Verschlüsselung — Krypthographie

Kryptologie := Kryptographie + Kryptonanlyse
Kryptographie := Entwicklung von Kodes
Kryptoanalyse := Brechen von Kodes

$\kappa\rho\upsilon\pi\tau o$ geheim, verborgen
$\gamma\rho\alpha\phi\eta$ schreiben

$$\text{Plaintext} \xrightarrow{\text{Encryption}} \text{Ciphertext} \xrightarrow{\text{Decryption}} \text{Plaintext}$$

$$\text{Klartext} \xrightarrow{\text{Verschlüsselung}} \text{Geheimtext} \xrightarrow{\text{Entschlüsselung}} \text{Klartext}$$

Kryptosystem

Ein Kryptosystem besteht aus 5 Teilen (nur eine grobe Defininition):

- Eine Menge von Klartexten K
- Eine Menge von Geheimtexten G
- Eine Menge von Schlüsseln S
- Einem Paar von Funktionen
  $f : S \times K \longrightarrow G$ (Verschlüsselung, Chiffrieren)
  $f^* : S \times G \longrightarrow K$ (Entschlüsselung, Dechiffrieren)
  so daß für jeden Schlüssel $k \in S$ und jeden Klartext $m \in K$ gilt:

$$f^*(k, f(k, m)) = m.$$

Ein einfaches Beispiel: Verschiebechiffren

Buchstaben werden mit Zahlen identifiziert und modulo 26 verschoben:
$a \to 1, b \to 2, \dots y \to 25, z \to 0.$

- Klartexte K: geordnete Mengen von Worten über $\{a, \dots z\}$
- Geheimtexte G: geordnete Mengen von Worten über $\{a, \dots z\}$
- Schlüssel S: Zahlen $1, \dots 26$
- Für $k \in \{1, \dots, 26\}$, $a \in \{a, \dots, z\}$:
  $f(k, a) = a + k$ mod 26
  $f^*(k, a) = a - k$ mod 26
  auf Worte und Texte fortgesetzt.
- Beispiel: $f(3, hallo) = kdoor$

Gehört zu den monoalphabetischen Chiffrierungen.

Monoalphabetische Chiffrierung

- Eindeutige Abbildung vom Klartextalphabet in das Geheimtextalphabet, so daß gleiche Buchstaben gleich kodiert werden.
- Stimmen die beiden Alphabete überein, so sind dies Permultationen, d.h. es gibt $26! \approx 4 \cdot 10^{26}$ Möglichkeiten.
- weitere Varianten: addieren, subtrahieren, multiplizieren und dividieren von Buchstaben und Schlüsselwortvarianten.
- für natürliche Sprachen sehr leicht zu knacken:
  - Ausprobieren (Rätsel in Zeitschriften)
  - statistische Analyse: Häufigkeitstabellen für Buchstaben.

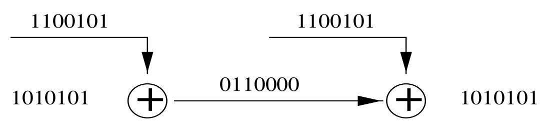
a	6,51	f	1,66	k	1,21	p	0,79	u	4,35
b	1,89	g	3,01	l	3,44	q	0.02	v	0,67
c	3,06	h	4,76	m	2,53	r	7	w	1,89
d	5,08	i	7,55	n	9,78	s	7,27	x,y	0,07
e	17,4	j	0,27	o	2,51	t	6,15	z	1,13

- Alternative: polyalphabetische Chiffrierungen

Ein zweites Beispiel: One-time Pad

Stromchiffren — Zeichenweise verschlüsseln.

- Klartexte K: n-Bit Folgen
- Geheimtexte G: n-Bit Folgen
- Schlüssel S: n-Bit Folgen
- Für $k \in S$, $m \in K$, $m' \in G$:
  $f(k, m) = k \oplus m$
  $f^*(k, m') = k \oplus m'$



- Vorteile:
  - Einfach zu berechnen
  - Gleicher Klartext ergibt in der Regel nicht gleichen Geheimtext!
  - So sicher wie möglich!
    Ist das One-time Pad zufällig, so sind für einen Geheimtext alle möglichen Klartexte gleich wahrscheinlich.
- Nachteile:
  - Der Schlüssel muß so lang wie der Text sein.
  - Der Schlüssel darf nur einmal benutzt werden.
  - Der Schlüssel muß ausgetauscht werden (in der Praxis häufig durch Pseudozufallsgenerator erzeugt, der inilialisiert wird).

