

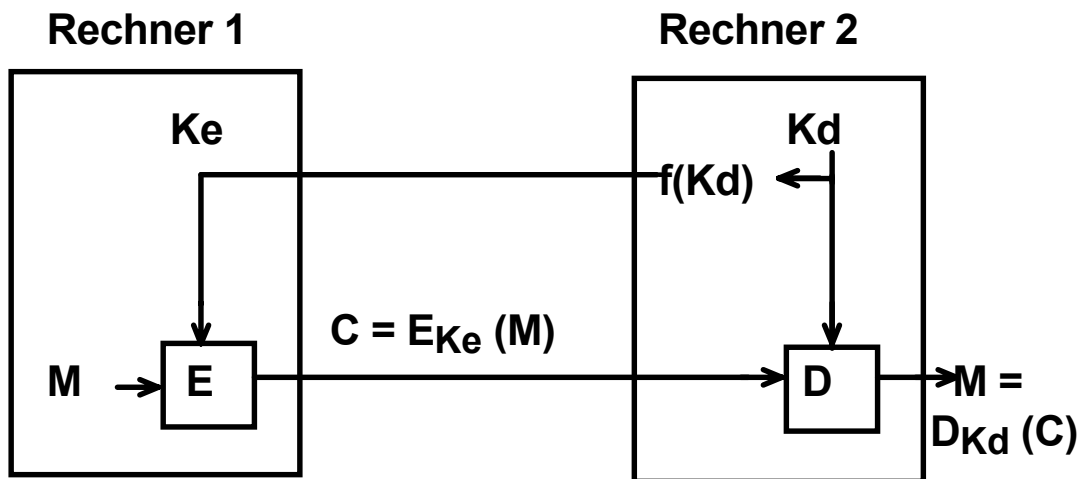
# **Client/Server-Systeme**

**Prof. Dr.-Ing. Wilhelm G. Spruth**

**WS 2003/2004**

**Teil 5**

**Authentifizierung**



## Asymetrische Verfahren

### Public Key Encryption

Für die Verschlüsselung wird ein anderer Key verwendet als für die Entschlüsselung. Deswegen kann der Verschlüsselungskey bekannt gegeben werden.

Asymetrische Verfahren benötigen etwa 1000 mal mehr Verarbeitungszeit als symmetrische Verfahren.

# Asymmetrische Verfahren

## Public Key Encryption

Algorithmen E und D sind öffentlich

$K_e = f(K_d)$ ;  $K_e$  ist öffentlich;  $K_d$  ist geheim

Nur der Empfänger, der  $K_d$  kennt, kann  $M$  lesen

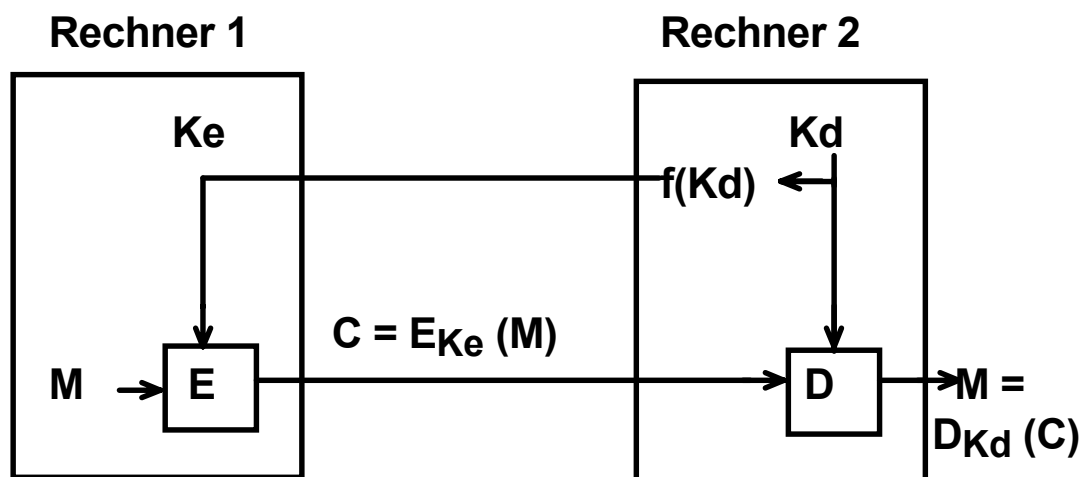
$K_e$  kann einfach aus  $K_d$  errechnet werden, aber die Berechnung von  $K_d$  aus  $K_e$  ist sehr aufwendig (Einweg-, Falltür-, Trapdoor Funktion).

D ist die Inverse zu E :

