E-Mails und Daten verschlüsseln
Eine Einführung in GnuPG

Jens Kubieziel
mailto:jens@kubieziel.de

13. März 2004

– Typeset by FoilTEX –

Warum soll ich Verschlüsselung benutzen?

Welche Vorteile hat Verschlüsselung bzw. Nutzung kryptographischer Software?

Geheimhaltung Nachricht soll nicht durch Dritte entziffert werden können

Echtheit des Kommunikationspartners Stammt die Nachricht wirklich von dem, der er vorgibt zu sein?

Echtheit der Nachricht Ist die Nachricht so verfasst worden, wie sie angekommen ist?

Vertrauen in die Kommunikation Erreicht, falls obige Punkte eingehalten werden.

Warum soll ich Verschlüsselung benutzen?

Moderne Kommunikationsformen (E-Mail, Handy) sind sehr wenig geschützt und können leicht abgehört und z.T. auch geändert werden.

E-Mails abhören

- E-Mail-Verkehr kann schon mit einfachen Tools wie Ethereal belauscht werden
- „komfortable“ Programme existieren und legen E-Mails als Mailbox ab
- Jeder mit Zugriff auf das Netz kann mitschneiden und ändern
- Prominentes Beispiel: ECHELON

PS.: Abhören von Handys ist fast ebenso einfach. Der „Verschlüsselungs“-Algorithmus wurde letztes Jahr dreimal gebrochen.

Die Lösung?

Die Lösung ist:

Kryptographie

= Wissenschaft von der Verschlüsselung und Verschleierung

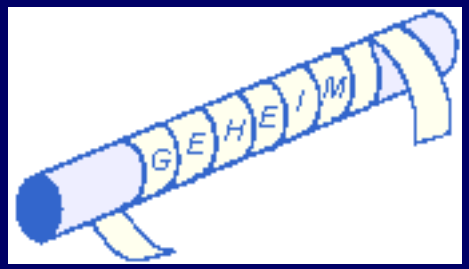
Im Gegensatz dazu:
Kryptoanalyse = Wissenschaft von der Entschlüsselung

Wie ist Kryptographie entstanden?

- Ursprünge liegen sehr weit zurück
- „herrschaftliches“ Wissen sollte vor dem Volk versteckt werden
- zunächst wurden Daten wirklich versteckt
- später ging man zur Verschlüsselung der Texte über

Skytala von Sparta

- älteste Verschlüsselungsmethode (ca. 2500 Jahre)
- von Plutarch überliefert
- Band wird um einen Zylinder gewickelt, beschriftet und abgerollt
- *Transpositionschiffre*



Beispiel Skytala

Geheimtext: VPUUIAMXESLTLSIASZNG

Test mit Durchmesser 3cm: VUMSLANPIXLSSGUAETIZ

Test mit Durchmesser 4cm: VIELSPASSZUMLINUXTAG

Cäsarchiffre

- von C. Julius Cäsar (100-44 v.Chr.) erfunden
- Buchstaben werden um einige Stellen verschoben
- *Verschiebechiffren*
- ROT13 als „moderne“ Variante

Beispiel Cäsarchiffre

Klartextalphabet:
a b c d e f g h i j k l m n o p q r s t u v w x y z

Geheimtextalphabet:
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Klartext: Betriebssystem

Geheimtext: EHWULHEVVBVWHP

Problem: Buchstabenhäufigkeiten bleiben erhalten
Lösungsversuch: polyalphabetische Chiffren

Vigenère-Verschlüsselung

- von Blaise de Vigenère (1523 - 1596) um 1586 entwickelt
- verschiedene monoalphabetische Chiffrierungen werden im Wechsel benutzt
- Prototyp für viele Algorithmen der heutigen Tage
- ca. 300 Jahre später fand Kasiski Angriff gegen die Verschlüsselung

Vigenère-Quadrat

ABCDEFGHIJKLMNOPQRSTUVWXYZ	ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ	NOPQRSTUVWXYZABCDEFGHIJKLM
BCDEFGHIJKLMNOPQRSTUVWXYZA	OPQRSTUVWXYZABCDEFGHIJKLMN
CDEFGHIJKLMNOPQRSTUVWXYZAB	PQRSTUVWXYZABCDEFGHIJKLMNO
DEFGHIJKLMNOPQRSTUVWXYZABC	QRSTUVWXYZABCDEFGHIJKLMNOP
EFGHIJKLMNOPQRSTUVWXYZABCD	RSTUVWXYZABCDEFGHIJKLMNOPQ
FGHIJKLMNOPQRSTUVWXYZABCDE	STUVWXYZABCDEFGHIJKLMNOPQR
GHIJKLMNOPQRSTUVWXYZABCDEF	TUVWXYZABCDEFGHIJKLMNOPQRS
HIJKLMNOPQRSTUVWXYZABCDEFG	UVWXYZABCDEFGHIJKLMNOPQRST
IJKLMNOPQRSTUVWXYZABCDEFGH	VWXYZABCDEFGHIJKLMNOPQRSTU
JKLMNOPQRSTUVWXYZABCDEFGHI	WXYZABCDEFGHIJKLMNOPQRSTUV
KLMNOPQRSTUVWXYZABCDEFGHIJ	XYZABCDEFGHIJKLMNOPQRSTUVW
LMNOPQRSTUVWXYZABCDEFGHIJK	YZABCDEFGHIJKLMNOPQRSTUVWX
MNOPQRSTUVWXYZABCDEFGHIJKL	ZABCDEFGHIJKLMNOPQRSTUVWXY