Neues Ziel: Man konstruiere ein Verfahren, mit dem man die vertrauliche Übertragung beliebig vieler, beliebig langer Nachrichten auf die einmalige geheime Übermittlung eines kurzen Datensatzes (des Schlüssels) zurückführen kann.

Verschlüsselung allgemein

Grundidee: If lots of smart people have failed to solve a problem, then it probably won't be solved (soon).

Bausteine:

Algorithmus	+	Schlüssel	=	Kryptosystem
$\uparrow$		$\uparrow$		
bekannt		geheim		

- Algorithmus muß effizient sein.
- Schlüssel muß hart zu brechen sein.

Zahlenschloß

- Mechanismus ist bekannt $\longrightarrow$ Algorithmus.
- Zahlenkombination ermöglicht das Öffnen $\longrightarrow$ Schlüssel.

24	10	36

Geht man von $n \in \{1, \ldots, 40\}$ aus, so hat man $40^3 = 64000$ Möglichkeite. Setzt man 10 sec pro Kombination an, so braucht man etwa eine Woche. Bei 4 Zahlen und 13 sec pro Kombination bereits mehr als ein Jahr!

Argument: Computer werden immer schneller:

- 1 Bit mehr verdoppelt die Anzahl der Schlüssel.
- Ist der neue Computer 1000 mal schneller, so holen 10 Bit längere Schlüssel dies wieder auf. Zusätzlich wird die Verschlüsselung selbst schneller sein.

Angriffe auf Verschlüsselungen

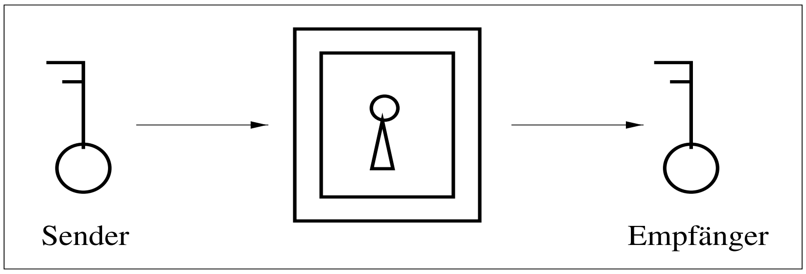
1. *Ciphertext Only*
   - Kennt nur Geheimtext und versucht ihn durch Raten des Schlüssels zu entschlüsseln.
   - Braucht "genug" und "gute" Beispiele.
     XYZ kann alles mögliche sein:
     The new cat was sad but you may now sit and use her big red pen.
   - Schlüssel sind einfacher zu erraten, wenn sie aus Passwörtern gebildet werden, wie dies bei manchen Systemen der Fall ist. (Kerberos erzeugt aus einem vom Benutzer eingegebenen Passwort nach einem publizierten Verfahren den DES-Schlüssel)
   - Da der Geheimtext der bekannte Text ist, muß ein Kryptosystem gegen diese Attacke sicher sein.

2. *Known Plaintext*
   - Kennt Paare (Klartext, Geheimtext).
   - Bei monoalphabetischen Kodes bereits kritisch.
3. *Chosen Plaintext*
   - Kann selbst verschlüsseln oder verschlüsseln lassen.
   - Anfällig für geschicktes Raten: The quick brown fox jumps over the lazy dog.
     "Fight" und "Surrender"
4. *Chosen Ciphertext*
   - Kann ausgewählten Geheimtext entschlüsseln.

Verschlüsselung gilt als *sicher*, wenn sie alle diese 4 Attacken übersteht.

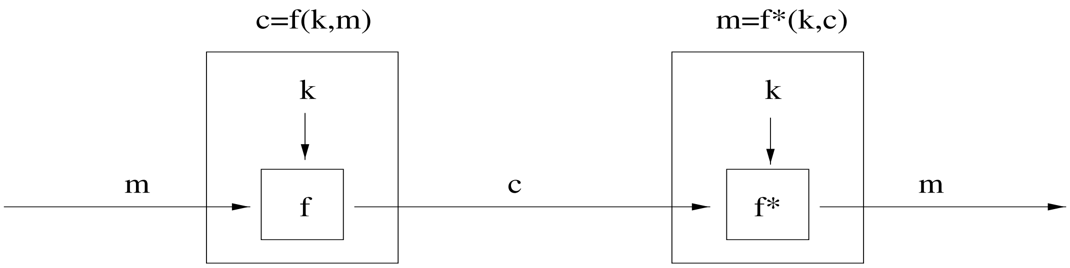
Kryptologische Grundlagen 1: Symmetrische Verschlüsselung

Prinzip “Tresor”



Beide haben den gleichen Schlüssel zu einem Tresor, in den die Nachricht gelegt wird.