$$M = D_{K_d}(C) = D_{K_d}(E_{K_e}(M))$$

Vereinfacht die Schlüsselverteilung

100 - 1000 mal langsamer als symmetrische Verfahren



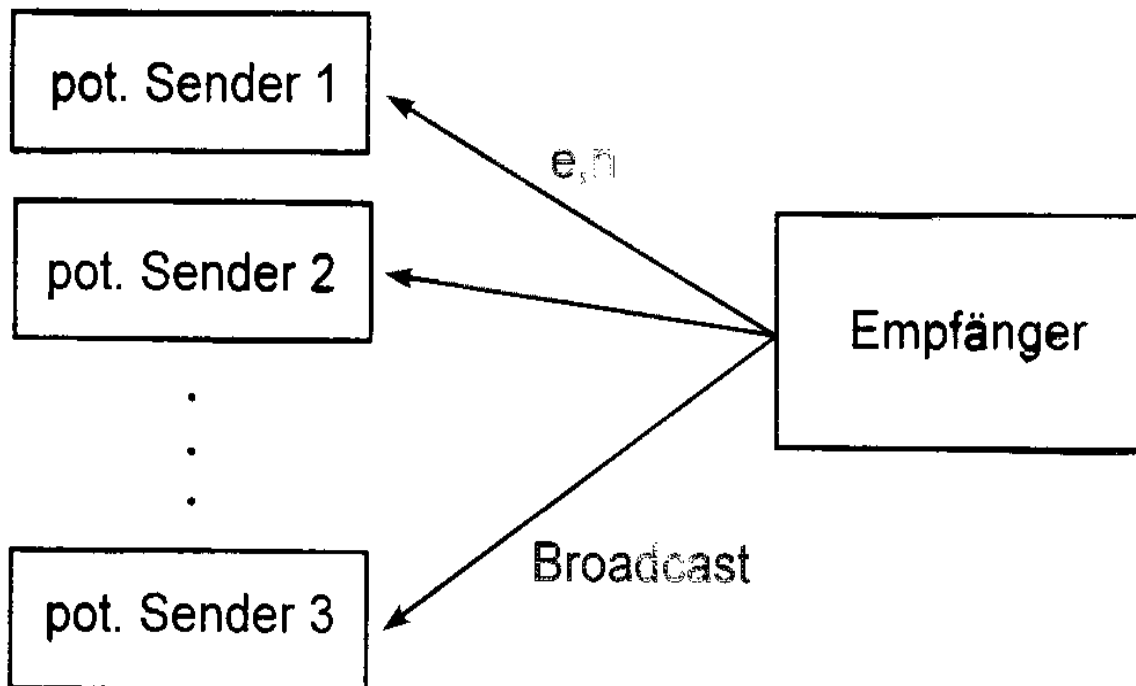
**Entschlüsselung**

**Verschlüsselung**

Eine Einweg-Funktion ist eine beliebige Funktion  $a = F(b)$ , die, gegeben  $a$ ,  $b$  relativ leicht berechnen läßt, bei der es aber sehr rechenaufwendig ist, gegeben  $b$ , den Wert  $a$  zu berechnen.

Angenommen eine große Zahl  $b$  z.B. mit 150 - 300 Stellen, die das Produkt aus 2 Primzahlen  $y$  und  $z$  ist.

Gegeben  $y$  und  $z$ , ist es einfach, hieraus  $b$  zu berechnen. Gegeben  $b$ , ist es extrem rechenaufwendig,  $y$  und  $z$  zu finden.



3. Sender chiffriert Klartext  $m$

$$c = m^e \bmod n$$

4. Empfänger dechiffriert Text

$$m = c^d \bmod n$$

## Vorgehensweise RSA Verschlüsselung

# **RSA Algorithmus (1)**

**Empfänger erzeugt öffentlichen und privaten Schlüssel**

**Ausgangsbasis sind 2 Primzahlen  $p$  und  $q$   
sowie deren Produkt  $n = p \times q$**

**Aus  $p$ ,  $q$  und  $n$  werden 2 Schlüssel abgeleitet:**

- e    öffentlicher Schlüssel (gemeinsam mit  $n$ )**
- d    privater Schlüssel**

## RSA Algorithmus (2)

Der private Schlüssel ist eine ganze positive Zahl  $d$ . Der öffentliche Schlüssel besteht aus 2 ganzen positiven Zahlen  $e$  und  $n$ .

Die Schlüssel werden vom Empfänger festgelegt. Aus 2 zufällig ausgewählten Primzahlen  $p$  und  $q$  wird das Produkt  $n = pq$  gebildet. Die Zahl  $e$  wird so ausgewählt, daß  $(p-1)(q-1)$  und  $e$  keinen gemeinsamen Teiler haben; der größte gemeinsame Teiler soll  $= 1$  sein.

Die Zahl  $d$  hat die Eigenschaften

$$d < (p-1)(q-1)$$

Das Produkt  $ed$  liefert nach Division durch  $(p-1)(q-1)$  den Rest 1.

Die Zahl  $d$  wird vom Empfänger als privater Schlüssel geheim gehalten; die Zahlen  $e$  und  $n$  werden allen Interessenten mitgeteilt (können z.B. beim Empfänger auf einer Web Seite wiedergegeben oder in einer allgemein zugänglichen Datenbank abgefragt werden). Der Sender erfragt  $e$  und  $n$ , und verschlüsselt damit seine Nachricht. Nur der Empfänger kann sie entschlüsseln, weil nur er die Zahl  $d$  kennt.

## **RSA Algorithmus (3)**

**Der Sender teilt die zu verschlüsselnde Nachricht in Blöcke der Länge  $k$  Bits auf. Jeder Block repräsentiert eine Zahl  $m$ . Es muß**

$$2^k < n$$

**sein, d.h., der numerische Wert eines Blockes ist immer kleiner als  $n$ .**

**Zur Verschlüsselung berechnet der Sender Ziffernblöcke  $c$**

$$c := m^e \bmod n$$

**( $e$ -te Potenz der Zahl  $m$ , und davon den Rest nach Division durch  $n$ )**

**Diese Zahl  $c$  ist der Geheimtext, der zum Klartext  $m$  gehört. Der Empfänger, der die chiffrierte Botschaft  $c$  erhält, entschlüsselt sie**

$$m := c^d \bmod n$$

**Es läßt sich nachweisen, daß für Blöcke der Größe  $2^k < n$  die Enkodier- und Dekodierfunktionen invers sind.**

# **RSA Algorithmus (4)**

## **numerisches Beispiel**

**Wir wählen**

$$p = 374\ 1966\ 9101$$

$$q = 111\ 1069\ 3267$$

**(In praktischen Anwendungen würden p und q eine weit größere Anzahl an Stellen haben.) Es ist**

$$pq = n = 4\ 1575\ 8465\ 5338\ 4864\ 2967$$

**Die Zahl e muß zu**

$$(p-1)(q-1) = 4\ 1575\ 8465\ 4853\ 1828\ 0600$$

**teilerfremd sein. Das geht z.B. mit**

$$e = 65537$$

**Der private Schlüssel d muß  $< (p-1)(q-1)$  sein und die Bedingung**

$$ed \bmod (p-1)(q-1) = 1$$

**erfüllen. Dies trifft zu für**

$$d = 1648\ 1384\ 4596\ 3130\ 5873$$

**Sender**

**Empfänger**

- 
1. Primzahlen  $p, q$
  2.  $\leftarrow$   $n = pq$
  3.  $e$  wählen  
 $\leftarrow$   $e$  und  $(p-1)(q-1)$  haben keinen gemeinsamen Teiler
  4.  $d$  wählen  
 $d < (p-1)(q-1)$   
 $ed / (p-1)(q-1)$  hat Rest = 1

$m =$  Klartext  
 $c =$  Chiffrierter Text

$$c = m^e \bmod n \quad \longrightarrow$$

$$m = c^d \bmod n$$

# RSA Algorithmus (5)

Wir verschlüsseln die Nachricht

**kryptologie macht spass**

indem wir die Buchstaben durch die Ziffern 01 .. 26 und das Leerzeichen durch 00 darstellen:

