5 Kryptographische Maßnahmen (2)

- Funktionen
  - Verschlüsselungsfunktion *E (encrypt)*: $E(K_1, T) \rightarrow C$
  - Entschlüsselungsfunktion *D (decrypt)*: $D(K_2, C) \rightarrow T$
  - K = Schlüssel, T= zu verschlüsselnder Text/Daten
- Verwandte Schlüssel
  - es gilt: K_1 und K_2 sind verwandt, wenn gilt: $\forall T: D(K_2, E(K_1, T)) = T$
- Symmetrisches Verschlüsselungsverfahren
  - es gilt: $K_1 = K_2$

5 Kryptographische Maßnahmen (3)

- Forderungen an ein Verschlüsselungsverfahren
  - Wenn K_2 unbekannt ist, soll es sehr aufwendig sein aus $E(K_1, T)$ das T zu ermitteln (Entschlüsselungsangriff)
  - Es soll sehr aufwendig sein aus T und $E(K_1, T)$ den Schlüssel K_1 zu ermitteln (Klartextangriff)
  - Bei asymmetrischen Verfahren soll es sehr aufwendig sein, aus K_1 den Schlüssel K_2 zu ermitteln und umgekehrt.

5.1 Monoalphabetische Verfahren

- Verfahren nach Caesar

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

  - Verschlüsselungsfunktion: $E: M \rightarrow (M + k) \bmod 26$
  - k ist variierbar (26 Möglichkeiten)

- Zufällige Substitution

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

  - 26! Möglichkeiten

5.1 Monoalphabetische Verfahren (2)

★ Nachteil

- vollständiges Ausprobieren möglich bei Caesar
- Häufigkeitsanalyse der Buchstaben
  - für eine Sprache gibt es häufigere Buchstaben, z.B. e im Deutschen
  - durch die Häufigkeitsanalyse können die Möglichkeiten stark eingeschränkt werden; vollständiges Probieren wird ermöglicht

5.2 Polyalphabetische Verschlüsselung

- Einsatz von vielen Abbildungen, die durch einen Schlüssel ausgewählt werden
  - Beispiel: Vigenére (Caesar-Verschlüsselung mit zyklisch wiederholten Folgen von Verschiebungswerten)

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

5.2 Polyalphabetische Verschlüsselung (2)

- Auswahl der Zeile durch den entsprechenden Buchstaben des Schlüsselwortes

W	I	C	H	T	I	G	E	N	A	C	H	R	I	C	H	T	Originaltext
G	E	H	E	I	M	G	E	H	E	I	M	G	E	H	E	I	Schlüsselwort (wiederholt)
C	M	J	L	B	U	M	I	U	E	K	T	X	M	J	L	B	verschlüsselter Text

- Gilt als nicht sicher
  - Koinzidenzanalyse
  - Häufigkeitsanalysen und Brute force Attacke

5.2 Polyalphabetische Verfahren (3)

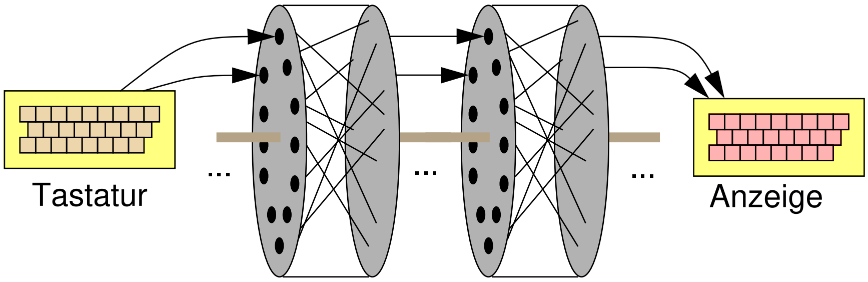
- Koinzidenz
  - Wahrscheinlichkeit für zwei gleiche Buchstaben untereinander bei umbrechendem Text
    - zufällige Buchstabenwahl: 3,8%
    - englischer Text: 6,6%
  - Brechen polyalphabetischer Verfahren
    - Bestimmen der Koinzidenz für verschiedene Textlängen
    - Textlänge mit höchster Koinzidenz ist wahrscheinlich Schlüsseltextlänge
    - danach Häufigkeitsanalyse pro Buchstabe des Schlüsseltexts

5.3 One-Time Pad Verfahren

- Theoretisch sicheres Verfahren
  - Liste von Zufallszahlen (soviele wie Zeichen in der Nachricht): *r[i]*
  - Zeichen *z[i]* der Nachricht wird verschlüsselt mit *c[i] = (z[i] + r[i]) mod 26*
  - Empfänger braucht die gleiche Liste
  - theoretisch sicher, da aus dem *c[]* nicht auf *z[]* geschlossen werden kann
- ▲ Praktisch unbrauchbar
  - echte Zufallszahlen nötig
  - lange Liste nötig
    - jede Liste kann nur einmal verwendet werden
    - Liste muss so lang wie die Nachricht sein

