

1 Grundlagen

Allgemeines

$$x^k \bmod p \equiv (x \bmod p)^{k \bmod \varphi(p)} \bmod p$$
$$\gcd(a, m) = 1 \Rightarrow a^{\varphi(m)} \equiv 1 \bmod m$$

Berechnung der Stellenzahl

Die Anzahl a der Ziffern der b -adischen Darstellung einer nat rlichen Zahl $n \in \mathbb{N}_0$ berechnet sich wie folgt:

$$a = \begin{cases} 1, & \text{wenn } n = 0 \\ \lfloor \log_b n \rfloor + 1, & \text{wenn } n \geq 1 \end{cases}$$

Teilbarkeit

Zwei Zahlen $a, b \in \mathbb{Z}$ werden als teilerfremd bezeichnet, wenn $\gcd(a, b) = 1$ ist.

Ordnung

Die Ordnung eines Gruppenelementes g einer Gruppe (G, \cdot) ist die kleinste nat rliche Zahl $n > 0$ f r die gilt $g^n = e$, wobei e das neutrale Element der Gruppe ist.

Multiplikative Ordnung $\text{ord}_m(a)$ ist die multiplikative Ordnung modulo m des Elementes a , d.h. der kleinste positive Exponent n f r den gilt:

$$x^n \equiv 1 \pmod{m}$$

Eine Erweiterung dessen ist:

$$\text{ord}(x^l) = \frac{\text{ord}(x)}{\gcd(\text{ord}(x), l)}$$

Primzahlen Die Ordnung f r Primzahlen l sst sich wie folgt bestimmen:

$$\text{ord}(p) = \varphi(p) = p - 1$$

1.1 Multiplikative Inverse

Das Multiplikative Inverse von a im Modul m l sst sich mit dem erweiterten euklidischen Algorithmus berechnen. Der Algorithmus liefert die Linearkombination

$$\gcd(a, m) = u \cdot a + v \cdot m = 1$$

wenn a und m teilerfremd sind. Somit l sst sich das Inverse dann einfach ablesen:

$$a^{-1} \equiv u \bmod m$$

1.2 Eulersche Phi-Funktion

Die Eulersche Phi-Funktion gibt an, wie viele ganze Zahlen teilerfremd zu n sind. Wenn p eine Primzahl ist dann kann man folgende aussagen treffen:

$$\varphi(p) = p - 1$$
$$\varphi(p^k) = p^{k-1} \cdot (p - 1)$$
$$\varphi(n \cdot m) = \varphi(n) \cdot \varphi(m)$$
$$\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$$

Wobei $p|n$, die Primfaktoren der Zahl n sind.

1.3 Kontravalenz

\oplus	0	1
0	0	1
1	1	0

1.4 Diskreter Logarithmus

Der diskrete Logarithmus ist die kleinste L sung f r x der Gleichung $a^x \equiv m \bmod p$ mit $m, a \in \mathbb{N}, p \in \mathbb{Z}_p$. Da sich die diskrete Exponentiation leicht berechnen l sst, gilt das nicht f r die Umkehrfunktion. (*Diffie-Hellman-Annahme*) Aufgrund dessen wird diese Einwegfunktio-n . a. im Diffie-Hellman-Key-Exchange, der ElGamal-Encryption und vielem mehr eingesetzt. Jedoch ist diese Funktion ungeeignet f r Verschl sselungsmethoden, da es keine "Fall-t rskum entschl sselt gibt.

1.5 Modulares Potenzieren

Seien $x, k, m \in \mathbb{N}$, gesucht ist $z = x^k \bmod m$

- 1. Bin rdarstellung von k
- 2. Ersetzen jeder 0 durch **Q** und jeder 1 durch **QM**
- 3. Dabei wird **Q** als Anweisung zum *Quadrieren* und **M** als Anweisung zum *Multiplizieren* mit der Basis x aufgefasst.
- 4. Begonnen wird mit 1 bzw. kann die erste **QM** Anweisung durch x substituiert werden.