**1118 2516 2015 1215 0709 0500 1301 0308 usw.**

Da diese Zahl  $> n$  ist, muß sie in Blöcke aufgeteilt werden.  
Der erste dieser Blöcke ist

**$m := 1118\ 2516\ 2015\ 1215\ 0709$**

Es ist

**$c := m^e \bmod n$   
 $= 1118\ 2516\ 2015\ 1215\ 0709\ 65537$   
 $\bmod 4\ 1575\ 8465\ 5338\ 4864\ 2967$   
 $= 7104\ 3117\ 9918\ 9756\ 5951$**

Zur Entschlüsselung berechnen wir

**$m := c^d \bmod n$   
 $7104\ 3117\ 9918\ 9756\ 5951\ (1648\ 1384\ 4596\ 3130\ 5873)$   
 $\bmod 4\ 1575\ 8465\ 5338\ 4864\ 2967$   
 $= 1118\ 2516\ 2015\ 1215\ 0709$**

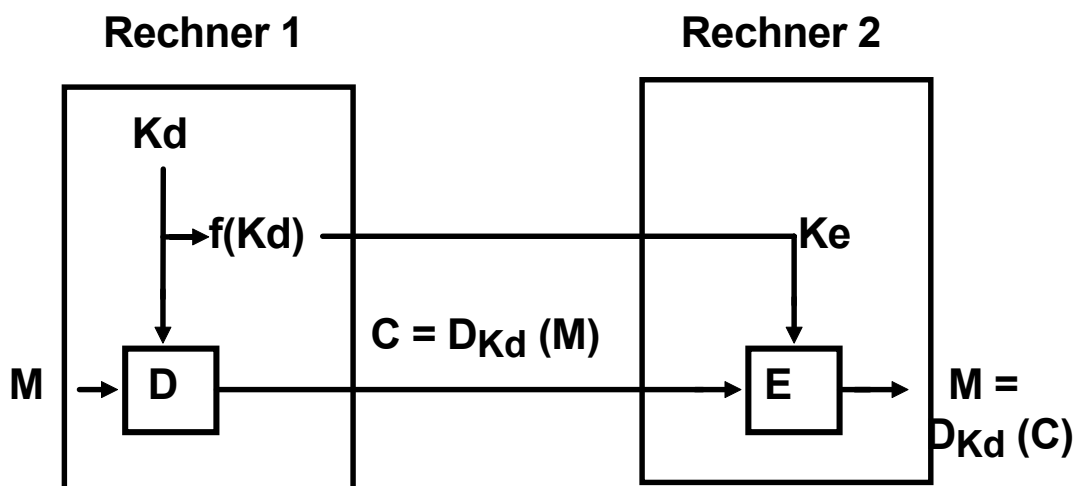
welches der erste Block der geheimen Nachricht ist.

# Digitale Unterschrift

Empfänger kann die angebliche Identität des Senders verifizieren. Späteres Nichtanerkennen der Nachricht durch den Sender ist ausgeschlossen

Realisierung mit asymmetrischen Verfahren mit den beiden Verschlüsselungsalgorithmen D (geheim) und E (öffentlich) möglich, wenn

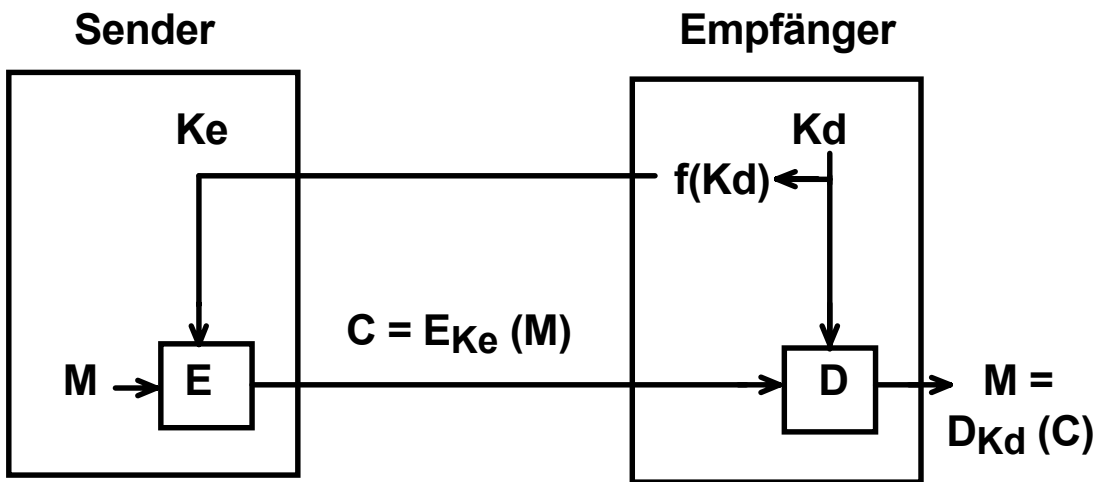
$$M = D(E(M)) \text{ und } M = E(D(M))$$



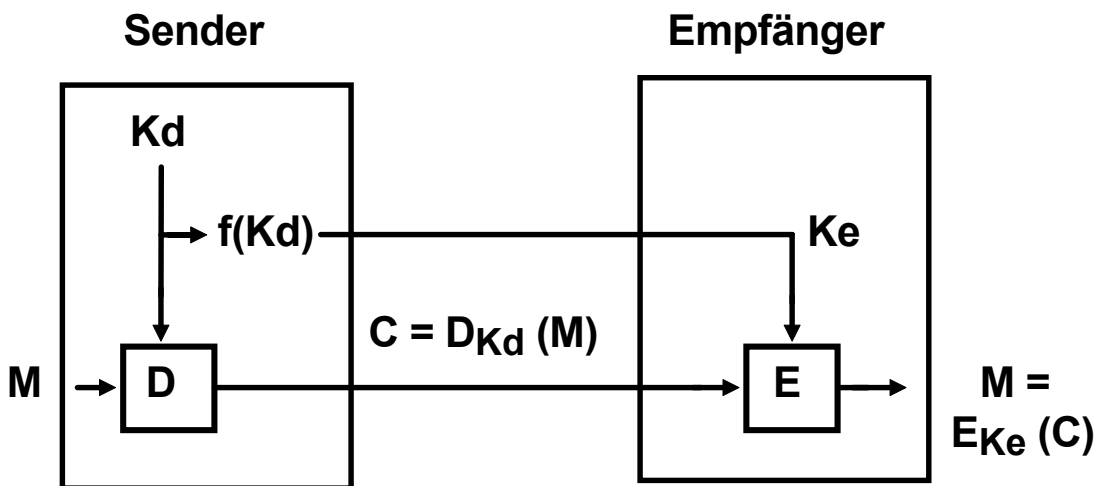
Jeder, der  $K_e$  besitzt, kann  $M$  dechiffrieren  
keine Geheimhaltung