5.4 Rotormaschinen

- Drehende Scheiben verändern ständig die Permutation

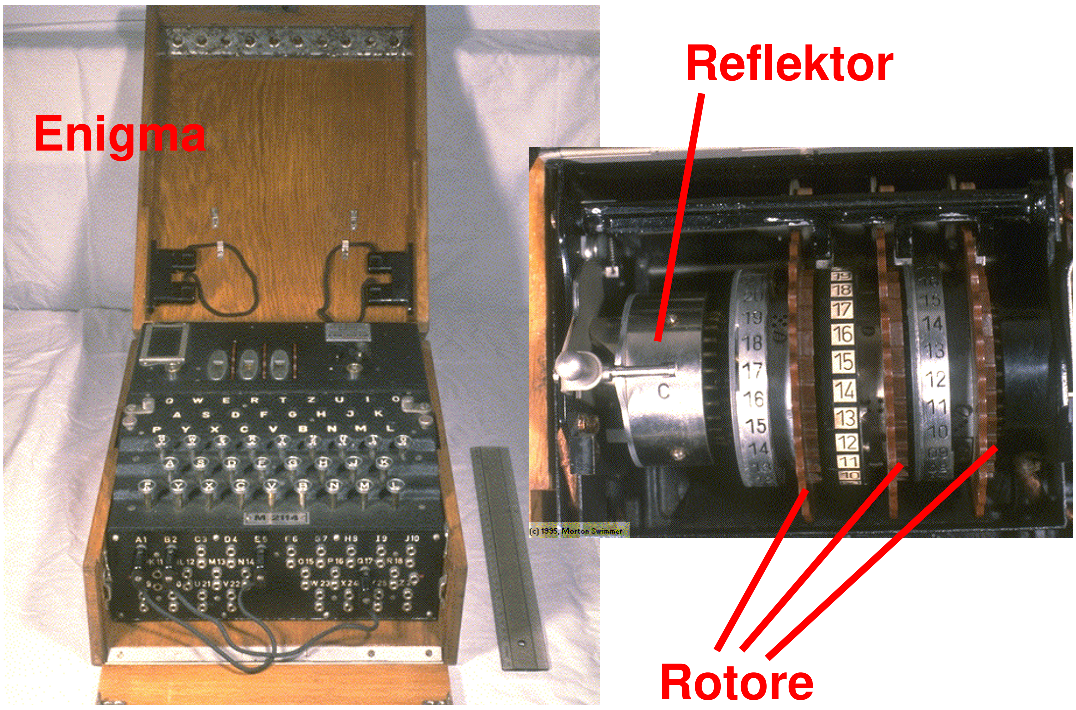


  - Einstellen einer Anfangsposition für die Rotoren
  - bei jedem Zeichen wird erster Rotor um eine Position weitergedreht
  - zweiter Rotor rotiert mit niedrigerer Geschwindigkeit
  - zum Entschlüsseln sind entsprechende Gegenstücke nötig

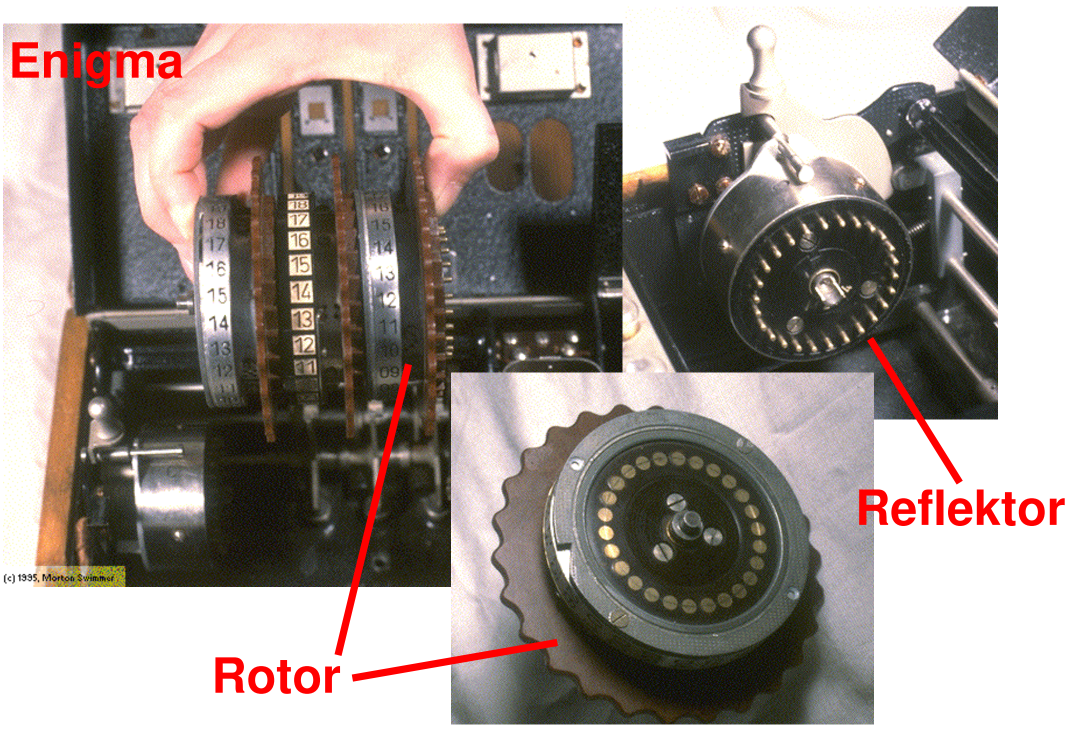
5.4 Rotormaschinen (2)

