

Woche 10: 4. Juli 2024

Thema: *Public-Key Cryptographie*

Woche 10: 4. Juli 2024

Thema: *Public-Key Cryptographie*

10.1 Wiederholung

Lehrevaluation

Lehrevaluation.

<https://befragung.tu-berlin.de/evasys/online.php?pswd=UT5DX>



Wiederholung: Modulare Arithmetik

Wir rechnen Modulo m für $m \geq 2$

Für $a \in \mathbb{Z}$ existieren $g \in \mathbb{Z}$, $r \in \{0, \dots, m-1\}$

sodass $a = gm + r$ Damit gilt $(a \bmod m) = r$

Notation. $\mathbb{Z}_m := \{0, \dots, m-1\}$.

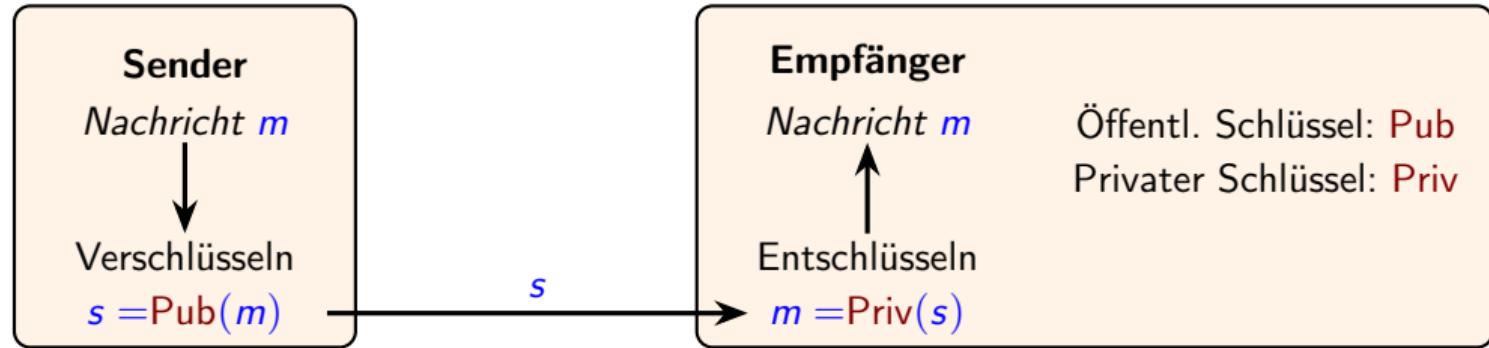
Es gilt $a \equiv b \pmod{m}$ genau wenn $m | (a - b)$

10.2 Public-Key Cryptography

Kryptographische Protokolle

Internetkommunikation. Viele Anwendungen des e-Commerce ebenso wie Kommunikation via EMails oder Messenger Services bringen es mit sich, dass Personen verschlüsselt kommunizieren wollen, die sich nicht kennen und daher auch keinen Schlüssel austauschen können.

Asymmetrische Verschlüsselung. Der Empfänger hat zwei Schlüssel, einen *öffentlichen* und einen *geheimen*.



Assymetrische Kodierungsverfahren

Assymetrische Kodierungsverfahren. Das Konzept wurde 1976 von *Diffie* und *Hellman* eingeführt.

Als Standard durchgesetzt hat sich aber das nach *Rivest*, *Shamir* und *Adleman* benannte *RSA*-Verfahren.

Das RSA-Verfahren funktioniert wie folgt.

Assymetrische Kodierungsverfahren

Assymetrische Kodierungsverfahren. Das Konzept wurde 1976 von **Diffie** und **Hellman** eingeführt.

Als Standard durchgesetzt hat sich aber das nach **Rivest**, **Shamir** und **Adleman** benannte **RSA**-Verfahren.

Das RSA-Verfahren funktioniert wie folgt.

Schlüsselgenerierung durch Empfänger:in.

- Wähle zwei Primzahlen p und q und berechne $n = p \cdot q$.
- Berechne $\varphi(n) = (p - 1) \cdot (q - 1)$. ← Eulersche φ -Funktion
- Berechne Zahlen k und ℓ mit
 $\text{ggT}(k, \varphi(n)) = 1$ und $k \cdot \ell \equiv 1 \pmod{\varphi(n)}$.

Öffentlicher Schlüssel. n, k ← damit wird verschlüsselt

Geheimer Schlüssel. ℓ ← damit wird entschlüsselt

Assymetrische Kodierungsverfahren

Assymetrische Kodierungsverfahren. Das Konzept wurde 1976 von *Diffie* und *Hellman* eingeführt.

