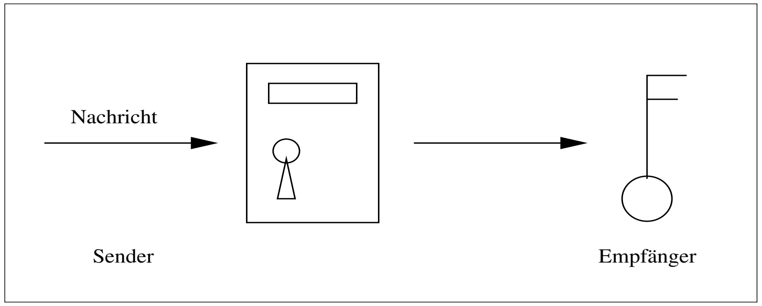
Wiederholung

- klassische Verschlüsselung: nur **ein** geheimer Schlüssel
  - man gibt mit der Verschlüsselung dem Gegenüber den Schlüssel preis, damit auch die Entschlüsselung.
  - Schlüsselaustausch ist wichtig
  - Protokolle basieren darauf, daß ich glaube, daß nur der/die Gegenpart/e den geheimen Schlüssel kennen
  - in der Regel sehr effiziente Verfahren (sowohl für Ver- als auch Entschlüsselung)
- Frage war: Muß man wirklich den geheimen Schlüssel verraten, um sicher zu kommunizieren?

Kryptologische Grundlagen 2: Asymmetrische Verschlüsselung

1976 Diffie & Hellmann: nur Empfänger braucht den geheimen Schlüssel.

Prinzip "Briefkasten"

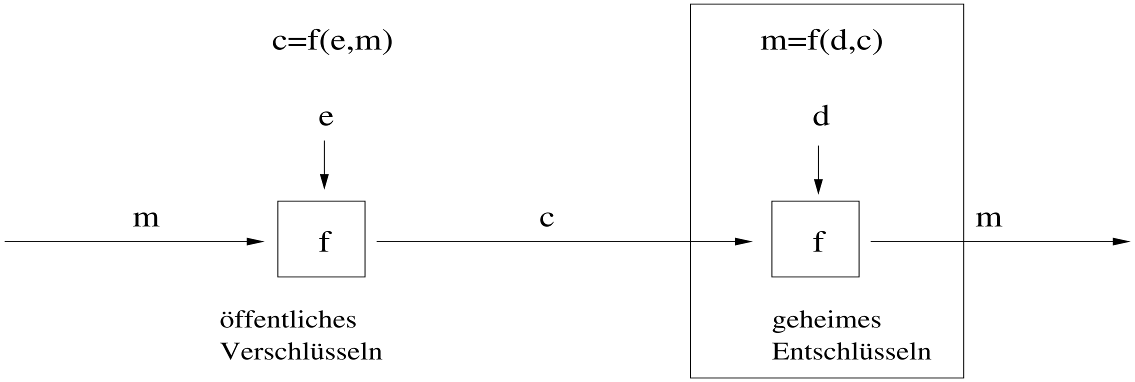


Nur der Empfänger hat den Schlüssel zum Briefkasten, um an die Nachricht zu gelangen.

$d \in$ private Schlüssel
$e \in$ öffentliche Schlüssel
$f(e,.)$: Klartext $\longrightarrow$ Geheimtext, Verschlüsselungsfunktion
$f(d,.)$: Geheimtext $\longrightarrow$ Klartext, Entschlüsselungsfunktion

Es muß gelten

- Korrektheit: Für $f(e,m) = c$ folgt $f(d,c) = f(d, f(e,m)) = m$.
- Public-Key-Eigenschaft: Es ist praktisch unmöglich von e auf d zu schließen.



Protokolle

Asymmetrisches Prinzip: Kann jemand korrekt eine Nachricht mit dem privaten Schlüssel der von ihm behaupteten Identität verschlüsseln, so gehe ich davon aus, daß dies seine Identität beweist.

k_P sei der öffentliche und k_P^{-1} der private Schlüssel von P.