1.6 Chinesischer Restsatz

Seien $m_1, \dots, m_n \in \mathbb{N}$ paarweise teilerfremd, dann hat das System von Kongruenzen eine eindeutige L sung $x \in \mathbb{Z}_m$, wobei $m = m_1 \cdot \dots \cdot m_n$ das Produkt der einzelnen Module ist.

$$x \equiv a_1 \bmod m_1, \dots, x \equiv a_n \bmod m_n$$

Eine L sung x kann wie folgt ermittelt werden:

$$x = \left(\sum_{i=1}^n a_i \cdot M_i \cdot N_i \right) \bmod m$$

mit folgenden Vorraussetzungen:

$$m = m_1 \cdot \dots \cdot m_n$$
$$M_i = \frac{m}{m_i}$$
$$N_i = M_i^{-1} \bmod m_i$$

1.7 Euklidischer Algorithmus

Setze $r_0 := a, r_1 := b$

$$\begin{aligned} r_0 &= q_2 \cdot r_1 + r_2 \\ r_1 &= q_3 \cdot r_2 + r_3 \\ &\vdots \\ r_{n-2} &= q_n \cdot r_{n-1} + r_n \\ r_{n-1} &= q_{n+1} \cdot r_n + 0 \end{aligned}$$

Erweiterung

$x_0 = 1$	$x_1 = 0$	$y_0 = 0$	$y_1 = 1$
$x_2 = x_0 - q_2 \cdot x_1$		$y_2 = y_0 - q_2 \cdot y_1$	
$x_3 = x_1 - q_3 \cdot x_2$		$y_3 = y_1 - q_3 \cdot y_2$	
$x_n = x_{n-2} - q_n \cdot x_{n-1}$		$y_n = y_{n-2} - q_n \cdot y_{n-1}$	

dann gilt $x_n a + y_n b = \gcd(a, b)$.

1.8 Primitivwurzeln

Eine ganze Zahl a ist eine Primitivwurzel modulo m wenn gilt dass die Ordnung von a modulo m gleich der Gruppenordnung der primen Restklassengruppe ist:

$$\text{ord}_m(a) = \varphi(m)$$

Primitivwurzeltest Um festzustellen, ob eine Zahl g eine Primitivwurzel in der Restklassengruppe \mathbb{Z}_p^* mit p ist Primzahl ist, f hre man folgende Schritte aus:

- 1. Primfaktorzerlegung von $p - 1$:
 $p - 1 = p_1 \cdot \dots \cdot p_i$
- 2. Pr fe f r alle $q \in \{p_1, \dots, p_i\}$ ob gilt $g^{(p-1)/q} \not\equiv 1 \pmod{p}$
- 3. Sollten demnach alle Primfaktoren ungleich $1 \bmod p$ sein, dann ist g eine Primitivwurzel.

Falls g eine Primitivwurzel von \mathbb{Z}_p^* ist, dann ist auch $a = g^t$ eine Primitivwurzel von \mathbb{Z}_p^* genau dann wenn gilt: $\gcd(t, \varphi(p)) = 1$. Somit l sst sich folgendes aussagen:

$$\gcd(t, \varphi(p)) = 1 \Rightarrow \langle g^t \rangle = \mathbb{Z}_p^*$$

1.9 Miller-Rabin

2 Verschl sselungsalgorithmen

2.1 Asymetrische Verfahren

2.1.1 RSA

Schl sselerzeugung

- 1. W hle zwei gro e Primzahlen p, q mit $p \neq q$ und vorgegebener Bitl nge k .
- 2. Berechne $n = p \cdot q$.
- 3. Berechne $\varphi(n) = (p - 1)(q - 1)$
- 4. W hle $e \in \{3, \dots, \varphi(n) - 1\}$, wobei $\gcd(e, \varphi(n)) = 1$.
- 5. Berechne mit Hilfe des erweiterten Euklid das zu e multiplikativ-inverse Element d bez glich $\varphi(n)$:
 $\gcd(e, \varphi(n)) = 1 = e \cdot d + k \cdot \varphi(n)$
- 6. $(pk, sk) \leftarrow ((n, e), (n, d))$.