Als Standard durchgesetzt hat sich aber das nach *Rivest*, *Shamir* und *Adleman* benannte *RSA*-Verfahren.

Das RSA-Verfahren funktioniert wie folgt.

Schlüsselgenerierung durch Empfänger:in.

- Wähle zwei Primzahlen p und q und berechne $n = p \cdot q$.
- Berechne $\varphi(n) = (p - 1) \cdot (q - 1)$.
- Berechne Zahlen k und ℓ mit
 $\text{ggT}(k, \varphi(n)) = 1$ und $k \cdot \ell \equiv 1 \pmod{\varphi(n)}$.

Öffentlicher Schlüssel. n, k

Geheimer Schlüssel. ℓ

Hinweis.

Die Zahlen $p, q, \varphi(n)$ und ℓ müssen geheim gehalten werden, das Schlüsselpaar (n, k) wird veröffentlicht.

Das RSA-Verfahren

Schlüsselgenerierung durch Empfänger:in.

- Wähle zwei Primzahlen p und q und berechne $n = p \cdot q$.
- Berechne $\varphi(n) = (p - 1) \cdot (q - 1)$.
- Berechne Zahlen k und ℓ mit
 $\text{ggT}(k, \varphi(n)) = 1$ und $k \cdot \ell \equiv 1 \pmod{\varphi(n)}$.

Öffentlicher Schlüssel. n, k

Geheimer Schlüssel. ℓ

Das RSA-Verfahren

Schlüsselgenerierung durch Empfänger:in.

- Wähle zwei Primzahlen p und q und berechne $n = p \cdot q$.
- Berechne $\varphi(n) = (p - 1) \cdot (q - 1)$.
- Berechne Zahlen k und ℓ mit
 $\text{ggT}(k, \varphi(n)) = 1$ und $k \cdot \ell \equiv 1 \pmod{\varphi(n)}$.

Öffentlicher Schlüssel. n, k

Geheimer Schlüssel. ℓ

Ver- und Entschlüsseln.

Nachricht: $M \in \{0, \dots, n - 1\}$

Verschlüsseln: $S := M^k \pmod{n}$

Gesendete Nachricht: S

Entschlüsseln: $M' := S^\ell \pmod{n}$

Das RSA-Verfahren

Schlüsselgenerierung durch Empfänger:in.

- Wähle zwei Primzahlen p und q und berechne $n = p \cdot q$.
- Berechne $\varphi(n) = (p - 1) \cdot (q - 1)$.
- Berechne Zahlen k und ℓ mit
 $\text{ggT}(k, \varphi(n)) = 1$ und $k \cdot \ell \equiv 1 \pmod{\varphi(n)}$.

Öffentlicher Schlüssel. n, k

Geheimer Schlüssel. ℓ

Ver- und Entschlüsseln.

Nachricht: $M \in \{0, \dots, n - 1\}$

Verschlüsseln: $S := M^k \pmod{n}$

Gesendete Nachricht: S

Entschlüsseln: $M' := S^\ell \pmod{n}$

Fragen.

- *Korrektheit.* Gilt immer $M' = M$.
- *Sicherheit.* Kann man den Code knacken?
- *Implementierung.* Wie aufwendig ist das Verfahren?

10.3 Die Zahlentheorie hinter RSA

Wiederholung: Der Algorithmus von Euklid

Algorithmus: Euklidischer Algorithmus.

Eingabe: Zahlen $m, n \in \mathbb{N}$ mit $m \leq n$.

Ausgabe: ggT(m, n).

Algorithmus Euklid(m, n)

Wenn $m \mid n$ dann

gib m zurück

sonst

gibt Euklid($n \bmod m, m$) zurück.

Beispiel.

n	m	$(n \bmod m)$
24948	8712	7524
8712	7524	1188
7524	1188	396
1188	396	0 (da $\frac{1188}{396} = 3$)

Lemma. Sind $m, n \in \mathbb{N}$ mit $m \leq n$ und $m \nmid n$, so gilt

$$\text{ggT}(m, n) = \text{ggT}(n \bmod m, m).$$

Wiederholung: Erweiterter Euklidischer Algorithmus

Erweiterter Euklidischer Algorithmus.

Eingabe: $m, n \in \mathbb{N}$ mit $m \leq n$.

Ausgabe: $x, y \in \mathbb{Z}$ mit $\text{ggT}(m, n) = mx + ny$.