Nur wer  $K_d$  kennt, kann  $M$  generiert haben  
Authentizität

Empfänger speichert  $C$  - Beweis für den Empfang



### Asymmetrische Verschlüsselung



### Digitale Unterschrift

$$M = D(E(M)) \text{ und } M = E(D(M))$$

# Asymmetrisches Chiffrierverfahren

Mit dem öffentlichen Schlüssel wird chiffriert  
Mit dem geheimen Schlüssel wird dechiffriert

Empfänger (normalerweise Server) gibt öffentlichen Schlüssel  $K_e$  bekannt.

Sender (normalerweise Klient) verschlüsselt mit öffentlichen Schlüssel, nur Empfänger kann Geheimtext mit privaten Schlüssel  $K_d$  entschlüsseln.

# Elektronische Unterschrift

Mit dem geheimen Schlüssel wird chiffriert  
Mit dem öffentlichen Schlüssel wird dechiffriert

Sender gibt öffentlichen Schlüssel  $K_e$  bekannt, verschlüsselt mit geheimen Schlüssel  $K_d$ . Jeder kann Geheimtext entschlüsseln.

Die Nachricht muß von demjenigen stammen, der den öffentlichen Schlüssel  $K_e$  generiert hat.

# **Elektronische Unterschrift**

**Verwendet asymmetrisches Chiffrierverfahren, z.B. RSA. Normalerweise wird mit dem öffentlichen Schlüssel verschlüsselt und mit dem privaten Schlüssel entschlüsselt. Es ist aber genauso möglich, mit dem privaten Schlüssel zu verschlüsseln und mit dem öffentlichen Schlüssel zu entschlüsseln.**

**Personen (Sender) werden durch ihren allgemein zugänglichen öffentlichen Schlüssel eindeutig identifiziert (z.B. durch ein Feld auf ihrer WWW Homepage). Wenn jemand eine Nachricht empfängt, die mit diesem öffentlichen Schlüssel dechiffriert werden kann, dann kann der Absender nur derjenige sein, der den dazugehörigen geheimen Schlüssel besitzt. Im Zweifelsfall muß sich der Sender nachsagen lassen, daß nur er die verschlüsselte Nachricht hat senden können, die mit seinem öffentlichen Schlüssel dechiffriert werden konnte.**

**Der Empfänger hebt als Beweismittel die Nachricht in dechiffrierter und chiffrierter Form auf.**

# Elektronische Unterschrift

**Sender**

**Empfänger**

- 
- 1. berechnet öffentlichen und geheimen Schlüssel**
  - 2. gibt öffentlichen Schlüssel bekannt**
  - 3. sendet Nachricht, die mit dem geheimen Schlüssel chiffriert wurde**
  - 4. dechiffriert Nachricht mit öffentlichem Schlüssel**

**Wer immer einen öffentlichen Schlüssel bekannt gibt, ist als Absender einer Nachricht, die damit dechiffriert werden kann, eindeutig identifiziert.**

# Digitale Unterschrift

**M**        **Message**  
**A**        **Autor**  
**K<sub>d</sub>**      **geheimer Schlüssel des Autors**

**Sende**

**M, A, D<sub>K<sub>d</sub></sub>(M)**

**Falls**

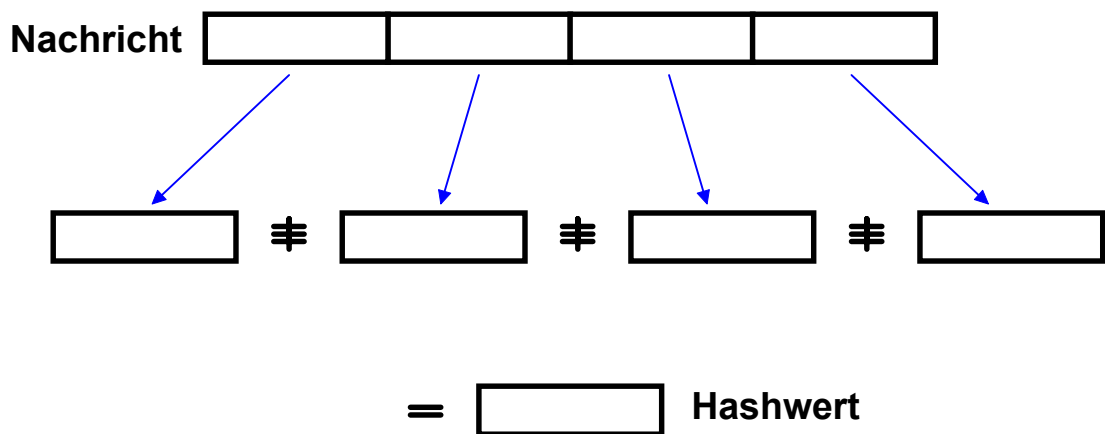
**D<sub>K<sub>d</sub></sub>(M)**

zu lang, verwende Message Digest Funktion  $D(M)$ , auch "secure hash function" genannt.

$D(M)$  ist ein aus  $M$  leicht berechenbarer Wert.

Es muß sichergestellt sein, daß  $D(M) \neq D(M')$  für alle realistischen unterschiedlichen Paare  $M$  und  $M'$ .

# Hashing



**Beispiel: Nachricht in gleichlange Teile zerlegen  
Teile mit Exclusive Oder verknüpfen**

**Es ist leicht, eine andere Nachricht mit dem gleichen Hashwert zu erzeugen**

# Hash Funktionen

Hash Funktionen erzeugen von längeren Nachrichten einen digitalen Fingerabdruck. Sie bilden einen unbeschränkten Wertebereich („alle möglichen Nachrichten“) in einen endlichen Wertebereich ab.