- Enigma
  - deutsche Chiffriermaschine aus dem zweiten Weltkrieg
  - drei Rotore und Reflektor
    - Reflektor leitet Strom wieder bei einer anderen Position durch die Rotoren zurück: Verfahren wird symmetrisch
    - Entschlüsseln mit den gleichen Rotoren möglich
- Verfahren gilt als nicht sicher
  - Brute force Attacke: *Collosus* Computer
- Schlüsseldemo
  - http://www.ugrad.cs.jhu.edu/~russell/classes/enigma/

5.4 Rotormaschinen



5.4 Rotormaschinen (2)

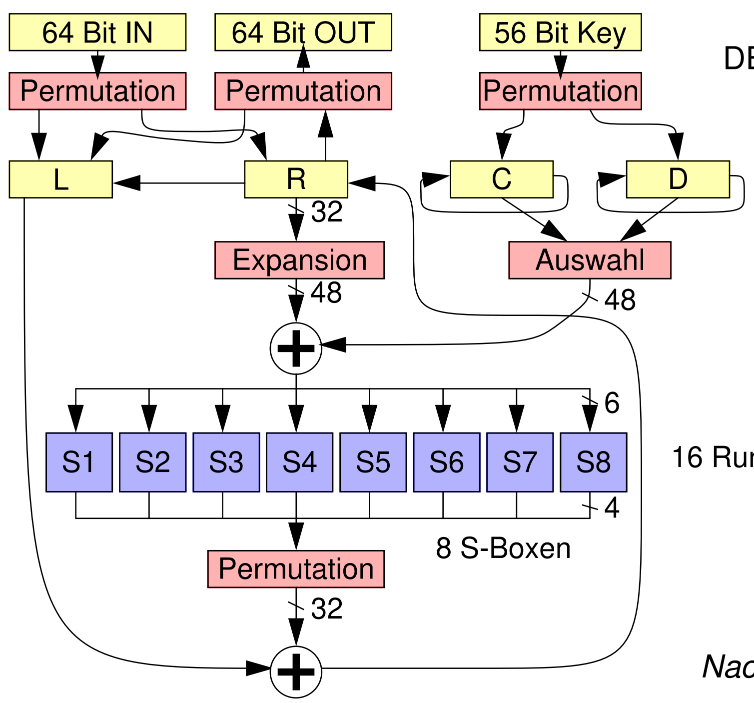


5.5 Heutige symmetrische Verfahren

- DES *(Data Encryption Standard, 1977)*
  - entwickelt von IBM
  - amerikanischer Standard (Kriegswaffe)
  - blockorientiertes Verfahren (64 Bit Block, 56 Bit Schlüssel)
  - 16 Runden
  - gilt heute als nicht mehr ganz sicher, da Rechenleistung von Großrechnern oder Rechenverbünden zum Brechen manchmal ausreicht

5.5 Heutige symmetrische Verfahren (2)

DES Blockdiagramm



16 Runden

Nach: Fumy/Rieß 1994

5.5 Heutige symmetrische Verfahren (3)

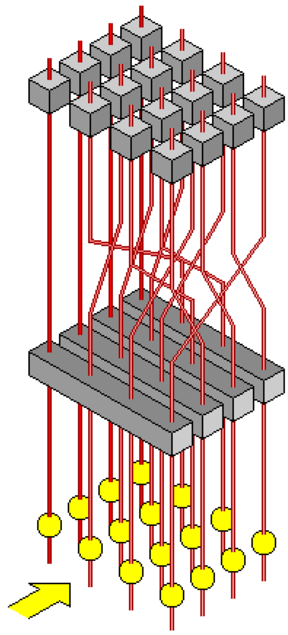
- Tripple DES
  - ◆ dreifache Verschlüsselung mit DES
  - ◆ Nutzung von drei oder mindestens zwei verschiedenen Schlüsseln
- IDEA *(International Data Encryption Algorithm)*
  - ◆ Alternative zu DES
  - ◆ 64 Bit Blockgröße
  - ◆ 128 Bit Schlüssel
  - ◆ keine Permutationen und S-Boxen
  - ◆ stattdessen: Addition, Multiplikation und XOR
  - ◆ 8 Runden und Output-Transformation
  - ◆ Einsatz: z.B. PGP *(Pretty Good Privacy)*

5.5 Heutige symmetrische Verfahren (4)

- AES — Advanced Encryption Standard (Rijndael)
  - ◆ entwickelt von Joan Daemen und Vincent Rijmen
  - ◆ blockorientiertes Verfahren
    - Blockgröße 128, 192 oder 256 Bits
    - Schlüsselgröße 128, 192 oder 256 Bits
  - ◆ 9, 11 oder 13 Runden je nach Schlüssellänge
  - ◆ wurde aus mehreren Vorschlägen als Nachfolger für DES ausgewählt

5.5 Heutige symmetrische Verfahren (5)

- Blockdiagramm einer Runde (stark vereinfacht)



Nach John Savard 2000

5.6 Beispiel: UNIX Passwörter

- Passwörter wurden zunächst im Klartext gespeichert
  - Passwortdatei muss streng geschützt werden
  - strenger Schutz oft nicht möglich (z.B. Backup der Platte)
  - Superuser kann die Passwörter von Benutzern einsehen
- Verschlüsseln der Passwörter
  - nur die verschlüsselte Version wird gespeichert
  - verschlüsselte Passwörter dürfen nicht leicht entschlüsselt werden können
- Ausprobieren von Passwörtern
  - Benutzer wählen Namen und Gegenstände als Passwort
  - Verschlüsseln von gängigen Begriffen und Vergleich mit verschlüsselt gespeicherten Passwörtern
  - Verschlüsselungszeit fließt mit ein in die Sicherheitsbetrachtung