Algorithmus Erw-Euklid(m, n)

dann hier
griff
 $\text{ggT}(m, n) =$ Wenn $m | n$ dann
sonst gib $(1, 0)$ zurück

$m = 1 \cdot m + 0 \cdot n$ sei $(x', y') = \text{Erw-Euklid}(n \bmod m, m)$.
sei $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$
 $y := x'$
gib (x, y) zurück.

$$\begin{aligned} x'm + y'n &= (y' - x' \cdot \lfloor \frac{n}{m} \rfloor) \cdot m + x' n \\ &= y'm - x' \cdot \lfloor \frac{n}{m} \rfloor \cdot m + x' n \\ &= y'm + (\underbrace{n - \lfloor \frac{n}{m} \rfloor m}_{\text{und } m}) \cdot x' \\ &= \text{ggT}(n, m) \end{aligned}$$

Satz.

Für alle $m, n \in \mathbb{N}$ mit $m \leq n$ gibt es $x, y \in \mathbb{Z}$ mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe (m, n) solche Zahlen zurück.

Wiederholung: Erweiterter Euklidischer Algorithmus

Rate k und prüfen, ob $\text{ggT}(k, \varphi(n)) = 1$ mit dem euklidischen Algorithmus

Berechnen von k und ℓ . Der erweiterte Euklidische Algorithmus erlaubt uns, k und ℓ auszurechnen.

Um ℓ zu berechnen, kann man Erw-Euklid($\varphi(n), k$) ausführen, der Algorithmus liefert x, y mit $x\varphi(n) + yk = 1$

$$\begin{aligned} x\varphi(n) &\equiv 0 \pmod{\varphi(n)} \\ yk &\equiv 1 \pmod{\varphi(n)} \end{aligned}$$

$\ell = y$ erfüllt die Bedingung

Wiederholung: Erweiterter Euklidischer Algorithmus

Schlüsselgenerierung.

- Wähle Primzahlen p, q . Berechne $n = p \cdot q$.
 - Berechne $\varphi(n) = (p - 1) \cdot (q - 1)$.
 - Berechne Zahlen k und ℓ mit
 $\text{ggT}(k, \varphi(n)) = 1$ und $k \cdot \ell \equiv 1 \pmod{\varphi(n)}$.
- Öffentlicher Schlüssel.* n, k
Geheimer Schlüssel. ℓ

Berechnen von k und ℓ . Der erweiterte Euklidische Algorithmus erlaubt uns, k und ℓ auszurechnen.

10.4 Lineare Gleichungssysteme in der Modularen Arithmetik

Lineare Gleichungen in der modularen Arithmetik

Lösen linearer Gleichungssysteme.

Für reelle Zahlen hat eine lineare Gleichung $a \cdot x = b$ für $a \neq 0$ genau eine Lösung.

In der modularen Arithmetik gilt das nicht mehr.

Beispiele. Ist x_0 eine Lösung von $a \cdot x \equiv b \pmod{m}$, dann ist auch $x_0 + k \cdot m$ eine Lösung für alle $k \in \mathbb{N}$.

Lineare Gleichungen in der modularen Arithmetik

Lösen linearer Gleichungssysteme.

Für reelle Zahlen hat eine lineare Gleichung $a \cdot x = b$ für $a \neq 0$ genau eine Lösung.

In der modularen Arithmetik gilt das nicht mehr.

Beispiele. Ist x_0 eine Lösung von $a \cdot x \equiv b \pmod{m}$, dann ist auch $x_0 + k \cdot m$ eine Lösung für alle $k \in \mathbb{N}$.

Für $m = 6$ und $a = 2$ hat

- $2x \equiv 3 \pmod{6}$ keine Lösung.

Lineare Gleichungen in der modularen Arithmetik

Lösen linearer Gleichungssysteme.

Für reelle Zahlen hat eine lineare Gleichung $a \cdot x = b$ für $a \neq 0$ genau eine Lösung.

In der modularen Arithmetik gilt das nicht mehr.

Beispiele. Ist x_0 eine Lösung von $a \cdot x \equiv b \pmod{m}$, dann ist auch $x_0 + k \cdot m$ eine Lösung für alle $k \in \mathbb{N}$.

Für $m = 6$ und $a = 2$ hat

- $2x \equiv 3 \pmod{6}$ keine Lösung.
- $2x \equiv 4 \pmod{6}$ zwei Lösungen in \mathbb{Z}_m : $x = 2$ und $x = 5$.

Lineare Gleichungen in der modularen Arithmetik

Lösen linearer Gleichungssysteme.