Eine Hash Funktion ist unumkehrbar, wenn es praktisch unmöglich ist, für einen vorgegebenen Hash Wert  $h$  eine Nachricht  $M$  mit  $H(M) = h$  zu finden. Eine Hash Funktion ist zusätzlich kollisionsfrei, wenn es praktisch nicht möglich ist, zu einer gegebenen Nachricht  $M$  eine weitere Nachricht  $M'$  zu berechnen mit  $H(M) = H(M')$ .

Bekannte Hash Funktionen sind:

## **MD5**

128 Bit Hash Wert. Wird u. A. von PGP eingesetzt.

## **SHA(-1)**

Vom USA Geheimdienst entwickelt.

160 Bit Hash Wert

## **RIPE-MD160**

Europäische Entwicklung, für PGP vorgesehen.

160 Bit Hash Wert

## Message Digest Function

Message Digest Functions produzieren eine zufallzahlenähnliche Ziffer begrenzter Länge (z.B. 128 Bit) aus einer Zeichenkette beliebiger Länge.

Der Cyclic Redundancy Check (CRC) oder der „Secure Hash Algorithm“ (SHA) sind Beispiele für Message Digest Functions.

Die Firma RSA Data Security und auch die PGP (Pretty Good Privacy) Software verwenden die „MD5“ Message Digest Function.

MD5 (There is \$1500 in the blue box.) =

**05f8cfc03f4e58cbee731aa4a14b3f03**

MD5 (The meeting last week was swell.) =

**050f3905211cddf36107ffc361c23e3d**

MD5 (There is \$1100 in the blue box.) =

**d6dee11aae89661a45eb9d21e30d34cd**

# **Digest als Digitale Unterschrift PGP Verfahren**

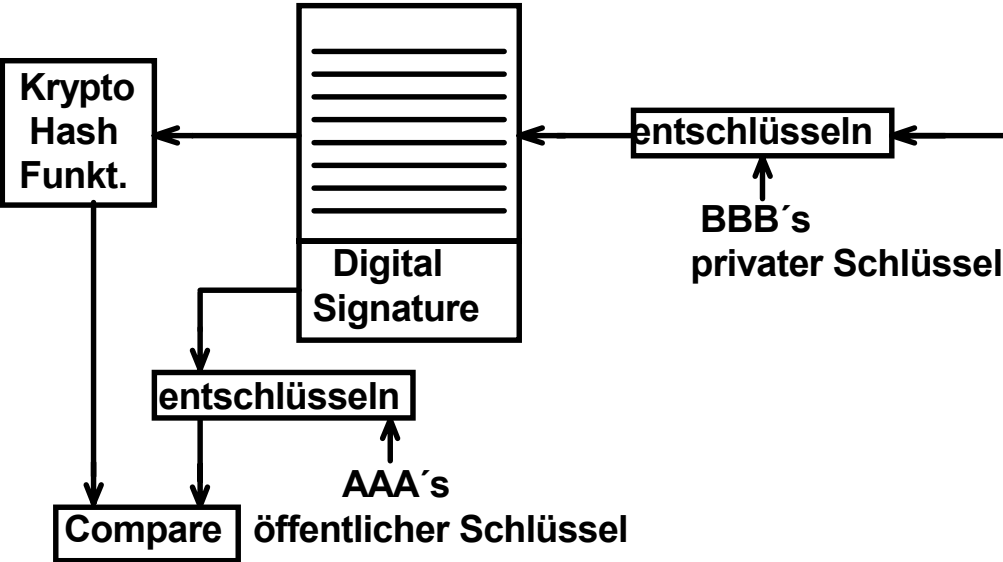
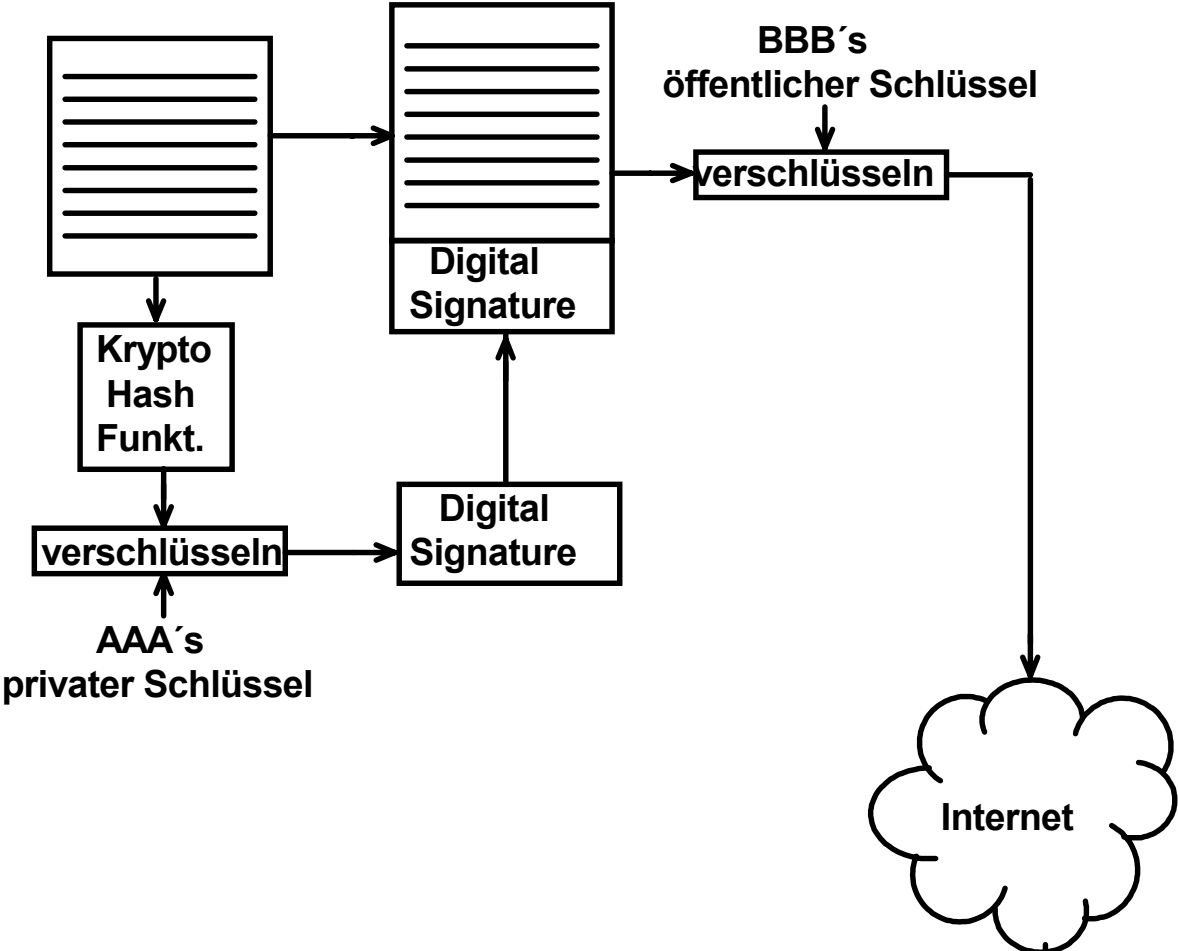
- 1. Hash Verfahren MD5 erzeugt Digest des Dokumentes (MD5) stellt sicher, daß aus dem Digest der Ursprungstext nicht ermittelt werden kann).**
- 2. Digest wird mit privatem Schlüssel des Senders verschlüsselt.**
- 3. Empfänger entschlüsselt Digest mit öffentlichen Schlüssel und verifiziert Übereinstimmung mit dem Dokument.**
- 4. Empfänger weiß nun, daß das Dokument von demjenigen kommen muß, von dem er den öffentlichen Schlüssel erhalten hat.**