Beispiele Vigenère

Klartext	CHEMNITZ
Schlüssel	linuxlin
Geheimtext	NMYGKTBM

Klartext	LEERERTEEKESSEL
Schlüssel	abcabcabcabcabc
Geheimtext	LFGRFTTFGKFUSFN

One-Time-Pad

- mathematisch beweisbar sichere Verschlüsselungsmethode
- Voraussetzungen:
  - Schlüssel gleichlang wie Klartext
  - Schlüsselerzeugung streng zufällig
  - Schlüssel wird nur einmal verwendet
- wegen des hohen Aufwands meist in militärischen Umgebungen angewendet
- Problem: menschlicher Faktor

Enigma

- erste Version im Jahr 1904
- wurde von der Kriegsmarine eingesetzt
- Hardwareverschlüsselung
- wurde von polnischen und englischen Wissenschaftlern gebrochen
- Bau von Bomben, die systematisch alle Schlüssel probieren

Symmetrische vs. Asymetrische Verschlüsselung

Nachteil aller bisherigen Methoden:

Zur Kommunikation über unsichere Kanäle (Internet) muss der Schlüssel zuerst über einen sicheren Kanal übertragen werden.

Bei der Kommunikation mit vielen Partnern benötigt man sehr viele Schlüssel.

$=>$ asymetrische Verschlüsselung

RSA

- Prinzip von Whitfield Diffie und Martin Hellman 1976 entwickelt
- Ron Rivest, Adi Shamir und Leonard Adleman wollten das Gegenteil beweisen ...
- ... und erfanden RSA
- ältester und angesehenster Algorithmus

Prinzip von RSA

1. Wahl zweier grosser Primzahlen a und b

2. Ermittlung des Produkts $n = a \cdot b$ und Ermittlung der Anzahl der zu n teilerfremden Zahlen $\varphi(n) = (a-1)(b-1)$

3. Suche einer Zahl d, für die gilt: $e \cdot d \mod \varphi(n) = 1$

4. e und n sind der öffentliche Schlüssel, d ist der private Schlüssel

DES und 3DES

- Data Encryption Standard von 1977
- Blockchiffre mit 56 Bit
- im kommerziellen Bereich am häufigsten eingesetzt
- u.a. 1999 in ca. 22 Stunden gebrochen
- Weiterentwicklung: 3DES

Verschlüsselung mit GnuPG

- PGP (Pretty Good Privacy) 1991 von Phil Zimmermann, zunächst Open Source, später (1997) Verkauf an NAI
- div. Inkombatibilitäten der PGP-Versionen führten 1998 zu RfC 2440
- 1998 Entwicklung von GnuPG, Förderung von BMBF
- aktuelle Version 1.2.4

Schlüssel erzeugen - Wahl des Signatur- und/oder Verschlüsselungsalgorithmus

```
gpg --gen-key
Please select what kind of key you want:
   (1) DSA and ElGamal (default)
   (2) DSA (sign only)
   (4) RSA (sign only)
Your selection?
```

DSA Digital Signature Algorithm, zum Signieren von Dateien

ElGamal Verschlüsselungsalgorithmus von El Gamal (Israel), ähnlich wie RSA

RSA Rivest, Shamir, Adleman, wie bereits vorgestellt

Schlüssel erzeugen - Wahl der Schlüssellänge

```
DSA keypair will have 1024 bits.
About to generate a new ELG-E keypair.
                minimum keysize is  768 bits
                default keysize is 1024 bits
                highest suggested keysize is 2048 bits
What keysize do you want? (1024)
```

Es gibt Anzeichen, dass 1024 nicht mehr ausreichend sicher sind. Deshalb sollte eine Schlüssellänge von mind. 2048 bit gewählt werden.

Schlüssel erzeugen - Wahl der Gültigkeitsdauer

```
Please specify how long the key should be valid.
         0 = key does not expire
       <n>  = key expires in n days
       <n>w = key expires in n weeks
       <n>m = key expires in n months
       <n>y = key expires in n years
Key is valid for? (0)
```

Gültigkeitsdauer hängt von eigenen Präferenzen ab. Oftmals wird ein Hauptschlüssel ohne feste Gültigkeit generiert und dann jeweils Unterschlüssel mit einjähriger Laufzeit.

Schlüssel erzeugen - Eingabe des Namens, E-Mail-Adresse und Passphrase

- Eingabe von Namen und E-Mail-Adresse und Bestätigung der Angaben
- Eingabe der Passphrase

  Passphrase kein „Wort“ aus dem Wörterbuch, möglichst ein verfremdeter Satz, z.B. „Ich habe den Linuxtag in Chemnitz besucht.“ –> „1(h 4abe d3n L!nv)(tag /n Ch0mn:tz b?sucht+“

Schlüssel

```
pub   1024D/AE4B83EB 2004-03-02 Jens Kubieziel <jens@example.org>
      Key fingerprint = D75C 3827 32DA 1536 22C1\
                                538B 54BA 985E AE4B 83EB
sub   1024g/E4941357 2004-03-02
```

Anzeige des Fingerprints am Ende der Schlüsselerzeugung

Widerrufszertifikat erzeugen

- Vorbeugung vor kompromittierten Schlüsseln und vergessenen Passwörtern
- Widerrufszertifikat ausdrucken und an sicherem Ort aufbewahren

```
gpg --gen-revoke AE4B83EB

sec  1024D/AE4B83EB 2004-03-12  Jens Kubieziel <jens@example.org>

Create a revocation certificate for this key? y
Please select the reason for the revocation:
  0 = No reason specified
  1 = Key has been compromised
  [...]
```

Aufbau des Web of Trust

Problem: Jeder kann Schlüssel für beliebige Namen erzeugen. Viele falsche Schlüssel existieren.

Lösung: Keysigning
Inhaber von PGP-Schlüsseln treffen sich und verifizieren ihre Identitäten.