Für reelle Zahlen hat eine lineare Gleichung $a \cdot x = b$ für $a \neq 0$ genau eine Lösung.

In der modularen Arithmetik gilt das nicht mehr.

Beispiele. Ist x_0 eine Lösung von $a \cdot x \equiv b \pmod{m}$, dann ist auch $x_0 + k \cdot m$ eine Lösung für alle $k \in \mathbb{N}$.

Für $m = 6$ und $a = 2$ hat

- $2x \equiv 3 \pmod{6}$ keine Lösung.
- $2x \equiv 4 \pmod{6}$ zwei Lösungen in \mathbb{Z}_m : $x = 2$ und $x = 5$.

Satz. Seien $a, m \in \mathbb{Z}$. Wenn $m \geq 2$ und $\text{ggT}(a, m) = 1$, dann hat $ax \equiv b \pmod{m}$ für jedes $b \in \mathbb{Z}$ genau eine Lösung in \mathbb{Z}_m .

Lösen linearer Gleichungen in der modularen Arithmetik

Satz. Seien $a, m \in \mathbb{Z}$. Wenn $m \geq 2$ und $\text{ggT}(a, m) = 1$, dann hat $ax \equiv b \pmod{m}$ für jedes $b \in \mathbb{Z}$ genau eine Lösung in \mathbb{Z}_m .

Wir wissen, dass $\forall a, b \in \mathbb{Z} \exists x, y \in \mathbb{Z} : \text{ggT}(a, b) = xa + yb$

Fall $b=1$: es existieren $x_1, y_1 \in \mathbb{Z} : ax_1 + my_1 = \text{ggT}(a, m) = 1$

Damit gilt für alle $k \in \mathbb{Z} : a(x_1 + km) + m(y_1 - ks) = 1$

es ist also möglich, x_1 so zu wählen, dass $x_1 \in \mathbb{Z}_n$ gilt

Wegen $my_1 \equiv 0 \pmod{m}$ folgt $ax_1 \equiv 1 \pmod{m}$

Somit ist x_1 eine Lösung für $ax \equiv b \pmod{m}$
im Fall $b=1$

Lösen linearer Gleichungen in der modularen Arithmetik

$b \in \mathbb{Z}$ (nachw.)

Satz. Seien $a, m \in \mathbb{Z}$. Wenn $m \geq 2$ und $\text{ggT}(a, m) = 1$, dann hat
 $ax \equiv b \pmod{m}$ für jedes $b \in \mathbb{Z}$ genau eine Lösung in \mathbb{Z}_m .

Fall $b \in \mathbb{Z}$: Wir wissen $a \not\equiv 1 \pmod{m}$ hat Lösung x_1

Es folgt $ax_1 \equiv b \pmod{m}$ hat Lösung $x_1 \cdot b$

$x_2 := (x_1 \cdot b \pmod{m})$ ist eine Lösung in \mathbb{Z}_m

Damit ist die Existenz einer Lösung gezeigt

Lösen linearer Gleichungen in der modularen Arithmetik

Satz. Seien $a, m \in \mathbb{Z}$. Wenn $m \geq 2$ und $\text{ggT}(a, m) = 1$, dann hat $ax \equiv b \pmod{m}$ für jedes $b \in \mathbb{Z}$ genau eine Lösung in \mathbb{Z}_m .

Lemma. Für alle $a, b, c, d, m \in \mathbb{Z}$ mit $m \geq 2$ gilt: Wenn

$$\underline{a \equiv b \pmod{m}}$$

und

$$\underline{c \equiv d \pmod{m}}$$

dann

$$\underline{(a + c) \equiv (b + d) \pmod{m}}$$

und

$$\underline{a \cdot c \equiv b \cdot d \pmod{m}}.$$

Lösen linearer Gleichungen in der modularen Arithmetik

Satz. Seien $a, m \in \mathbb{Z}$. Wenn $m \geq 2$ und $\text{ggT}(a, m) = 1$, dann hat $ax \equiv b \pmod{m}$ für jedes $b \in \mathbb{Z}$ genau eine Lösung in \mathbb{Z}_m .