$k \in$ Schlüssel
$m \in$ Klartext
$c \in$ Geheimtext
f: Schlüssel $\times$ Klartext $\longrightarrow$ Geheimtext (Verschlüsselung)
f^*: Schlüssel $\times$ Geheimtext $\longrightarrow$ Klartext (Entschlüsselung)



Kerckhoffsches Prinizip: Die Sicherheit eines Kryptosystems darf nicht von der Geheimhaltung des Algorithmus abhängen. Die Sicherheit gründet sich nur auf die Gehimhaltung des Schlüssels.

Schlüsselaustausch

- Verteilen des Schlüssels an die beteiligten Partien (mit entsprechendem Sicherheitsrisiko)
- Geht man von passiven Gegnern aus: Exponentieller Schlüsseltausch nach Diffie und Hellman.
  A und B wollen gesichert kommunizieren.
  1. (öffenticher) Austausch von einer großen Primzahl p und g (Erzeuger der multiplikativen Gruppe $\mathbf{Z}_p^*$)
  2. A und B wählen geheime große Zahlen a und b
  3. A schickt B den Wert $g^a \bmod p$
  4. B schickt A den Wert $g^b \bmod p$
  5. beide berechnen $g^{ab} \bmod p$ als gemeinsamen geheimen Schlüssel

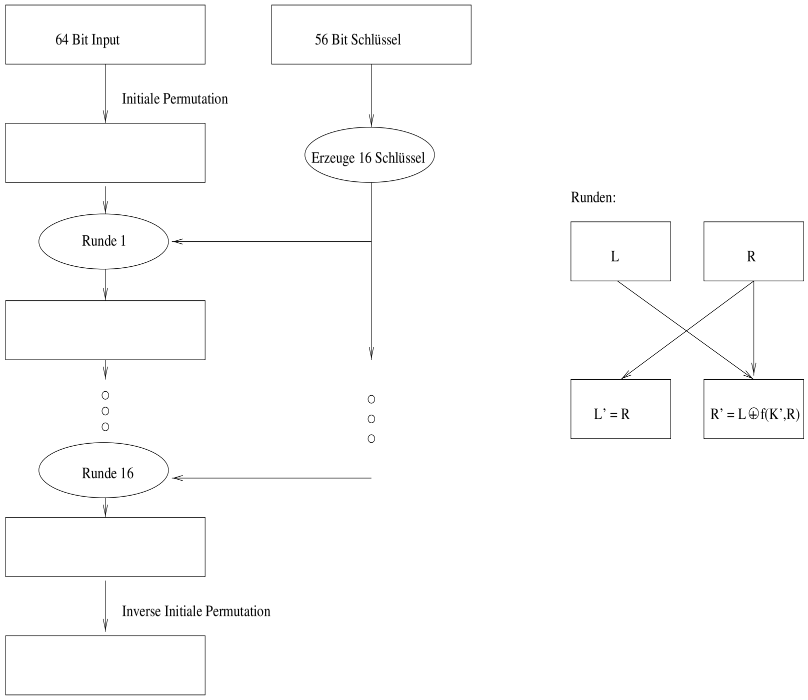
  Bei diesem Verfahren wird der geheime Schlüssel **nicht** ausgetauscht. Er müsste von einem Lauscher aus $g^a \bmod p$ und $g^b \bmod p$ berechnet werden.

Stromchiffren: siehe One-time Pad.

Blockchiffren: Nachricht in Blöcken $m_1, m_2, \ldots$.

- Jeder Block wird unter Verwendung des Schlüssels k einzeln verschlüsselt $c_i = f(k, m_i)$.
- **D**ata **E**ncryption **S**tandard:
  Blocklänge 64 Bit, Schlüssel 56 Bit.
- **I**nternational **D**ata **E**ncryption **S**tandard:
  Blocklänge 64 Bit, Schlüssel 128 Bit.
- Man unterscheidet verschiedene Arten, die Blöcke der Nachricht zu verschlüsseln:
  - **E**lectronic **C**ode **B**ook: Jeder Block wird einzeln mit dem geheimen Schlüssel verschlüsselt.
  - **C**ipher **B**lock **C**haining: Jeder Block wird vor der Verschlüsselung mit dem verschlüsselten Vorgängerblock mit x-or verrechnet, der erste Block mit einer Zufallszahl.

Beispiel DES



monoalphabetische Chiffrierung von 64-Bit Wörtern über $\{0, 1\}$.

