cases}$ $\quad E(X_{i,j}) = \frac{1}{m}$

$$E(\text{Anzahl der Kollisionen}) = E\left(\sum_{i<j} X_{i,j}\right) = \sum_{i<j} E(X_{i,j})$$

$$= \sum_{i<j} P(a_i = a_j) = \sum_{i<j} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m} = \frac{n(n-1)}{2m}$$

$\approx \frac{(n-0{,}5)^2}{2m}$. Diesen Wert auf 1 setzen. Dies ergibt: $n \approx 0{,}5 + 1{,}414 \cdot \sqrt{m}$. (Bei $m = 365 \Rightarrow n \approx 28$).

3. Rechnung: Definiere Zufallsvariable X = der erste Wert n so dass eine Dublette auftritt, also $a_n = a_j$ für ein $j < n$.

$$E(X) = \sum_{n=2}^{\infty} n \cdot P(X=n) = \sum_{n=2}^{\infty} P(X \geq n) = \sum_{n=1}^{\infty} P(X > n)$$

bedeutet, dass alle Werte $a_1 \dots a_n$ voneinander verschieden sind.

Bei erster Rechnung wurde bereits ausgerechnet:

$$P(X > n) \approx e^{-(n-0{,}5)^2/2m}$$

Somit $E(X) \approx \sum_{n=1}^{\infty} e^{-(n-0{,}5)^2/2m} \approx \int_1^{\infty} e^{-(x-\frac{1}{2})^2/2m} \, dx$

$$E(X) \approx \frac{2}{3} + \sqrt{\frac{\pi}{2} m} \approx 0{,}667 + 1{,}253 \cdot \sqrt{m}$$

Bei $m = 365$ ergibt sich $n \approx 25$.

Diese Abschätzungen werden z.B. bei der Analyse des Pollard ρ-Algorithmen zur Faktorisierung bzw. für den Diskreten Logarithmus benötigt.

Bei Verwendung von Hashing beim Signieren spielt die Abwehr von „Geburtstagsangriffen" eine Rolle.

Ein Dokument m soll unterschrieben werden. Man kann aus m viele Varianten erzeugen $m_1, m_2, m_3, \ldots$ (Einfügen von Blanks in den Text).

Der Unterschreiber würde jedes dieser inhaltlich gleichartigen Dokumente unterschreiben, wenn man es vorlegt.

Analog kann man aus einem „bösartigen" Dokument m', das der Unterschreiber nicht unterschreiben würde,

viele Varianten erzeugen: $m_1', m_2', m_3', \ldots$

Der Angreifer sucht nun nach i,j so dass $h(m_i) = h(m_j')$. Die Anzahl der Dokumente n so dass $i,j \leq n$, die man bereitstellen sollte, so dass es „wahrscheinlich" ist, eine solche Kollision zu finden, beträgt $O(\sqrt{n})$. (Die versteckte Konstante in der O-Notation liegt knapp über 1.)

Man legt nun m_i dem Unterschreiber vor, dieser unterschreibt; später wird behauptet, er habe m_j' unterschrieben, was er nur schwer abstreiten kann, wenn $h(x_i) = h(x_j')$.

Konsequenz: Will man ein Sicherheitsniveau von 80 Bit erreichen, so sollte man Hashwerte von mindestens 160 Bit verwenden,

denn $\sqrt{2^{160}} = 2^{80}$ ...