Nun beweisen wir die Eindeutigkeit der Lösung.
 Für jedes $b \in \mathbb{Z}_m$ finden wir Lösung x_b wie oben gezeigt.
 Für $b, b' \in \mathbb{Z}_m$ mit $b \neq b'$ gilt $x_b \neq x_{b'}$
 denn aus $x_b = x_{b'}$ folgt $b \equiv ax_b \equiv ax_{b'} \equiv b' \pmod{m}$
 Jedes Element aus \mathbb{Z}_m ist also die Lösung von $ax \equiv b \pmod{m}$
 für genau ein $b \in \mathbb{Z}_m$. Also ist die Lösung eindeutig für $b \in \mathbb{Z}_m$
 Für $b \in \mathbb{Z}$ hat $ax \equiv b \pmod{m}$ die gleichen Lösungen
 wie für $ax \equiv (b \pmod{m}) \pmod{m}$. Also ist die Lösung
 auch hier eindeutig. \square

Der Chinesische Restsatz

Der nächste Satz, bekannt als Chinesischer Restsatz, erweitert die Existenz einer Lösung einer linearen Gleichung auf Systeme linearer Gleichungen in der modularen Arithmetik.

Chinesische Restsatz. Wenn $b_1, \dots, b_k, m_1, \dots, m_k$ natürliche Zahlen sind mit $\text{ggT}(m_i, m_j) = 1$, für alle $1 \leq i < j \leq k$, dann gibt es für $m = m_1 \cdot m_2 \cdot \dots \cdot m_k$ genau ein $x \in \mathbb{Z}_m$ mit

$$x \equiv b_i \pmod{m_i} \quad \text{für alle } i = 1, 1, \dots, k.$$

$$\begin{aligned} x &\equiv b_1 \pmod{m_1} \\ x &\equiv b_2 \pmod{m_2} \\ &\vdots \\ x &\equiv b_n \pmod{m_n} \end{aligned}$$

gesucht ist $x \in \mathbb{Z}_n$,
da alle Kongruenzen
erfüllt.

Der Chinesische Restsatz

Chinesische Restsatz. Wenn $b_1, \dots, b_k, m_1, \dots, m_k$ natürliche Zahlen sind mit

$\text{ggT}(m_i, m_j) = 1$, für alle $1 \leq i < j \leq k$, dann gibt es für $m = m_1 \cdot m_2 \cdot \dots \cdot m_k$ genau ein $x \in \mathbb{Z}_m$ mit $x \equiv b_i \pmod{m_i}$ für alle $i = 1, 2, \dots, k$.

Wir definieren die Abbildung

$$\mu: \mathbb{Z}_m \rightarrow \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \dots \times \mathbb{Z}_{m_k}$$

$$x \mapsto (x \pmod{m_1}, x \pmod{m_2}, \dots, x \pmod{m_k})$$

Wir wollen zeigen, dass μ bijektiv ist, denn dann folgt, dass es für jedes Tupel $(b_1, \dots, b_k) \in \mathbb{Z}_{m_1} \times \dots \times \mathbb{Z}_{m_k}$ genau ein x gibt mit $x \equiv b_1 \pmod{m_1}, \dots, x \equiv b_k \pmod{m_k}$.

$$\text{Es gilt } |\mathbb{Z}_m| = |\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \dots \times \mathbb{Z}_{m_k}|$$

Und deshalb reicht es, die Injektivität von μ zu zeigen.

Der Chinesische Restsatz

Chinesische Restsatz. Wenn $b_1, \dots, b_k, m_1, \dots, m_k$ natürliche Zahlen sind mit

$\text{ggT}(m_i, m_j) = 1$, für alle $1 \leq i < j \leq k$, dann gibt es für $m = m_1 \cdot m_2 \cdot \dots \cdot m_k$ genau ein $x \in \mathbb{Z}_m$ mit $x \equiv b_i \pmod{m_i}$ für alle $i = 1, 2, \dots, k$.

Seien $x, y \in \mathbb{Z}_m$ mit $\mu(x) = \mu(y)$. Dann gilt für $1 \leq i \leq k$:

$$(x \pmod{m_i}) = (y \pmod{m_i})$$

Da dies bedeutet $m_i | (x - y)$ für alle $1 \leq i \leq k$. Da die Zahlen m_1, \dots, m_k teilerfremd sind folgt $m | (x - y)$. Wir wissen, dass $(x - y) \in \{-m+1, \dots, 0, \dots, m-1\}$. Der einzige Wert in dieser Menge, der möglich ist, ist 0.

Also gilt $(x - y) = 0$ und damit $x = y$. \square

10.5 Die Eulersche φ -Funktion und der kleine Satz von Fermat

Der kleine Satz von Fermat

Satz ("kleiner Satz von Fermat"). Für alle $n \in \mathbb{N}$ mit $n \geq 2$ gilt:

n ist eine Primzahl gdw. $a^{n-1} \equiv 1 \pmod{n}$ für alle $a \in \mathbb{Z}_n \setminus \{0\}$.

