CFB und OBC (Cipher/Output FeedBack)

- der CFB- bzw. OFB-Mous verschlüsselt auch kleinere Einheiten als die Blockgröße
- Prinzip: Schieberegister, in das nur so viele Bits geschrieben werden, wie auch verschlüsselt werden sollen (Ressourcenverschwendung) → z.B. bei Telnet (nur 8 statt 64 Bit)
- Vorteil: eine Blockchiffre kann wie eine Stromchiffre verwendet werden
- Beispiel: 56 Bits aus vorhergehendem Schritt, 8 werden nachgeschoben, dadurch fliegen 8 „alte“ Bits heraus

Asymmetrische Verfahren

- prinzipieller Nachteil von symmetrischen Verfahren → gesicherte Schlüsselübertragung
- bei symmetrischen Verfahren: ein Schlüssel zur Verschlüsselung, einer zur Entschlüsselung
- Prinzip von Public-Key-Verfahren:
  - Die Nachricht wird mit einem öffentlich bekannten Schlüssel verschlüsselt
  - Sie kann nur noch mit einem geheimen (privaten) Schlüssel wieder entschlüsselt werden
- Vorteil: Verfahren eignet sich für eine digitale Signatur (Authentifizierung)
- Problem: die Authentizität des öffentlichen Schlüssels
- In Software wird meistens kein rein asymmetrisches Verfahren benutzt (zu langsam)
- Basieren auf mathematischen Algorithmen, bei denen die Hinrechnung schnell ist, die Rückrechnung allerdings extrem aufwändig
- Werden auch als Falltür- bzw. Einwegverfahren bezeichnet
- Bsp. Multiplikation von Primzahlen:
  11*13=143, 221=17*13
  → kein Algorithmus verfügbar, welcher aus dem Produkt die Multiplikatoren herausrechnet
  → Nutzung von sehr langen Primzahlen erschwert diese Suche zusätzlich

RSA-Algorithmus

- RSA ist ein asymmetrisches Verfahren, das auch für digitale Signaturen verwendet wird
- Basiert auf einigen zahlentheoretischen Grundlagen (z.B. Satz von Euler) und Gesetzmäßigkeiten der Modulo-Rechnung
- Berechnung der Schlüssel::
  $e \cdot d = 1 \bmod \varphi(n)$ mit $n = p \cdot q \ (p, q \in P)$ (Primzahlen)

Satz von Euler

$a^{\varphi(n)} = 1 \bmod n \qquad (a, n \in \mathbb{N};\ ggT(a,n) = 1)$

$a^{\varphi(n)} = 1 \bmod n$ wäre auch durch 1 teilbar → es bleibt ein Rest von 1^

$\varphi(n)$ →Anzahl der zu n teilerfremden Zahlen

$n = 15$ → Teiler sind 3,5,6,9,10,12

➔ Anzahl der teilerfremden Zahlen $\varphi(15) = 9$, für die gilt ggT mit 15 und der Zahl =1

$\varphi(n)$ mit n als Primzahl → $\varphi(n) = n - 1$

$\varphi(n)$ mit $n = p \cdot q \ (p, q \in P)$ → $\varphi(n) = (p-1) \cdot (q-1)$

Berechnung der Schlüssel:

$e \cdot d = 1 \bmod \varphi(n)$

- aus zwei großen Primzahlen p,q lässt sich das phi(n) leicht berechnen → e kann man sich ausdenken → Berechnung der linearen Konvergenz (einfacher Algorithmus) → privater Schlüssel d lässt sich ermitteln

- Chiffrierung: $c = m^e \bmod n$
- Dechiffrierung: $m = c^d \bmod n$
- Zahlenbeispiel:

  $p = 7, q = 13 \rightarrow n = 13 \cdot 7 = 91 \rightarrow \varphi(n) = 6 \cdot 12 = 72$

  Länge des Klartextes $m < n$ → hier ist m im Maximalfall = 90

  Bestimmung der Schlüssel e und d

  $e \cdot d = 1 \bmod \varphi(n) \quad mit \quad ggT(e, \varphi(n)) = 1, z.B. e = 5$