P $\longrightarrow$ Q:	"Ich bin P"
Q:	erzeugt n (nonce)
Q $\longrightarrow$ P:	n
P:	berechne $m = f(k_P^{-1}, n)$
P $\longrightarrow$ Q:	m
Q:	verifiziere $f(k_P, m) = n$
	falls gleich, Verbindung akzeptieren, sonst ablehnen

Zertifikatstelle A: Datenbank öffentlicher Schlüssel

P $\longrightarrow$ Q:	"Ich bin P"
Q:	erzeugt n (nonce)
Q $\longrightarrow$ P:	n
P:	berechne $m = f(k_P^{-1}, n)$
P $\longrightarrow$ Q:	m
Q $\longrightarrow$ A:	"Ich brauche den öffentlichen Schlüssel von P"
A:	hole k_P aus der Datenbank
	erstelle Zertifikat $c = f(k_A^{-1}, k_P)$
A $\longrightarrow$ Q:	P, c
Q:	berechne $f(k_A, c) = k_P$
	verifiziere $f(k_P, m) = n$
	falls gleich, Verbindung akzeptieren, sonst ablehnen

Komplexität

- Basis asymmetrischer Verfahren sind “harte” Probleme und Trapdoors (Hintertüren)
  - Theoretisch sind die Probleme “berechenbar” (in endlich vielen Schritten)
  - echte Frage: Wieviel Schritte braucht man tatsächlich?
    Wie lange soll/muß etwas geheim bleiben?
- Zeit-Komplexität bei Algorithmen (Funktion in der Länge der Eingabe) abhängig vom Modell (meist Turingmaschinen in verschiedenen Varianten)
- wichtige Klassen:
  - P “Probleme mit polynomialen TM-Schranken”
  - NP “nicht-deterministische TM”
  - $P \subseteq NP$

"Harte" Probleme

- FACTOR(n): Faktorisiere n.
- PRIME(n): Teste ob n Primzahl oder nicht.
- FIND-PRIME($> n$): Finde eine Primzahl $> n$.
- DICRETE-LOG(a, b, n): Finde, falls möglich x mit $a^x \equiv b \bmod n$.
- KNAPSACK - Rucksackproblem

- Schritt 1: Wähle hartes Problem P.
  (P nach heutigem Wissen nicht polynomial und zwar nicht nur im worst case)
- Schritt 2: Wähle ein einfaches Unterproblem P_e von P
  (P_e mindestens polynomial, wenn möglich linear)
- Schritt 3: Verstecke P_e in einem Problem P_s, das P_e nicht ähnelt, jedoch dem Ursprungsproblem P.
- Schritt 4: Publiziere P_s als Verschlüsselungsverfahren.
  Wie man von P_s nach P_e kommt merkt man sich als Trapdoor (Geheimtür).
- Schritt 5: Sorge dafür, daß der legale Empfänger P_e zur Entschlüsselung benutzen kann, während andere es mit P versuchen müssen.

Beispiel Rucksackproblem

- Rucksackvektor: $A = (a_1, \ldots, a_n)$ mit ganzen Zahlen a_i.
- Instanz des Rucksackproblems: (A, α) mit Rucksackvektor A und ganzer Zahl α.
- Lösung von (A, α): Teilmenge von A, die sich zu α aufaddiert.
- Die Frage, ob eine Instanz eine Lösung hat ist NP-vollständig (d.h. liegt in NP und jedes Problem aus NP läßt sich polynomial darauf reduzieren).
- Verschlüsselung eines n-Bit Blockes $b_1 \ldots b_n$ erfolgt durch $\alpha = \sum_{i=1}^{n} b_i \cdot a_i$.
- Entschlüsselung bedeutet entweder den Block aus α herauszurechenen (symmetrisch) oder aus A und α (asymmetrisch).
- Trapdoor: man benutzt spezielle A, die "komplizierter" gemacht werden (z.B. mit $a_j > \sum_{i=1}^{j-1} a_i$; durch modulare Multiplikation verwischt $\tilde{a}_i = t \cdot a_i - [t \cdot a_i/m] \cdot m$, $m >$ max$\{a_1, \ldots, a_n\}$, $t < m$ als Trapdoorinformation)
- Problem: Es gibt Verfahren, um aus $\tilde{A}$ einen Vektor hinreichend nahe an A zu rekonstruieren.

RSA

Rivest **S**hamir **A**dleman Algorithmus (1978)

- Beruht auf dem Problem FACTOR(n)
- Es gibt keinen Beweis, daß FACTOR(n)$\notin$ P.
- Es gibt keinen Beweis, daß man faktorisieren **muß**, um RSA zu knacken.
- benutzt große Primzahlen (100 Dezimalstellen)
  es gibt ungefähr $x/\ln x$ Primzahlen kleiner x):

i	$\pi(10^i)$	i	$\pi(10^i)$
1	4	9	50.847.534
2	25	10	455.052.511
3	168	11	4.118.054.813
4	1.129	12	37.607.912.018
5	9.592	13	346.065.536.839
6	78.498	14	3.204.941.750.802
7	664.579	15	29.844.570.422.669
8	5.761.455	16	279.238.341.033.925