- 1977 vom National Bureau of Standards veröffentlicht für kommerzielle Anwendungen und unklassifizierte U.S. Regierungsanwendungen.
- Basiert auf einem Algorithmus "Lucifer cipher" von IBM.
- Ursprünglich effizient, wenn in Hardware implementiert und nur langsam, wenn in Software implementiert. Inzwischen nicht mehr so extrem.
- Warum 56 Bit? Streitfrage zwischen **N**ational **S**ecurity **A**gency und Wissenschaftlern.
  Vermutungen: NSA will Kontrolle behalten und alles zur Not selbst entschlüsseln können.
- Prognosen für die mögliche Entschlüsselung, d.h. dafür, daß man aus einer verschlüselten Nachricht den Schlüssel rekonstruieren kann:
  – 1977, Diffie und Hellman: Für 20 Millionen Dollar könnte man einen DES-Cracker bauen, der in 12 Stunden für eine (Klartext, Geheimtext)-Paar den Schlüssel finden kann. (Suche im Gesamten Schlüsselraum)

- “The reality is that, at least in 1995, DES is perfectly adequate for most applications.” (Kaufman, Network Security)
- 26.6.1997: Freech, Direktor des FBI: “And we do not have the computers, we do not have the technology to get either real-time access to that information or any kind of timely access.”
- NSA: “A group of students – not students – the Internet gang last week broke a single message using 56-bit DES. It took 78000 computers 96 days to break one message.”
  64-Bit — 7000 years
  128 Bit — 8.6 trillionen times the age of the universe (also ewig)
- 1997: etwa gleichzeitig entschlüsselt ein Student von INRIA die gleiche Nachricht mit dem lokalen Intranet.
- Dennoch wird darauf beharrt, daß DES sicher ist: es sei ein reines Ressourcenproblem.

- Problem: Alle diese Aussagen gehen davon aus, daß man herkömmliche Computer zum Cracken benutzt.
- 1998: John Gilmore und Paul Kocher gelingt es mit Unterstützung der **E**lectronic **F**rontier **F**oundation “Deep Crack” zu bauen. Dies ist ein selbstgebastelter Supercomputer aus 100 speziellen Chips und 27 Boards – Wert 210000 Dollar (80000 Arbeit, 130000 Material). Entwurf zum Nachbauen wurde veröffentlicht und ist im Netz erhältlich (lesenswert).

• Was nun? Drei-Schlüssel Dreifach-DES (jeden Block zuerst mit k_1 verschlüsseln, dann mit k_2 entschlüsseln und mit k_3 wieder verschlüsseln).

Protokolle

Symmetrisches Prinzip: Kann jemand korrekt mit einem geheimen Schlüssel, von dem ich annehme, daß nur er und ich ihn haben, Nachrichten verschlüsseln, so gehe ich davon aus, daß dies seine Identität beweist.

k geheimer Schlüssel, den P und Q kennen.

P:	erzeuge $m =$"ich bin P"
	berechne $m' = f(k, m)$
P $\longrightarrow$ Q:	m,m'
Q:	verifiziere $f(k, m) = m'$
	falls gleich, Verbindung akzeptieren, sonst ablehnen

Einweg-Authentifikation.

Problem: Kann mit Replay (Nachricht abhören und speichern) ausgetrickst werden.

Verhinderung von Replays durch "nonces" (number once), eine Art Zeitstempel

P $\longrightarrow$ Q:	"Ich bin P"
Q:	erzeugt n (nonce)
Q $\longrightarrow$ P:	n
P:	berechne $m = f(k, n)$
P $\longrightarrow$ Q:	m
Q:	verifiziere $f(k, n) = m$
	falls gleich, Verbindung akzeptieren, sonst ablehnen

Challenge – Response

Problem: Nicht praktikabel für viele Teilnehmer. Entweder muß man mit jedem Teilnehmer einen eigenen geheimen Schlüssel ausmachen (sehr viele Schlüssel verwalten) oder alle haben denselben Schlüssel und bereits ein Verlust kann das ganze System unterlaufen.

Authenifikationsserver A, der mit jedem Teilnehmer X einen geheimen Schlüssel k_{XA} teilt.

P $\longrightarrow$ Q:	"Ich bin P"
Q:	erzeugt n (nonce)
Q $\longrightarrow$ P:	n
P:	berechne $x = f(k_{PA}, n)$
P $\longrightarrow$ Q:	x
Q:	berechne $y = f(k_{QA}, x)$
Q $\longrightarrow$ A:	y
A:	berechne $f^*(k_{QA}, y) = x$
	berechne $f^*(k_{PA}, x) = n$
	berechne $f(k_{QA}, n) = m$
A $\longrightarrow$ Q:	m
Q:	verifiziere $f(k_{QA}, n) = m$
	falls gleich, Verbindung akzeptieren, sonst ablehnen