Beweis der Rückwärtsrichtung: Wir nehmen an, dass $a^{n-1} \equiv 1 \pmod{n}$ für alle $a \in \mathbb{Z}_n \setminus \{0\}$ gilt. Sei $g \neq 1$ ein beliebiger positiver Teiler von n . Es gibt also $k \in \mathbb{N}$ mit $k \cdot g = n$. Jeder derartige Teiler von n muss in $\mathbb{Z}_n \setminus \{0\}$ sein. Also gilt $g^{n-1} \equiv 1 \pmod{n}$. Daraus folgt $g^{n-1} - 1 \equiv 0 \pmod{n}$. Es gibt $\ell \in \mathbb{N}$ mit $\underline{g^{n-1} - 1 = \ell n = \ell \cdot k \cdot g}$. Die einzige Zahl, für die diese Gleichheit möglich ist, ist $g=1$. Also ist 1 der einzige positive Teiler von n und n ist prim.

Der kleine Satz von Fermat

Satz ("kleiner Satz von Fermat"). Für alle $n \in \mathbb{N}$ mit $n \geq 2$ gilt:

n ist eine Primzahl gdw. $a^{n-1} \equiv 1 \pmod{n}$ für alle $a \in \mathbb{Z}_n \setminus \{0\}$.

Beweis der Hinrichtung.

Der Satz wurde von Leonhard Euler bewiesen, der zum Beweis der Hinrichtung ein etwas allgemeineres Ergebnis bewiesen hat.

Dazu brauchen wir zunächst noch etwas Notation.

Der kleine Satz von Fermat

Satz ("kleiner Satz von Fermat"). Für alle $n \in \mathbb{N}$ mit $n \geq 2$ gilt:

n ist eine Primzahl gdw. $a^{n-1} \equiv 1 \pmod{n}$ für alle $a \in \mathbb{Z}_n \setminus \{0\}$.

Beweis der Hinrichtung.

Der Satz wurde von Leonhard Euler bewiesen, der zum Beweis der Hinrichtung ein etwas allgemeineres Ergebnis bewiesen hat.

Dazu brauchen wir zunächst noch etwas Notation.

Definition. Wir definieren

$$\mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \setminus \{0\} : \text{ggT}(a, n) = 1\}$$

und, für $n \geq 2$,

$$\varphi(n) := |\mathbb{Z}_n^*|.$$

Eulersche φ -Funktion

$\varphi(n)$ ist also die Anzahl der zu n teilerfremden Zahlen $\leq n$.

Die Eulersche φ -Funktion

Definition. Wir definieren

$$\mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \setminus \{0\} : \text{ggT}(a, n) = 1\}$$

und, für $n \geq 2$,

$$\varphi(n) := |\mathbb{Z}_n^*|.$$

$\varphi(n)$ ist also die Anzahl der zu n teilerfremden Zahlen $\leq n$.

Lemma. Sei $n = p_1^{e_1} \cdot \dots \cdot p_k^{e_k}$ für paarweise verschiedene Primzahlen p_1, \dots, p_k . Dann gilt

$$\varphi(n) := \prod_{i=1}^k (p_i - 1) \cdot p_i^{e_i - 1}.$$

Für Primzahlen p gilt $\varphi(p) = p - 1$

Für Primzahlpotenzen

p^e gilt

$$\begin{aligned}\varphi(p^e) &= p^e - p^{e-1} \\ &= (p-1)p^{e-1}\end{aligned}$$

$p, 2p, 3p, \dots, p^{e-1}p = p^e \subset p^{e-1}$ viele
sind Teiler von p^e

Die Eulersche φ -Funktion

Lemma. Sei $n = p_1^{e_1} \cdot \dots \cdot p_k^{e_k}$ für paarweise verschiedene Primzahlen p_1, \dots, p_k . Dann gilt $\varphi(n) := \prod_{i=1}^k (p_i - 1) \cdot p_i^{e_i - 1}$.

Die Eulersche φ -Funktion

Definition. Wir definieren

$$\mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \setminus \{0\} : \text{ggT}(a, n) = 1\}$$

und, für $n \geq 2$,

$$\varphi(n) := |\mathbb{Z}_n^*|.$$

$\varphi(n)$ ist also die Anzahl der zu n teilerfremden Zahlen $\leq n$.