  $5 \cdot d = 1 \bmod 72$ (→ Berechnung mit erweitertem euklidschen Algorithmus)

  z.B. d = 29 löst diese Gleichung

  Chiffrieren

  $m = 71 (ASCII \rightarrow G)$

  $c = m^e \bmod n = 71^5 \bmod 91 = 15$

  Dechiffrieren:

  $m = c^d \bmod n = 15^{29} \bmod 91 = 71$

  $15^{29} \bmod 91 = \left(\left(15^{14} \bmod 91 \right) \cdot \left(15^{15} \bmod 91 \right) \right) \bmod 91$

  $= \left(\left(15^7 \bmod 91 \cdot 15^7 \bmod 91 \right) \bmod 91 \ldots \right) \bmod 91$

  Schwäche → die Wahl von e

<u>Hybride Verschlüsselungsverfahren</u>

- Sender verschlüsselt den Klartext mit einem symmetrischen Verfahren (z.B. DES) und verschlüsselt dessen geheimen Schlüssel mit einem asynchronen Verfahren (z.B. RSA) → er nutzt dabei den öffentlichen Schlüssel des Empfängers
- Der Empfänger kehrt diesen Weg mit Hilfe des privaten Schlüssels um und macht dann die Schritte rückgängig

<u>Diffie-Helmann-Verfahren</u>

- Eine Alternative für hybride Verfahren
- Schlüssel-Austausch-Verfahren
- Es erfolgt keine Verschlüsselung der Daten
- Das Verfahren basiert auf eine Falltür- bzw. Einwegfunktion → Modulo-Logarithmus (diskreter Logarithmus)
- Beispiel:

  $a^x = b \bmod n$

  - Wenn a,x,n vorliegen, geht die Rechnung schnell
  - Wenn a,b,n vorliegen, ist die Rechnung sehr komplizert
- Ablauf:
  - Sender und Empfänger einigen sich auf eine Primzahl p (möglichst groß → Zuverlässigkeit des Verfahrens) und eine Zahl g < p; g & p können zwischen Sender und Empfänger unverschlüsselt ausgetauscht werden
  - Der Sender wählt eine geheim zu haltende Zahl x, der Empfänger tut Entsprechendes mit einer Zahl y
  - Der Sender berechnet eine Zahl $a = g^x \bmod p$ und schickt diese an den Empfänger

- Der Empfänger berechnet die Zahl $b = g^y \mod p$ und schickt das Ergebnis an den Sender
- Der Sender berechnet nun die Zahl $k = b^x \mod p$
- Der Empfänger berechnet nun die Zahl $k = a^y \mod p$
- Zum Beweis, dass es sich bei beiden Werten um denselben Wert für k handelt:

$$k = \left(g^x \mod p\right)^y \mod p = \left(g^y \mod p\right)^x \mod p = g^{xy} \mod p$$

<u>Hash-Funktionen</u>

- Erzeugen digitale Signaturen für ein Dokument
- Auch symmetrische Verfahren sind hierfür geeignet
- Nachteil: sehr langsam
- Mit Hash-Funktionen wird eine eindeutige Prüfsumme aus dem Dokument berechnet (und nur die wird verschlüsselt)

<u>Authentifizierung</u>

- Im wahren Leben Authentifizierung durch:
  - Etwas, das man ist (Aussehen, Fingerabdruck)
  - Etwas, das man weiß (Passwort, PIN, Codes)
  - Etwas, das man hat (Ausweis, Schlüssel)
- Integrität → Bsp. einer Umformung einer Mail durch Manipulation
- Online-Authentifizierung → biometrische Verfahren, Passwörter, TokenCards
- Probleme bei „Offline“-Authentifizierung:
  - Authentifizierung des öffentlichen Schlüssels
  - Authentifizierung des privaten Schlüssels
  - Verbindlichkeit
- Lösung: Vertrauensmodelle