5.6 Beispiel: UNIX Passwörter (2)

- Heutiges Verfahren
  - zufällige Auswahl eines von 4096 Werten *(Salt)*
  - der Salt fließt mit in die Verschlüsselung ein, so dass ein und dasselbe Passwort in 4096 Varianten vorkommen kann
  - Verschlüsselung mit DES
  - Zugriff auf verschlüsselte Passwörter wird weitestgehend verhindert (Shadow-Passwortdatei)
- ★ Vorteil
  - Ausprobieren von Passwörtern benötigt mehr Zeit
  - Vergleich zweier Passwörter weitgehend unmöglich

5.6 Beispiel: UNIX Passwörter (3)

- Politik am Institut für Informatik
  - Mindestlänge 8 Zeichen
  - mindestens 5 verschiedene Zeichen
  - mindestens 3 Zeichenklassen (Groß-, Kleinbuchstaben, Ziffern, Sonderzeichen)
  - keine Wiederholungen von Zeichenfolgen erlaubt
  - keine aufeinanderfolgenden Zeichen erlaubt, z.B. "123"
  - ...
  - Begriffe, Namen, etc. werden ausgeschlossen und müssen hinreichend verfremdet sein
- ★ Angriff durch Ausprobieren wird weitestmöglich erschwert

5.7 Heutige asymmetrische Verfahren

- RSA (Rivest, Shamir und Adleman)
  - Öffentlicher Schlüssel (zum Verschlüsseln) besteht aus (e, N)
  - Ein Block M wird verschlüsselt durch: $C = E((e, N), M) = M^e \bmod N$
  - C wird entschlüsselt durch: $M = D((d, N), C) = C^d \bmod N$
  - Wahl der Schlüssel:
    - Es muss gelten $\forall M : (M^e)^d = M \bmod N$
    - Aus Kenntnis von e und N darf d nur mit hohem Aufwand ermittelbar sein
  - Lösung:
    - $N = pq$ mit p und q zwei hinreichend große Primzahlen
    - zufällige Wahl von d, teilerfremd zu $(p-1)(q-1)$
    - Berechnung von e aus der Bedingung: $ed = 1 \bmod ((p-1)(q-1))$
  - Es ist aufwendig, die Primfaktoren von N zu berechnen (mit diesen wäre es möglich d zu ermitteln)

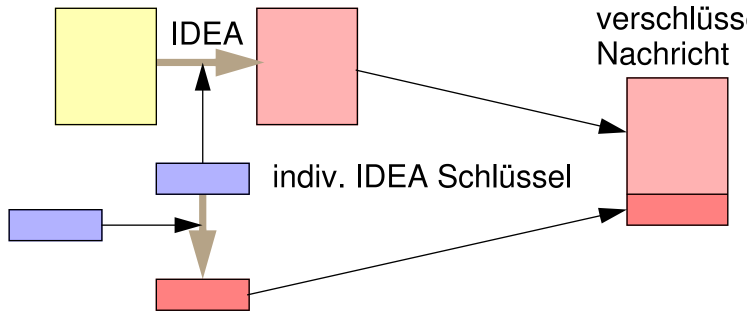
5.7 Heutige asymmetrische Verfahren (2)

- Vorteil asymmetrischer Verfahren (*Public key*-Verfahren)
  - nur ein Schlüsselpaar pro Teilnehmer nötig
    (sonst ein Schlüsselpaar pro Kommunikationskanal!)
  - Schlüsselverwaltung erheblich vereinfacht
    - jeder Teilnehmer erzeugt sein Schlüsselpaar und
    - veröffentlicht seinen öffentlichen Schlüssel
  - Authentisierung durch digitale Unterschriften möglich
  - gilt als sicher bei hinreichend großer Schlüssellänge (1024 Bit)

- Nachteil
  - relativ langsam berechenbar
  - gemischter Betrieb von asymmetrischen und symmetrischen Verfahren zur Geschwindigkeitssteigerung

5.7 Heutige asymmetrische Verfahren (3)

- Beispiel: PGP Verschlüsselung



- Daten werden mit einem individuellen Schlüssel IDEA-verschlüsselt
- IDEA-Schlüssel wird RSA-verschlüsselt der Nachricht angehängt

5.7 Heutige asymmetrische Verfahren (4)

★ Nachricht an mehrere Adressaten verschickbar

- lediglich der IDEA-Schlüssel muss in mehreren Varianten verschickt werden
(je eine Version verschlüsselt mit dem öffentl. Schlüssel des jeweiligen Empfängers)

5.8 Digitale Unterschriften

- Authentisierung des Absenders
  - Bilden eines Hash-Wertes über die zu übermittelnde Nachricht
    - Hash-Wert ist ein Codewort fester Länge
    - es ist unmöglich oder nur mit hohem Aufwand möglich, für einen gegebenen Hash-Wert eine zugehörige Nachricht zu finden
  - Verschlüsseln des Hash-Wertes mit dem geheimen Schlüssel des Absenders (digitale Unterschrift, digitale Signatur)
  - Anhängen des verschlüsselten Hash-Wertes an die Nachricht