Lemma. Sei $n = p_1^{e_1} \cdot \dots \cdot p_k^{e_k}$ für paarweise verschiedene Primzahlen p_1, \dots, p_k . Dann gilt

$$\varphi(n) := \prod_{i=1}^k (p_i - 1) \cdot p_i^{e_i - 1}.$$

在数学中, Z_n 表示模 n 意义下的整数集, 即所有非负整数从 0 到 $n - 1$ 组成的集合。通常情况下, Z_n 表示如下形式的集合:

$$Z_n = \{0, 1, 2, \dots, n - 1\}$$

这个集合中的元素在进行加法和乘法运算时, 结果都取模 n 。

在你提供的内容中, Z_n^* 表示的是在 Z_n 中与 n 互质的所有数的集合, 即:

$$Z_n^* = \{a \in Z_n \setminus \{0\} : \text{gcd}(a, n) = 1\}$$

其中, $\text{gcd}(a, n) = 1$ 表示 a 和 n 的最大公约数为 1, 也就是 a 和 n 是互质的。

Die Eulersche φ -Funktion

Definition. Wir definieren

$$\mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \setminus \{0\} : \text{ggT}(a, n) = 1\}$$

und, für $n \geq 2$,

$$\varphi(n) := |\mathbb{Z}_n^*|.$$

“互质”或“互素”。如果两个数是互质的，意味着它们的最大公约数是 1，也就是说，除了 1 以外，这两个数没有其他共同的因数。例如，8 和 15 是互质的，因为它们没有共同的因数，最大公约数是 1。

$\varphi(n)$ ist also die Anzahl der zu n teilerfremden Zahlen $\leq n$.

Lemma. Sei $n = p_1^{e_1} \cdot \dots \cdot p_k^{e_k}$ für paarweise verschiedene Primzahlen p_1, \dots, p_k . Dann gilt

$$\varphi(n) := \prod_{i=1}^k (p_i - 1) \cdot p_i^{e_i - 1}.$$

Satz (Euler). Für alle $n \in \mathbb{N}$ mit $n \geq 2$ gilt:

$$a^{\varphi(n)} \equiv 1 \pmod{n} \quad \text{für alle } a \in \mathbb{Z}_n^*.$$

Die Eulersche φ -Funktion

Satz (Euler). Für alle $n \in \mathbb{N}$ mit $n \geq 2$ gilt:

$$a^{\varphi(n)} \equiv 1 \pmod{n} \quad \text{für alle } a \in \mathbb{Z}_n^*.$$

Für Primzahlen gilt $\varphi(p) = p - 1$

Satz ("kleiner Satz von Fermat"). Für alle $n \in \mathbb{N}$ mit $n \geq 2$ gilt:

$$n \text{ ist eine Primzahl gdw. } a^{n-1} \equiv 1 \pmod{n} \text{ für alle } a \in \mathbb{Z}_n \setminus \{0\}.$$

Beweis der Hinrichtung.

Wenn n prim ist, so gilt $\varphi(n) = n - 1$ und $\mathbb{Z}_n^* = \mathbb{Z}_n \setminus \{0\}$.

Aus dem Satz von Euler folgt also die Hinrichtung des kleinen Satzes von Fermat.

10.6 Zurück zu RSA

Das RSA-Verfahren

Schlüsselgenerierung durch Empfänger:in.

- Wähle zwei Primzahlen p und q und berechne $n = p \cdot q$.
- Berechne $\varphi(n) = (p - 1) \cdot (q - 1)$.
- Berechne Zahlen k und ℓ mit
 $\text{ggT}(k, \varphi(n)) = 1$ und $k \cdot \ell \equiv 1 \pmod{\varphi(n)}$.

Öffentlicher Schlüssel. n, k

Geheimer Schlüssel. ℓ

Ver- und Entschlüsseln.

Nachricht: $M \in \{0, \dots, n - 1\}$

Verschlüsseln: $S := M^k \pmod{n}$

Gesendete Nachricht: S

Entschlüsseln: $M' := S^\ell \pmod{n}$

Fragen.

- Korrektheit.* Gilt immer $M' = M$.
- Sicherheit.* Kann man den Code knacken?
- Implementierung.* Wie aufwendig ist das Verfahren?

Korrektheit des RSA-Verfahrens

Korrektheit des Verfahrens. Sei $M \in \{0, \dots, n-1\}$ und $S := M^k \bmod n$ die verschlüsselte Nachricht.