Verfahren

$$\text{ENC}(pk, m) = m^e \bmod n$$
$$\text{DEC}(sk, c) = c^d \bmod n$$
$$\text{SIG}(sk, m) = m^d \bmod n$$
$$\text{VER}(pk, m, \sigma) = 1 : \Leftrightarrow m = \sigma^e \bmod n$$

Verschl sselung mit RSA-OAEP Die RSA-OAEP (*optimal asymmetric encrypti-*

on padding) Variante ist gegen deterministische Angriffe sicher! Die Funktionen $g(x)$ und $h(x)$ sind Hashfunktionen.

Verschlüsselung

1. Wähle r zufällig

2. Berechne
 $x = m \oplus g(r)$ und $y = r \oplus h(x)$

3. Verschlüssele
 $\text{ENC}_{\text{OAEP}}(pk, m) = (x || y)^e \bmod n$

Entschlüsselung

1. Rekonstruiere $(x || y) = \text{DEC}(sk, c)$

2. Rekonstruiere r mit $r = y \oplus h(x)$

3. Berechne m mit $m = x \oplus g(r)$

Signieren und Entschlüsseln mit dem Chinesischen Restsatz

$$i = \begin{cases} c & \text{für Entschlüsseln} \\ m & \text{für Signieren} \end{cases}$$
$$o = \begin{cases} m & \text{für Entschlüsseln} \\ s & \text{für Signieren} \end{cases}$$

1. $d_p = d \bmod (p-1)$, $d_q = d \bmod (q-1)$

2. $i_1 = i^{d_p} \bmod p$, $i_2 = i^{d_q} \bmod q$

3. $h = \begin{cases} q^{-1} \cdot (i_1 - i_2) & \text{falls } i_1 > i_2 \\ q^{-1} \cdot (i_1 - i_2 + p) & \text{falls } i_1 < i_2 \end{cases}$

4. $h = h \bmod p$

5. $o = i_2 + (h \cdot q)$

2.1.2 ElGamal
Schlüsselerzeugung

1. Wähle eine große Primzahlen p mit vorgegebener Bitlänge k

2. Suche eine Primitivwurzel g in der Gruppe \mathbb{Z}_p

3. Wähle einen zufälligen Exponenten $x \in \{2, \dots, p-2\}$

4. Berechne $y = g^x \bmod p$

5. $(pk, sk) \leftarrow ((p, g, y), (p, g, x))$

Verfahren

$k \in \{1, \dots, p-2\}$ wird für jedes Verfahren erneut zufällig gewählt.

$$\text{ENC}(pk, m) = ((g^k), (y^k \cdot m)) \pmod p$$
$$\text{DEC}(sk, (g^k, c)) = (g^k)^{p-1-x} \cdot c \pmod p$$
$$\text{SIG}(sk, m) = (s_1, s_2)$$
$$\text{VER}(pk, m, \sigma) = 1 : \Leftrightarrow v_1 = v_2$$

Für SIG gilt zudem dass $\gcd(k, p-1) = 1$

$$s_1 := g^k \bmod p$$
$$s_2 := k^{-1} \cdot (m - a \cdot x) \bmod (p-1)$$

Für VER gilt zudem dass $1 \leq s_1 \leq p-1$

$$v_1 := g^m \bmod p$$
$$v_2 := y^{s_1} \cdot s_1^{s_2} \bmod p$$

2.2 Symmetrische Verfahren

2.2.1 DES

Struktur	Feistelchiffre
Schlüssellänge	56 Bit
Blocklänge	64 Bit
Rundenanzahl	16

2.2.2 AES

Struktur	Substitutionschiffre
Schlüssellänge	128, 192 oder 256 Bit
Blocklänge	128 Bit
Rundenanzahl	10, 12 oder 14

2.3 Blockverschlüsselung

Blockorientiert	Stromorientiert
ECB	CFB
CBC	OFB

Stromorientierte Betriebsarten erfordern kein Padding des Klartextes, und unterstützen keine asymmetrische Verschlüsselung. Um einen Klartext m verschlüsseln zu können muss dieser in Blöcke der Länge $r \leq n$ eingeteilt werden. Dabei ist n die Blocklänge des Verschlüsselungsverfahrens, r die Länge der Klartextblöcke. m_i bezeichnet dabei einen Block des Klartextes m .

n = Die Blocklänge des Verschlüsselungsverfahrens
 r = Die Blocklänge der Nachrichtenfragmente

2.3.1 ECB

$$r = n$$
$$c_i = \text{ENC}_k(m_i)$$
$$m_i = \text{DEC}_k(c_i)$$

Übertragungsfehler

Bitfehler in $c_i \Rightarrow m_i$ ist zufällig, alle anderen werden korrekt entschlüsselt.