**Der Sender kann nicht leugnen, daß er das Dokument geschickt hat, da nur er den Digest mit seinem privaten Schlüssel verschlüsselt haben kann.**

- 5. Problem : Der Sender streitet ab, daß der öffentliche Schlüssel von ihm stammt.**

**Lösung: die öffentlichen Schlüssel werden in einem Trust Center hinterlegt.**

# Sender AAA



# Empfänger BBB

## **Certification Authority (CA) Trust Center (TC)**

**Wie verifiziert der Empfänger, daß der für die elektronische Unterschrift verwendete öffentliche Schlüssel Ke tatsächlich von dem richtigen Sender stammt ?**

**Die Certification Authority garantiert, daß ein öffentlicher Schlüssel zu einer bestimmten Person gehört.**

**Ein Trust Center im EU-Sinne verlangt gleichzeitig die Hinterlegung des privaten Schlüssels. Eine Certification Authority verlangt dies nicht.**

**Zur Authentifizierung eines Kommunikationspartners wendet sich ein Interessent an eine CA. Von dort erhält der Interessent ein mit deren privatem Schlüssel der CA signiertes Zertifikat, was mit dem öffentlichen Schlüssel der CA entschlüsselt werden kann. Im Netscape Browser sind hierfür die öffentlichen Schlüssel verschiedener CA's hinterlegt. 'Das Zertifikat bestätigt den öffentlichen Schlüssel des Kommunikationspartners.**

**Woher weiß man, daß die CA selbst vertrauenswürdig ist? Der X.509 Standard spezifiziert eine „Public Key Infrastruktur“, die aus einer Hierarchie von CA's besteht. In Deutschland ist die Wurzel dieses Baums die Regierungsbehörde für Telekommunikation und Post.**

**Secure Socket Layer (SSL) arbeitet mit den X.509 Standard.**

Support Info-Center Produkte News Partner Jobs Wir über uns

Home Kontakt Site Map

TRUSTCENTER  
KEY TO INTERNET SECURITY

Willkommen bei TC TrustCenter

**Unternehmen**  
E2E, Intranet, PKI, Authentifikation, verbindl. E-Mails, Server-Zertifikate (SSL), Mobile Commerce (WAP, WTLS), Consulting

**Finanzinstitute**  
Identrus, E2E, PKI, Online-Banking, Mobile Solutions (WAP), Consulting, E-Mail

**Behörden**  
E-Government, PKI, SPHINX, elektron. Wahlen, verbindl. E-Mails, Consulting

**Privatanwender**  
vertraul. u. verbindl. E-Mails, elektron. Ausweisung

**Entwickler**  
PKCS#11

Arbeitsplätze powered by DELL

News

STEAG Konzern sichert weltweite Kommunikation mit Security-Lösung der TC TrustCenter AG

▶ Mehr...

Zertifikate von TC TrustCenter jetzt auch durch Schweizer Regierung akzeptiert

▶ Mehr...

▶ Mitteilung EAN

▶ Mitteilung Keyon

▶ Mitteilung ESTV

**SAP**

SAP-Kunden können TC Server-Zertifikate ab sofort auch über den SAP Service Marketplace bestellen. Den bequemen Antragsweg dazu finden Sie unter:

▶ [service.sap.com/tcs-ssl](http://service.sap.com/tcs-ssl)

**TC Server Pool**

Server-Zertifikate im Mehrfachpack:  
Kostengünstig und innerhalb eines Jahres nach Bedarf abrufbar

▶ [TC Server Pool](#)

**pki Challenge**

TC TrustCenter beteiligt sich an EU-finanziertem Projekt zur Verbesserung der Interoperabilität von IT-Komponenten innerhalb Europas.

▶ [www.eema.org/pki-challenge](http://www.eema.org/pki-challenge)

Sicherheitslösung für Continental in den USA realisiert

▶ [Mehr...](#)

COMMERZBANK Deutsche Bank Dresdner Bank HypoVereinsbank

Zertifikats-Services:

▶ [Beantragen](#)

▶ [Suchen](#)

▶ [Verlängern](#)

▶ [Sperren](#)

▶ [CA-Zertifikate](#)

▶ [CRL/Sperrlisten](#)

▶ [LDAP](#)

TrustCenterNews

Ausgabe 10/2002  
Internationale Entwicklungen der digitalen Sicherheit

▶ [Jetzt abonnieren](#)

▶ [Richtlinien](#)

▶ [AGB](#)

## Beispiel: [www.trustcenter.de](http://www.trustcenter.de)

Die Hamburger TC TrustCenter AG ist durch die Regulierungsbehörde für Telekommunikation und Post (RegTP) als Zertifizierungsstelle für digitale Signaturen gemäß dem deutschen Signaturgesetz akkreditiert. Dies ist zusätzlich zu internationalen Gütestandards wie Identrus und SET.