  - Empfänger kann den Hash-Wert mit dem öffentlichen Schlüssel des Absenders dechiffrieren und mit einem selbst berechneten Hash-Wert der Nachricht vergleichen
  - stimmen beide Werte überein muss die Nachricht vom Absender stammen, denn nur der besitzt den geheimen Schlüssel

5.8 Digitale Unterschriften (2)

- Kombination mit Verschlüsselung
  - erst signieren
  - dann mit dem öffentlichen Schlüssel des Adressaten verschlüsseln
  - sonst Signatur verfälschbar
- Reihenfolge wichtig
  - Man signiere nichts, was man nicht entschlüsseln kann / versteht.
- Heute gängiges Hash-Verfahren
  - MD5
  - 128 Bit langer Hash-Wert

5.8 Digitale Unterschriften (3)

- Woher weiß ich, dass ein öffentlicher Schlüssel authentisch ist?
  - Ich bekomme den Schlüssel vom Eigentümer (persönlich, telefonisch).
    - Hash-Wert auf öffentlichen Schlüsseln, die leichter zu überprüfen sind (Finger-Prints)
  - Ich vertraue jemandem (Bürge), der zusichert, dass der Schlüssel authentisch ist.
    - Schlüssel werden von dem Bürgen signiert.
    - Bürge kann auch eine ausgezeichnete Zertifizierungsstelle sein.
    - Netzwerk von Zusicherungen auf öffentliche Schlüssel *(Web of Trust)*
  - Möglichst weite Verbreitung von öffentlichen Schlüsseln erreichen (z.B. PGP: Webserver als Schlüsselserver)

5.8 Digitale Unterschriften (4)

- Mögliche Probleme von Public key-Verfahren
  - Geheimhaltung des geheimen Schlüssels (Time sharing-System, Backup; Schlüsselpasswort / Pass phrase)
  - Vertrauen in die Programme (z.B. PGP)
  - Ausspähung während des Ver- und Entschlüsselungsvorgangs

6 Authentisierung im Netzwerk

- Viele Klienten, die viele Dienste in Anspruch nehmen wollen
  - Dienste *(Server)* wollen wissen welcher Benutzer *(Principal)*, den Dienst in Anspruch nehmen will (z.B. zum Accounting, Zugriffsschutz, etc.)
  - Im lokalen System reicht die (durch das Betriebssystem) geschützte Benutzerkennung (z.B. UNIX UID) als Ausweis
  - Im Netzwerk können Pakete abgefangen, verfälscht und gefälscht werden (einfache Übertragung einer Benutzerkennung nicht ausreichend sicher)
- Public key-Verfahren
  - Authentisierung durch digitale Unterschrift (mit geheimen Schlüssel des Senders) und Verschlüsseln (mit öffentlichem Schlüssel des Empfängers)
  - Nachteile
    - jeder Dienst benötigt sicheren Zugang zu allen öffentlichen Schlüsseln
    - Verschlüsseln und Signieren mit RSA ist sehr teuer

6 Authentisierung im Netzwerk (2)

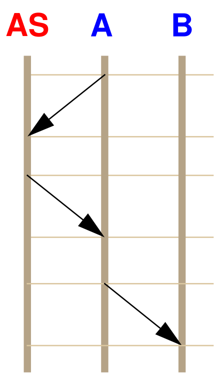
★ Einsatz von Authentisierungsdiensten
  - zentraler Server, der alle Benutzer kennt
  - Authentisierungsdienst garantiert einem Netzwerkdienst, dass ein Benutzer auch der ist, der er vorgibt zu sein
- Benutzerausweis
  - Authentisierungsdienst erkennt den Benutzer anhand eines geheimen Schlüssels oder Passworts
  - Schlüssel ist nur dem Authentisierungsdienst und dem Benutzer bekannt

6 Authentisierung im Netzwerk (3)

- Vorgang
  - Benutzer *(Principal)* will mit einem Programm *(Client)* einen Dienst *(Server)* in Anspruch nehmen
  - durch geeignetes Protokoll erhalten Client und Server jeweils einen nur ihnen bekannten Schlüssel, mit dem sie ihre Kommunikation verschlüsseln können *(Session key)*

6.1 Einfacher Authentisierungsdienst

- A will den Dienst B in Anspruch nehmen (Nach Needham-Schröder):



A schickt an den Authentisierungsdienst: a, b

AS erzeugt einen Session key $S_{A,B}$

AS antwortet: E(K_A, <b, $S_{A,B}$, E(K_B, <$S_{A,B}$, a>)>)

A entschlüsselt mit K_A die Nachricht

A sendet an B: E(K_B, <$S_{A,B}$, a>)

B entschlüsselt mit K_B die Nachricht

  - K_x ist der geheime Schlüssel, den nur Authentisierungsdienst und X kennen
  - nach dem Protokollablauf kennen sowohl A und B den Session key
  - A weiß, dass nur B den Session key kennt
  - B weiß, dass nur A den Session key kennt

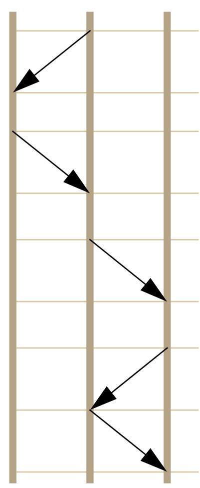
6.1 Einfacher Authentisierungsdienst (2)