Verlust von $c_i \Rightarrow m_i$ ist verloren, alle anderen werden korrekt entschlüsselt.

Angriff ist möglich da gleiche verschlüsselte Blöcke die selbe Nachricht enthalten.

2.3.2 CBC

$$r = n$$
$$c_0 = \text{Initialisierungsvektor}$$
$$c_i = \text{ENC}_k(m_i \oplus c_{i-1})$$
$$m_i = \text{DEC}_k(c_i) \oplus c_{i-1}$$

Übertragungsfehler

Bitfehler in $c_i \Rightarrow m_i$ ist zufällig und m_{i+1} hat den Bitfehler an gleicher Stelle wie c_i , alle anderen werden korrekt entschlüsselt.

Verlust von $c_i \Rightarrow m_i$ ist verloren und m_{m+1} ist zufällig, m_{i+2} und folgende werden korrekt entschlüsselt.

Angriff ist nicht möglich.

2.3.3 CFB

$$r \geq 1$$
$$n \geq r$$
$$x_0 = \text{Initialisierungsvektor}$$
$$x_{i+1} = \text{lsb}_{n-r}(x_i) || c_i$$
$$c_0 = x_0$$
$$c_i = m_i \oplus \text{msb}_r(\text{ENC}_k(x_i))$$
$$m_i = c_i \oplus \text{msb}_r(\text{DEC}_k(x_i))$$

Übertragungsfehler

Bitfehler in $c_i \Rightarrow m_i$ hat den Bitfehler an gleicher Stelle wie c_i , und alle folgenden $\lceil \frac{n}{r} \rceil$ Blöcke sind zufällig.

Verlust von $c_i \Rightarrow m_i$ ist verloren, und alle folgenden $\lceil \frac{n}{r} \rceil$ Blöcke sind zufällig.

Angriff ist nicht möglich.

2.3.4 OFB

$$r \geq 1$$
$$n \geq r$$
$$x_0 = \text{Initialisierungsvektor}$$
$$x_{i+1} = \text{ENC}_k(x_i)$$
$$c_0 = x_0$$
$$c_i = m_i \oplus \text{msb}_r(x_i)$$
$$m_i = c_i \oplus \text{msb}_r(x_i)$$

Übertragungsfehler

Bitfehler in $c_i \Rightarrow m_i$ hat den Bitfehler an gleicher Stelle wie c_i .

Verlust von $c_i \Rightarrow m_i$ ist verloren, und alle folgenden Blöcke sind zufällig (korrigierbar).

Angriff ist nicht möglich.

2.3.5 CTR

$$r \geq 1$$
$$n \geq r$$
$$\text{ctr}_i = \text{Initialisierungsvektor}$$
$$c_i = m_i \oplus \text{ENC}_k(\text{ctr}_i)$$
$$m_i = c_i \oplus \text{ENC}_k(\text{ctr}_i)$$

Übertragungsfehler

Bitfehler in $c_i \Rightarrow m_i$ hat den Bitfehler an gleicher Stelle wie c_i .

Verlust von $c_i \Rightarrow m_i$ ist verloren, und alle folgenden Blöcke sind korrekt.

Angriff ist nicht möglich.

Vorteile zum OFB: Wahlfreien Zugriff auf jeden verschlüsselten Block und ENC und DEC können parallel durchgeführt werden.