- Klartext als Zahl (dezimal oder binär)

Mathematische Grundlage: Sei $n = pq$ das Produkt zweier verschiedener Primzahlen. Dann gilt für jede natürliche Zahl $m \leq n$ und jede natürliche Zahl k die Gleichung

$$m^{k(p-1)(q-1)+1} \bmod n = m.$$

Man wählt zwei natürliche Zahlen e und d mit

$$ed = k(p-1)(q-1)+1$$

für eine natürliche Zahl k. Dann gilt

$$(m^e)^d \bmod n = m \text{ und } (m^d)^e \bmod n = m.$$

Konkretes Vorgehen:

1. Pro Teilnehmer zwei große Primzahlen p und q auswählen, $n = pq$
2. $\phi(n) = (p-1)(q-1)$ berechnen
3. e teilerfremd zu $\phi(n)$ wählen
4. d berechnen (erweiterter Euklidischer Algorithmus) mit ed mod $\phi(n) = 1$

Dann bilden e und n den öffentlichen Schlüssel und d den privaten Schlüssel des Teilnehmers. p, q und $\phi(n)$ können vernichtet werden.

- Verschlüsseln: $f(e, m) = m^e$ mod n.

- Entschlüsseln: $f(d, c) = c^d$ mod n.

- korrekt, da $f(d, f(e, m)) = (m^e)^d$ mod $n = m$.

- kennt man nur e, so kann man d aus n nicht berechnen (ohne Kenntnis der Faktoren oder von $\phi(n)$).

Beispiel für RSA

Sei

- $p = 47$, $q = 59$, $n = 2773$, $\phi(n) = 2668$, $e = 17$, $d = 157$.
- öffentlich also $n = 2773$ und $e = 17$.
- Verschlüsseln von SAUNA (Arto Salomaa, Public Key Cryptography)

Klartext	SA	UN	A
Zahlen	1901	2114	0100

ergibt $c_1 = 1901^{17} \bmod 2773$, $c_2 = 2114^{17} \bmod 2773$, $c_3 = 100^{17} \bmod 2773$.
Zum Beispiel erhält man $c_2 = 1644$ und $c_2^{157} \bmod 2773 = 2114$.

Anwendungen

- Gleiche Möglichkeiten wie bei symmetrischer Verschlüsselung, nur in der Regel langsamer. Daher häufig gemischte Verfahren, z.B. die Schlüssel für eine symmetrische Verschlüsselung mit einem asymmetrischen Verfahren austauschen.

- Digitale Unterschriften, z.B. Prüfsummen eines Dokumentes mit dem privaten Schlüssel verschlüsseln.

Kryptologische Grundlagen 3: Einwegfunktionen

Eine Einwegfunktion ist eine Funktion, die einfach auszuführen, aber schwer – praktisch unmöglich – umzukehren ist.

- Einwegpermutation := bijektive Einwegfunktion.
- kollisionsfrei := praktisch unmöglich, zwei verschiedene Werte x und x' in der Urbildmenge zu finden mit $f(x) = f(x')$.

Beispiele:

- herkömmliches Telefonbuch
  56707 56554 424200 8347 371137
  Telefonbuch 80, Jahrgang 1997/98
  Umkehrfunktion: Nach Nummern sortiertes Telefonbuch oder heute CD-Rom
- Passwortfile bei Unix

Es gilt: Einwegfunktionen existieren genau dann, wenn P $\neq$ NP.

Nach heutigem Wissenstand:

- Sei p eine Primzahl, g eine natürliche Zahl mit $g \leq p-1$. Dann ist die *diskrete Exponentialfunktion* zur Basis g definiert durch

$$k \longmapsto g^k \bmod p \qquad (1 \leq k \leq p-1).$$

- Quadrieren modulo $p \cdot q$, p, q Primzahlen.

- $n = pq$ und e, d wie gehabt für RSA.
  Potenzieren mit e modulo n ist eine Einwegfunktion.
  Umkehrfunktion: Potenzieren mit d.
  Weitere Trapdoorinformationen: $\phi(n)$ oder p, q. Sie ermöglichen das Berechnen von d.

- $F(.) = f(., m_0)$ wobei f symmetrisches Verschlüsselungsverfahren und m_0 feste Nachricht.

  Achtung: $G(.) = f(k_0, .)$ für festen Schlüssel k_0 ist **keine** Einwegfunktion. Warum?