- Problem
  - letzte Nachricht von A an B könnte aufgefangen und später erneut ins Netz gegeben werden *(Replay attack)*
  - Folge: Kommunikation zwischen A und B kann gestört werden
- Korrektur durch zusätzliches Versenden einer Verbindungsbestätigung durch B und A

6.2 Authentisierungsdienst mit Bestätigung

- Bestätigung enthält Einmalinformation *(Nonce)*

AS A B



A schickt an den Authentisierungsdienst: a, b, i_a

AS erzeugt einen Session key $S_{A,B}$
AS antwortet: $E(K_A, <i_a, b, S_{A,B}, E(K_B, <S_{A,B}, a>)>)$

A entschlüsselt mit K_A die Nachricht
A sendet an B: $E(K_B, <S_{A,B}, a>)$

B entschlüsselt mit K_B die Nachricht
B schickt an A: $E(S_{A,B}, i_b)$

A antwortet an B: $E(S_{A,B}, i_b+1)$

- ein Wiedereinspielen der Nachricht $E(K_B, <S_{A,B}, a>)$ oder $E(S_{A,B}, i_b+1)$ wird erkannt und kann ignoriert werden

6.2 Authentisierungsdienst mit Bestätigung (2)

▲ Problem

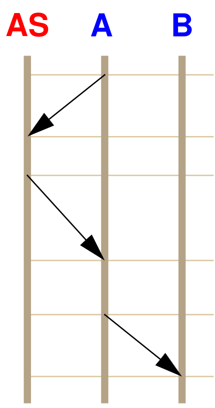
- Aufzeichnen von $E(K_B, <S_{A,B}, a>)$ und
- Brechen von $S_{A,B}$ erlaubt das Aufbauen einer Verbindung.
- ein Dritter kann dann die erste Bestätigung abfangen und die zweite Bestätigung verschicken

★ Lösung

- Einführung von Zeitstempeln *(Time stamp)* und Angaben zur Lebensdauer *(Expiration time)*

6.3 Authentisierungsdienst mit Zeitstempeln

■ Authentisierungsdienst versieht seine Nachrichten mit Zeitstempeln



A schickt an den Authentisierungsdienst: a, b

AS erzeugt einen Session key $S_{A,B}$

AS antwortet:
$E(K_A, <b, S_{A,B}, T_S, T_{Exp}, E(K_B, <S_{A,B}, a, T_S, T_{Exp}>)>)$

A entschlüsselt mit KA die Nachricht

A sendet an B: $E(K_B, <S_{A,B}, a, T_S, T_{Exp}>)$

B entschlüsselt mit KB die Nachricht

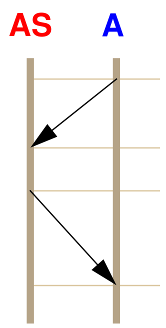
- T_S = Zeitstempel der Nachrichtenerzeugung
- T_{Exp} = maximale Lebensdauer der Nachricht
- aufgezeichnete Nachricht kann nach kurzer Zeit (z.B. 5min) nicht nocheinmal zum Aufbau einer Verbindung verwendet werden

6.4 Beispiel: Kerberos

- Kerberos V5
  - Softwaresystem implementiert Weiterentwicklung des Needham-Schröder-Protokolls
  - entwickelt am MIT seit 1986
- Ziel
  - Authentisierung und Erzeugung eines gemeinsamen Schlüssels durch den vertrauenswürdigen Kerberos-Server
- Idee
  - Trennung von Authentisierungsdienst und Schlüsselerzeugung
  - reduziert die nötige Übertragung einer Identifikation oder eines Passworts zum Kerberos-Server

6.4 Beispiel: Kerberos (2)

- Benutzer holt sich zunächst ein Ticket vom Authentisierungsdienst



A schickt an den Authentisierungsdienst: a, tgs, T_{Exp}, i_a

AS erzeugt ein Ticket für den Ticket Server tgs

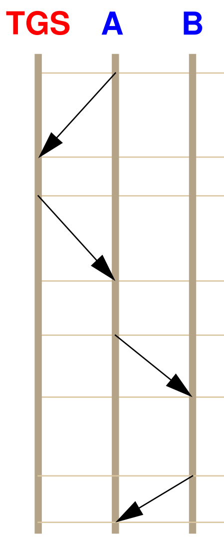
AS antwortet:
E(K_A, <$S_{A,TGS}$, tgs, T_{Exp}, i_a, E(K_{TGS}, <$S_{A,TGS}$, a, T_{Exp}>)>)

A entschlüsselt mit KA die Nachricht

  - das Ticket besteht aus <$S_{A,TGS}$, a, T_{Exp}>
  - es enthält einen Session key für die Kommunikation mit einem Ticket granting server, der dann die Verbindung zu einem Netzwerkdienst bereitstellen kann

6.4 Beispiel: Kerberos (3)

- Authentisierungsdienst versieht seine Nachrichten mit Zeitstempeln



A schickt an den Ticket granting server:
$E(S_{A,TGS}, T_s), E(K_{TGS}, \langle S_{A,TGS}, a, T_{Exp}\rangle), b, T_{Exp}, i_a'$

TGS erzeugt einen Session key $S_{A,B}$

TGS antwortet:
$E(S_{A,TGS}, \langle S_{A,B}, b, T_{Exp}, i_a'\rangle), E(K_B, \langle S_{A,B}, a, T_{Exp}\rangle)$