3 Angriffe gegen Verschlüsselungsalgorithmen

3.1 RSA
Common-Modulus-Attack

Möglich wenn die selbe Nachricht mit zwei unterschiedlichen Exponenten verschlüsselt wird, jedoch die Primfaktoren identisch sind. Die beiden Exponent sind dabei jedoch teilerfremd zu einander sind.

pk	c	m
(n, e_1)	$c_1 = m^{e_1} \bmod n$	m
(n, e_2)	$c_2 = m^{e_2} \bmod n$	m

- 1. Berechne mit dem erweitertem Euklid $a \cdot e_1 + b \cdot e_2 = 1$.
 a oder b wird negativ sein.
- 2. Berechne nun folgendes:

$$\begin{aligned} m &= m^1 \\ &= m^{a \cdot e_1 + b \cdot e_2} = (m^{e_1})^a \cdot (m^{e_2})^b \\ &= (c_1)^a \cdot (c_2)^b \bmod n \end{aligned}$$

- 3. Der negative Exponent kann nun auch wie folgt geschrieben werden:
 $x^{-2} = (x^{-1})^2$ wobei x^{-1} dem Inversen entspricht.
- 4. Sollte es kein Inverses geben (d.h. $\gcd(c_{1|2}, m) \neq 1$) dann lassen sich die Primfaktoren wie folgt berechnen:

$$n = \underbrace{\gcd(c_{1|2}, m)}_p \cdot \underbrace{\frac{n}{p}}_q$$

Low-Encryption-Exponent-Attack

Möglich bei einem kleinen Abstand zwischen den Primfaktoren von n .

- 1. $x = \lceil \sqrt{n} \rceil$
- 2. $y = \sqrt{x^2 - n}$
- 3. $\begin{cases} \text{gehe zu 2} & \text{wenn } y \text{ nicht ganzzahlig} \\ \text{gehe zu 4} & \text{wenn } y \text{ ganzzahlig} \end{cases}$
- 4. $p = x + y, q = x - y$

Small-Message-Space-Attack

Wenn die Anzahl der möglichen Nachrichten klein ist und der mögliche Inhalt im Vorraus

bekannt ist, dann kann der Angreifer eine Liste führen, in welcher die möglichen Nachrichten zusammen mit dem pk verschlüsselten Nachricht aufgeführt werden. Sollte dann eine Nachricht abgefangen werden, muss lediglich in der Liste gesucht werden. Durch den Einsatz von OAEP (Optimal Asymmetric Encryption Padding) kann dies verhindert werden.

3.2 ElGamal
Berechnungen

Ermitteln von k , wenn es mehrmals benutzt wurde um eine Nachricht zu signieren:

$$k = (s_{2_1} - s_{2_2})^{-1} \cdot (m_1 - m_2) \bmod (p - 1)$$

Bei bekannten k das x berechnen (Funktioniert nur, wenn das Inverse existiert):

$$x = s_1^{-1} \cdot (m - sk) \bmod (p - 1)$$

Existenzielles Fälschen

- 1. Wähle c zufällig mit $\gcd(c, p - 1) = 1$
- 2. Wähle b zufällig
- 3. $s_1 = g^b \cdot y^c$
- 4. $s_2 = -s_1 \cdot (c^{-1}) \bmod (p - 1)$
- 5. $m = -s_1 \cdot b \cdot (c^{-1}) \bmod (p - 1)$

Angriff mit Bleichenbacher

Falls eine Signatur (s_1, s_2) zur Nachricht m bekannt ist kann wie folgt eine Signatur gefälscht werden:

- 1. Berechne $m^{-1} \bmod (p - 1)$
- 2. Wähle eine zufällig Nachricht m'
- 3. $u = m' \cdot m^{-1} \bmod (p - 1)$
- 4. $v = s_1 \cdot u \bmod (p - 1)$
- 5. $s'_1 = (s_1 \cdot (p - 1)^2) + (v \cdot p) \bmod (p \cdot (p - 1))$ Chinesischer Restsatz
- 6. $s'_2 = s_2 \cdot u \bmod (p - 1)$

Das Ergebnis: (m', s'_1, s'_2) , dabei ist zu beachten das m' nicht $\bmod p$ genommen wird, d.h. es größer als p . Ohne den ersten Schritt in der Verifizierung ($1 \leq s_1 \leq p - 1$) wäre diese Signatur gültig.