  Man kann zu $y = G(x) = f(k_0, x)$ unter Benutzung des Schlüssels k_0 ein Urbild berechnen: $f^*(k_0, y) = f^*(f(k_0, x)) = x$.

- Spezialfall: Trapdoor-Einwegfunktionen Eine *Trapdoor-Einwegfunktion* ist eine Einwegfunktion, zu der es eine Geheiminformation (Geheimtür) gibt, mit deren Hilfe man die Funktion leicht umkehren kann.

Kryptologische Grundlagen 4: Hashfunktionen

Eine *Einweg-Hashfunktion* ist eine kollisionsfreie Einwegfunktion, die Nachrichten beliebiger Länge auf einen Hashwert einer festen Länge (typischerweise 128 Bit) komprimiert.

- unmanipulierbarer "Fingerabdruck" der Nachricht.
- Prüfsummen in der Regel ungeeignet, da sie für die manipulierte Nachricht neu berechnet werden können.
  Beispiel: Überweisung von DM $2580.-$ auf Konto 82677365 mit Prüfsumme Quersumme aller auftretenden Ziffern.
  Alternative Überweiseung auf das Konto 71234599 wäre für die Beträge DM $2980.-$ oder auch DM $982000.-$ möglich.

$$2+5+8+0+8+2+6+7+7+3+6+5=59$$

$$7+1+2+3+4+5+9+9+9+8+2+0+0+0=59$$

Kryptologische Grundlagen 5: Commitment und Bit-Commitment

Beim *Commitment* (Festlegung) geht es darum, daß eine Person A eine Nachricht so hinterlegt, daß sie

1. von niemandem gelesen werden kann und
2. von niemandem, insbesondere nicht von A, verändert werden kann.

Ein Commitment-Protokoll besteht aus 2 Phasen:

1. **Festlegen**: Partei A legt sich gegenüber B auf einen Datensatz m fest, indem sie $c = f(m)$ berechnet und c an B sendet. f ist eine kollisionsfreie Einwegfunktion.
2. **Öffnen**: A öffnet das Commitment, indem sie B das Urbild m von c mitteilt und B $c = f(m)$ überprüft.

Kryptologische Grundlagen 6: Digitale Signatur

Es geht darum, folgende Eigenschaften einer herkömmlichen Unterschrift in elektronischer Form zu realisieren:

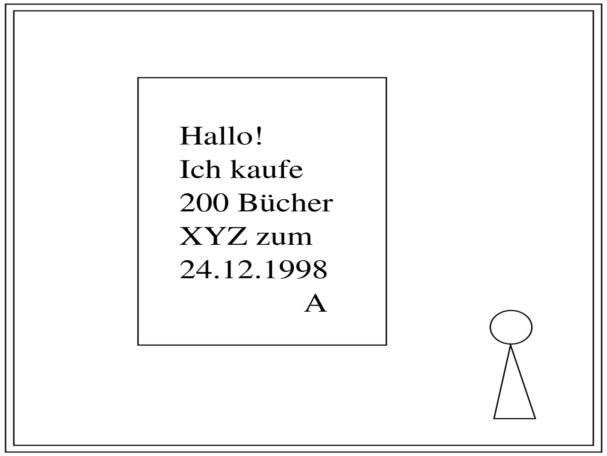
- Echtheitseigenschaft: Das Dokument stammt wirklich vom Unterzeichnenden, also Zusammenhang zwischen Dokument und Unterschrift. Wird herkömmlicherweise dadurch realisiert, daß die Unterschrift auf demselben Blatt steht.

- Identitätseigenschaft: Jede Unterschrift ist "persönlich".

- Abschlußeigenschaft: Dadurch, daß die Unterschrift am Ende steht, wird dir Vollendung des Unterschriebenen signalisiert.

- Verifikationseigenschaft: Unterschriftenvergleich

Prinzip “Gläserner Tresor”



Nur der Sender hat den Schlüssel zum gläsernen Tersor, um darin eine Nachricht (für andere sichtbar) zu deponieren.

Signaturschema: Jeder Teilnehmer T erhält eine Signaturfunktion s_T (geheim) und eine Verifikationsfunktion v_T (öffentlich). Es ist praktisch nicht möglich aus v_T die geheime Funktion s_T herzuleiten.

- Signatur der gesamten Nachricht:
  - T schickt m und $sig = s_T(m)$. v_t muß ermöglichen, aus m und sig auf T zu schließen.
  - Häufig ist v_T die Umkehrfunktion zu s_T, d.h. es gilt $v_T(sig) = m$. Dann überprüft man, ob m der geschickten Nachricht entspricht.

- Hashfunktion