A entschlüsselt mit $S_{A,TGS}$ die Nachricht

A sendet an B: $E(S_{A,B}, \langle T_s, C, S_{sub}\rangle), E(K_B, \langle S_{A,B}, a, T_{Exp}\rangle)$

B entschlüsselt mit K_B den zweiten Teil der Nachricht (Ticket) und mit $S_{A,B}$ den ersten Teil

B antwortet (optional): $E(S_{A,B}, T_s)$

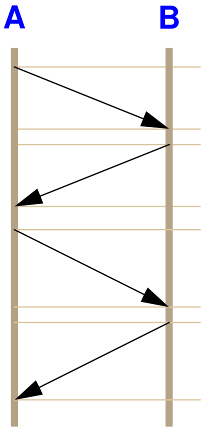
- ◆ C = Checksumme zur Überprüfung der richtigen Entschlüsselung
- ◆ A kann mehrere Verbindungen mit seinem Ticket öffnen

6.4 Beispiel: Kerberos (4)

- Unterscheidung zwischen Benutzer *(User)* und Benutzerprogramm *(Client)*
  - ◆ Wie kann ein Benutzerprogramm seinen Benutzer identifizieren?
  - ◆ Geheimer Schlüssel vom Benutzer (K_X) hängt von einem Passwort ab
  - ◆ mittels einer Einwegfunktion wird aus dem Passwort der Schlüssel K_X erzeugt
  - ◆ Benutzerprogramm braucht also das Passwort zur Verbindungsaufnahme
- Beispiel: *kinit, klogin*
  - ◆ Anmeldung beim Authentisierungsdienst mit *kinit* und Passworteingabe
  - ◆ Ticket wird im Benutzerverzeichnis gespeichert
  - ◆ *klogin* erlaubt das Einloggen auf einem entfernten Rechner mit Datenverschlüsselung und ohne Passwort

6.5 Austausch öffentlicher Schlüssel

- A und B tauschen ihre öffentlichen Schlüssel aus



A schickt an B seinen öffentlichen Schlüssel: K_A

B schickt an A seinen öffentlichen Schlüssel: K_B

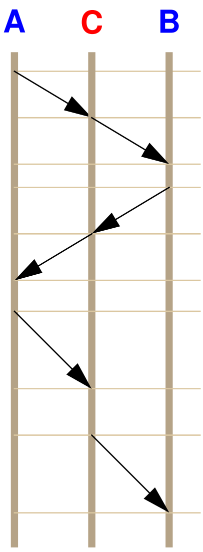
A verschickt signierte und verschlüsselte Nachricht:
$E(K_B, E(K_A^{-1}, T_1))$

B verschickt signierte und verschlüsselte Nachricht:
$E(K_A, E(K_B^{-1}, T_2))$

- Problem
  - A und B können nicht sicher sein, dass der öffentliche Schlüssel wirklich vom jeweils anderen stammt

6.5 Austausch öffentlicher Schlüssel (2)

- Aktiver Mithörer C fängt Datenverbindungen ab *(Man in the middle attack)*



A versucht an B seinen öffentlichen Schlüssel zu schicken: K_A

C fängt Nachricht ab und schickt an B seinen eigenen öffentlichen Schlüssel: K_C

B versucht an A seinen öffentlichen Schlüssel zu schicken: K_B

C fängt Nachricht ab und schickt an A seinen eigenen öffentlichen Schlüssel: K_C

A versucht an B eine Nachricht zu schicken:
$C_1 = E(K_C, E(K_A^{-1}, T_1))$

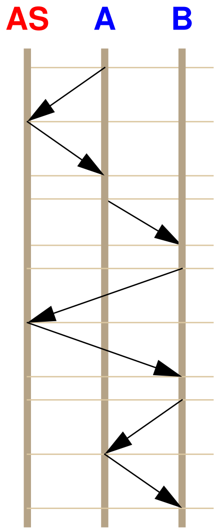
C dekodiert die Nachricht: $T_1 = D(K_C^{-1}, D(K_A, C_1))$