4 Hashfunktionen

Hashfunktionen sind Funktionen die von einer großen, potentiell unbeschränkten Menge in eine kleinere Menge abbilden, also

$$h_k : \{0, 1\}^* \rightarrow \{0, 1\}^k$$

4.1 Sicherheitseigenschaften

4.1.1 Kollisionsresistenz

Die Kollisionsresistenz (*collision resistance*) bedeutet dass, es praktisch unmöglich ist zwei verschiedene Eingabewerte x und x' zu finden, die denselben Hashwert ergeben.

$$x \neq x', h(x) = h(x')$$

4.1.2 Einweigeigenschaft

Die Einweigeigenschaft (*pre-image resistance*) bedeutet dass, es praktisch unmöglich ist zu einem Hashwert y einen Eingabewert x zu finden, den die Hashfunktion auf y abbildet.

$$h(x) = y$$

4.1.3 Zwischenschritt

Einen Zwischenschritt zwischen Kollisionsresistenz und Einweigeigenschaft stellt die Ziel Kollisionsresistenz (*second preimage resistance*) dar, dass bedeutet das es praktisch unmöglich ist zu einem gegebenen Eingabewert x einen davon verschieden Eingabewert x' zu finden, der den selben Hashwert ergibt.

$$x \neq x', h(x) = h(x')$$

4.2 Modification Detection Code

Modification Detection Code (MDC) ist ein Hashwert, welcher der Integritätsprüfung dient.

4.3 Message Authentication Code

Message Authentication Codes (MAC) sind ein symmetrisches Verfahren, um die Authentizität einer Nachricht sicherzustellen. Hierzu gibt es einen Signatur- und einen Verifikationsalgorithmus welche beide eine gemeinsames Geheimnis benötigen.

4.3.1 Hash-MAC

Um einen HMAC σ zu prüfen, erstellt der Empfänger selbst einen HMAC und prüft, ob diese identisch ist zu σ .

$$opad = 0x5C \quad ipad = 0x36$$

$$\text{SIG}(k, m) = h((k \oplus opad) || h((k \oplus ipad) || m))$$

4.4 Gebrochene Algorithmen

Algorithmus	Gebrochene Eigenschaft
MD5	collision resistance
SHA-1	collision resistance

5 Protokolle

5.1 Diffie-Hellman-Schlüsselaustausch

Alice und Bob haben einen gemeinsamen öffentlichen Schlüssel (p, g) , wobei p eine Primzahl ist und g eine Primitivwurzel von \mathbb{Z}_p .

- 1. Alice wählt zufälliges $a \in [0; p - 2]$ und berechnet $c = g^a \bmod p$ und übermittelt c an Bob.
- 2. Bob wählt zufälliges $b \in [0; p - 2]$ und berechnet $d = g^b \bmod p$ und übermittelt d an Alice.
- 3. Alice berechnet nun $k = d^a \bmod p$
- 4. Bob berechnet nun $k = c^b \bmod p$

5.2 Station-to-Station-Protokoll

Eine Erweiterung des Diffie-Hellman-Schlüsselaustausch um einen Man-in-the-Middle-Angriff auszuschließen.

- 1. Alice wählt zufälliges $a \in [0; p - 2]$ und berechnet $c = g^a \bmod p$ und übermittelt c an Bob.
- 2. Bob wählt zufälliges $b \in [0; p - 2]$ und berechnet $k = g^{ab} \bmod p$
- 3. Bob sendet nun g^b sowie $z = \text{ENC}_k(s)$ mit $s = \text{SIG}_{sk_B}(g^a || g^b)$
- 4. Alice $k = g^{ab} \bmod p$ und $s = \text{DEC}_k(z)$ sowie $\text{VER}((g^a || g^b), s, pk_B)$
- 5. Alice sendet nun $z = \text{ENC}_k(s)$ mit $s = \text{SIG}_{sk_A}(g^b || g^a)$
- 6. Bob entschlüsselt $s = \text{DEC}_k(z)$ und verifiziert $\text{VER}((g^b || g^a), s, pk_A)$

Sollte die letzte Überprüfung korrekt sein, dann ist ein gemeinsamer Schlüssel gewählt.