TC TrustCenter AG plant gemeinsam mit der HypoVereinsbank eine multifunktionale EC-Karte zur Anwendung der digitalen Signatur an Privatkunden ausgeben.

## Problem:

### Den öffentlichen Schlüssel zuverlässig einer Person zuordnen

Fritz Müller generiert seinen eigenen öffentlichen und privaten Schlüssel

Name	öffentlicher Schlüssel
Fritz Müller	3F817D.....



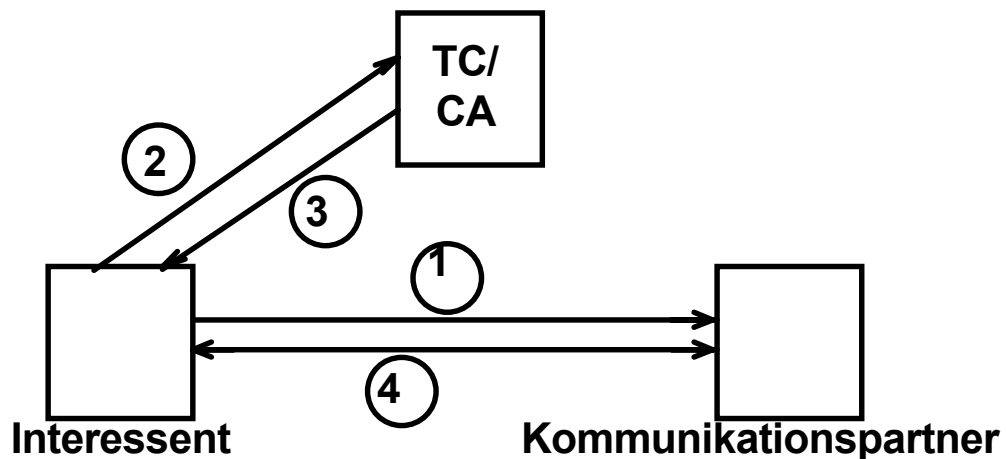
Fritz Müller geht zu seiner Bankfiliale und weist sich mit seinem Personalausweis aus.



Die Bank hinterlegt seinen öffentlichen Schlüssel beim TRUST CENTER

# Trust Center (TC) Certification Authority (CA)

**Annahme: der Kommunikationspartner hat sich zu einem früheren Zeitpunkt zertifizieren lassen. Er hat sich persönlich identifiziert und eine Kopie seines öffentlichen Schlüssels hinterlegt.**



- 1. Interessent erfragt vom Kommunikationspartner dessen öffentlichen Schlüssel**
- 2. Anfrage an TC/CA: Ist der Kommunikationspartner derjenige, der er vorgibt zu sein**
- 3. Bestätigung durch TC/CA mit signiertem Zertifikat**
- 4. Transaktion mit elektronischen Unterschriften**

**Auf der Home Page von Fritz Müller  
steht sein öffentlicher Schlüssel**

<b>Name</b>	<b>öffentlicher Schlüssel</b>
<b>Fritz Müller</b>	<b>3F817D.....</b>



**Der Empfänger erfragt beim TRUST CENTER den  
öffentlichen Schlüssel von Fritz Müller**



**Die Antwort ist verschlüsselt mit dem privaten Schlüssel  
des TRUST CENTER**



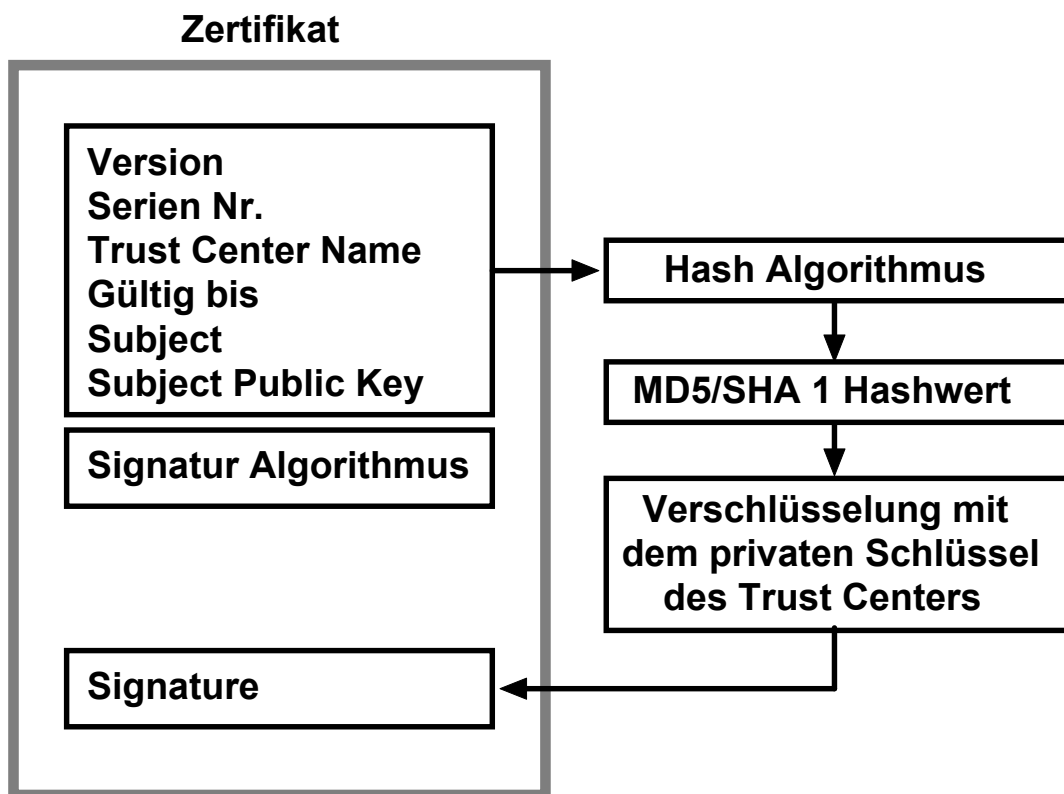
**Certificate**



**wird vom Empfänger mit dem (allseits bekannten)  
öffentlichen Schlüssel des TRUST CENTER entschlüsselt**



**Jawohl, 3F817D..... ist der öffentliche Schlüssel  
von Fritz Müller**



## Aufbau eines X.509 Zertifikates

Definiert in der Abstract Syntay Notation One (ASN.1) Derzeitiger Standard (2003) ist Version X.509 v 3