Wir müssen zeigen, dass $M' := S^\ell \bmod n = M$, d.h. dass

$$M^{k \cdot \ell} \equiv M \pmod{n}.$$

Da $n = pq$ das Produkt zweier Primzahlen ist, reicht es wegen des Chinesischen Restsatzes zu zeigen, dass

$$M^{k \cdot \ell} \equiv M \pmod{p} \text{ und } M^{k \cdot \ell} \equiv M \pmod{q}.$$

Wir zeigen $M^{k \cdot \ell} \equiv M \pmod{p}$, der andere Fall ist analog.

Falls $p|M$ ist die Aussage offensichtlich wahr. $M^{k \cdot \ell} = 0 \equiv M \pmod{p}$

Falls $p \nmid M$ folgt aus dem Satz von Fermat $M^{p-1} \equiv 1 \pmod{p}$.

Nach Wahl von k, ℓ existiert $t \in \mathbb{Z}$ mit

$$k \cdot \ell = t \cdot \varphi(n) + 1 = t(p-1)(q-1) + 1.$$

Also gilt $M^{k \cdot \ell} \equiv (M^{(p-1)})^{t(q-1)} \cdot M^1 \equiv M \pmod{p}$.

Schlüssel. Wähle Primzahlen p, q .

Berechne $n = p \cdot q$.

Berechne $\varphi(n) = (p-1) \cdot (q-1)$.

Berechne k und ℓ mit

- $\text{ggT}(k, \varphi(n)) = 1$ und

- $k \cdot \ell \equiv 1 \pmod{\varphi(n)}$.

Ver- und Entschlüsseln.

Nachricht: $M \in \{0, \dots, n-1\}$

Verschlüsseln: $S := M^k \bmod n$

Entschlüsseln: $M' := S^\ell \bmod n$

$$M^{k \cdot \ell} = M^{t(p-1)(q-1)+1} \equiv M^{t(p-1)(q-1)+1} \equiv M^{t(q-1)} \cdot M^1 \equiv M \pmod{n}$$

□

Implementierung des RSA-Verfahrens

Erzeugung der Schlüssel.

Als erstes wählen wir zwei sehr große Primzahlen p, q .

Wie das geht, haben wir in Woche 9 besprochen.

Aus p und q lässt sich $n = pq$ und $\varphi(n)$ berechnen.

Nun rät man eine nicht zu kleine Zahl $k < n$ und testet mit dem erweiterten Euklidischen Algorithmus, ob k und n teilerfremd sind.

Wenn nein, rät man eine andere Zahl.

Ansonsten liefert der erw. Eukl. Alg. die Zahl ℓ .

Schlüssel. Wähle Primzahlen p, q .

Berechne $n = p \cdot q$.

Berechne $\varphi(n) = (p - 1) \cdot (q - 1)$.

Berechne k und ℓ mit

- $\text{ggT}(k, \varphi(n)) = 1$ und

- $k \cdot \ell \equiv 1 \pmod{\varphi(n)}$.

Ver- und Entschlüsseln.

Nachricht: $M \in \{0, \dots, n - 1\}$

Verschlüsseln: $S := M^k \pmod{n}$

Entschlüsseln: $M' := S^\ell \pmod{n}$

Implementierung des RSA-Verfahrens

Erzeugung der Schlüssel.

Als erstes wählen wir zwei sehr große Primzahlen p, q .

Wie das geht, haben wir in Woche 9 besprochen.

Aus p und q lässt sich $n = pq$ und $\varphi(n)$ berechnen.

Nun rät man eine nicht zu kleine Zahl $k < n$ und testet mit dem erweiterten Euklidischen Algorithmus, ob k und n teilerfremd sind.

Wenn nein, rät man eine andere Zahl.

Ansonsten liefert der erw. Eukl. Alg. die Zahl ℓ .

Verschlüsselung. Da die beim RSA-System verwendeten Zahlen sehr groß sind, sind die Rechenoperationen recht aufwendig.

Daher wird oft das RSA-System nur verwendet um zu Beginn der Kommunikation einen *symmetrischen Schlüssel* auszutauschen.

Die eigentliche Verschlüsselung der Nachricht erfolgt dann mit einem symmetrischen Verfahren.

Schlüssel. Wähle Primzahlen p, q .

Berechne $n = p \cdot q$.

Berechne $\varphi(n) = (p - 1) \cdot (q - 1)$.

Berechne k und ℓ mit

- $\text{ggT}(k, \varphi(n)) = 1$ und
- $k \cdot \ell \equiv 1 \pmod{\varphi(n)}$.

Ver- und Entschlüsseln.

Nachricht: $M \in \{0, \dots, n - 1\}$

Verschlüsseln: $S := M^k \pmod{n}$

Entschlüsseln: $M' := S^\ell \pmod{n}$