C schickt die Nachricht weiter: $E(K_B, E(K_C^{-1}, T_1))$

B glaubt eine Nachricht von A zu empfangen

6.5 Austausch öffentlicher Schlüssel (3)

- Einsatz eines Authentisierungsdienstes



A kontaktet AS und möchte Verbindung mit B: a, b

AS verschickt den Schlüssel von B: $E(K_{AS}^{-1}, <K_B, b>)$

A schickt initiale Nachricht an B: $E(K_B, <a, i_a>)$

B kontaktet AS und möchte den Schlüssel von A: b, a

AS verschickt den Schlüssel von A: $E(K_{AS}^{-1}, <K_A, a>)$

B antwortet an A: $E(K_A, <i_a, i_b>)$

A bestätigt: $E(K_B, <i_b>)$

- Replay-Probleme
  - ◆ Hinzunahme von Zeitstempel und Lebendauer

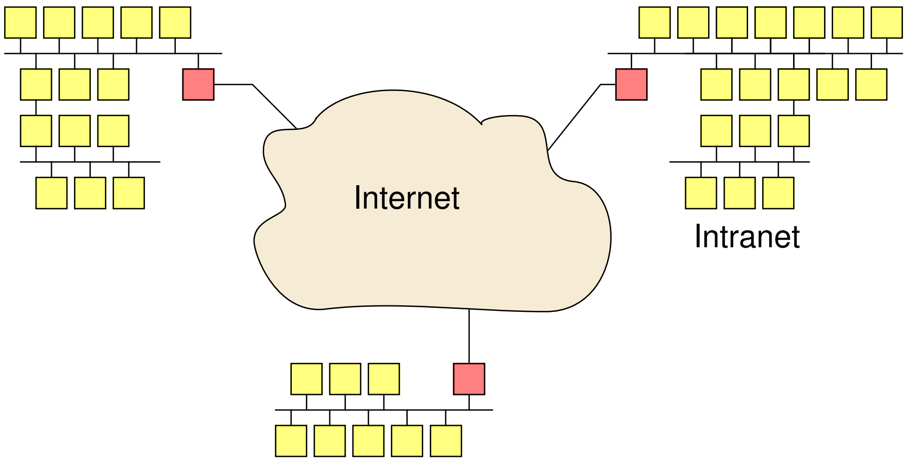
7 Firewall

- Trennung von vertrauenswürdigen und nicht vertrauenswürdigen Netzwerksegmenten durch spezielle Hardware *(Firewall)*
  - ◆ Beispiel: Trennen des firmeninternen Netzwerks (Intranet) vom allgemeinen Internet

- Funktionalität
  - ◆ Einschränkung von Diensten
    - von innen nach außen, z.B. nur Standarddienste
    - von außen nach innen, z.B. kein Telnet, nur WWW
  - ◆ Paketfilter
    - Filtern „defekter“ Pakete, z.B. SYN-Pakete
  - ◆ Inhaltsfilter
    - Filtern von Pornomaterial aus dem WWW oder News
  - ◆ Authentisieren von Benutzern vor der Nutzung von Diensten

7 Firewall (2)

- Virtual private network
  - Verbinden von Intranet-Inseln durch spezielle Tunnels zwischen Firewalls
  - getunnelter Datenverkehr wird verschlüsselt
  - Benutzer sieht ein „großes Intranet“ (nur virtuell vorhanden)



8 Richtlinien für den Benutzer

8.1 Passwörter

- Wahl eines Passworts
  - hinreichend komplexe Passwörter wählen
  - Schutz vor Wörterbuchangriffen
  - verschiedene Passwörter für verschiedene Aufgaben (z.B. PPP-Passwort ungleich Benutzerpasswort)
- Aufbewahrung
  - möglichst nirgends aufschreiben
  - nicht weitergeben
  - kein Abspeichern auf einem Windows-Rechner (Option immer wegklicken)

8.1 Passwörter

- Eingabe
  - ◆ niemals über eine unsichere Rechnerverbindung eingeben
    - **`ftp, telnet, rlogin`** Dienste vermeiden
    - nur sichere Dienste verwenden: **`ssh, slogin`**
    - Datenweg beachten, über den das Passwort läuft:
      ein unsicheres Netzwerk ist bereits genug
- Änderung
  - ◆ Passwörter regelmäßig wechseln
  - ◆ alte Passwörter nicht wiederverwenden

8.2 Schlüsselhandhabung

- Einsatz von PGP oder S/MIME
  - ◆ Zugang zu den privaten Schlüsseln für andere verhindern
  - ◆ Dateirechte auf der Schlüsseldatei prüfen
  - ◆ privater Schlüssel nur auf Diskette
  - ◆ Passphrase wie ein Passwort behandeln
  - ◆ privaten Schlüssel nie über unsichere Netze transportieren

8.3 E-Mail

- Authentisierung
  - Bei elektronischer Post ist der Absender nicht authentisierbar
  - Digitale Unterschriften einsetzen (z.B. mit PGP oder S/MIME)
- Abhören
  - Elektronische Post durchläuft viele Zwischenstationen und kann dort jeweils gelesen und verfälscht werden
  - Verschlüsselung einsetzen (z.B. mit PGP oder S/MIME)

8.4 Programmierung

- S-Bit Programme vermeiden
  - Oft kann das S-Bit durch geschickte Vergabe von Benutzergruppen an Dateien vermieden werden
- Verwendung zusätzlicher Rechte (z.B. durch S-Bit) nur in Abschnitten
  - Trusted Computing Base (TCB) [hier kein vollständiger Schutz]

```
seteuid(getuid());/* am Programmanfang Rechte wegnehmen */
...
seteuid(0);         /* setzt root Rechte */
fd = open("/etc/passwd", O_RDWR);
seteuid(getuid());/* nimmt root Rechte wieder weg */
...
```

- Sorgfältige Programmierung
  - Funktionen wie **strcpy, strcat, gets, sprintf, scanf, sscanf, system, popen** vermeiden oder durch **strncpy, fgets, snprintf** ersetzen

8.5 World Wide Web

- Cookies
  - Akzeptieren von Cookies erlaubt einer Website die angesprochenen Seiten genau einem Benutzer zuzuordnen
  - funktioniert über Sessions hinweg
- JavaScript
  - schwere Sicherheitslücken erlauben es, alle für den Benutzer lesbare Dateien an einen Dritten weiterzugeben
  - Ausschalten!
- Abhören
  - WWW-Verbindungen können abgehört werden
  - keine privaten Daten, wie z.B. Kreditkartennummern übertragen
  - Secure-HTTP mit SSL-Verschlüsselung benutzen; https-URLs

© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[I-Security.fm, 2004-02-02 08.52]
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.