# Authentifizierung

**authentifizieren:** die Echtheit bezeugen, beglaubigen  
Verifikation der Identität

**authentisieren:** glaubwürdig, rechtsgültig machen

## Ein-Weg-Authentifizierung (klassisch)

**Der Benutzer muß seine Identität gegenüber dem System beweisen, aber nicht umgekehrt**

## Zwei-Wege-Authentifizierung

**Auch der Server muß sich gegenüber dem Benutzer ausweisen**

## Nachricht

**Digitale Unterschrift**

# **Ein-Weg-Mechanismus: Passwort**

## **Risiken**

**Beim Eingeben: Über die Schulter sehen**

**Beim Speichern im Rechner:**

**Passwortdatei im Rechner stellt ein hohes  
Sicherheitsrisiko dar**

**Verschlüsselte Speicherung: Durch „Directory Guesses“  
können Passwörter dennoch schnell geraten werden**

**Aufbewahrung vom Benutzer:**

**Wenn vom Benutzer gewählt - oft leicht zu erraten  
wenn vom System gewählt - Benutzer schreibt auf**

**Änderungen durch den Benutzer: gefährlich, wenn nicht  
der gesamte Datenpfad sicher ist.**

**Besser sind biometrische Verfahren:**

**Fingerabdruck**

**Sprachanalyse**

**Augenhintergrund (Iris Scan)**

**Dynamik der Unterschrift**

# **Biometrische Authentifizierung**

## **Physiologische Lösungen**

**Fingerabdruck  
Handflächenerkennung  
Gesichtserkennung  
Iris Erkennung  
Retina Erkennung**

**unauffällig  
sehr sicher, sehr teuer**

## **Zukünftige Physiologische Lösungen**

**Ohr Geometrie  
Fingerkuppenaufbau  
Körpergeruch**

## **Verhaltenslösungen**

**Spracherkennung  
Handschriftendynamik  
Akustische Erkennung der Handschrift**

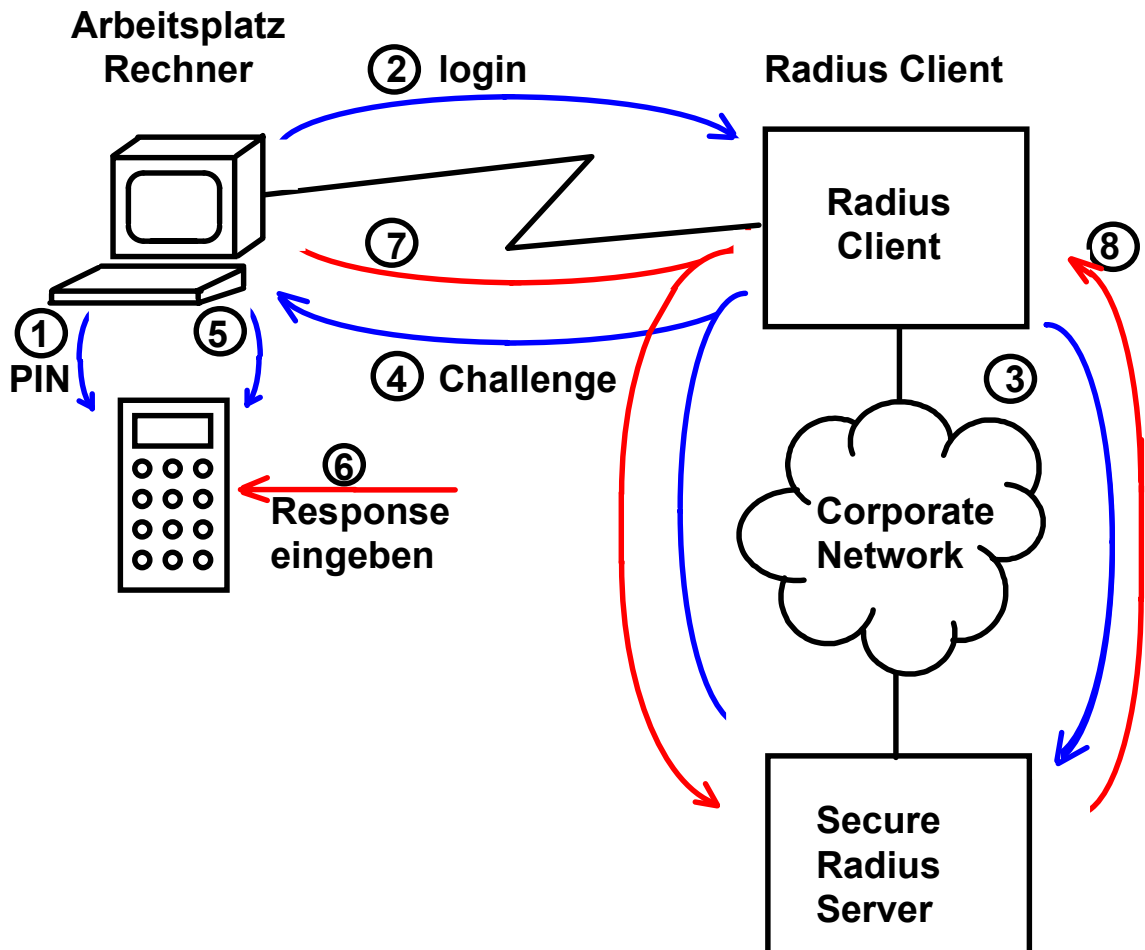
## **Challenges**

**Der Server fragt beim Klienten Informationen ab, von der er weiß, dass nur der Klient über diese Informationen verfügt.**

## **Beispiel**

**„Nenne mir die Namen von 3 Schulkameraden, die mit Dir im 3. Grundschuljahr in die gleiche Klasse gingen“.**

- 3 initiate Authentication Process
- 7 Response übertragen
- 8 authorise access



## Token based Authentication System Challenge - Response Verfahren

**Radius Remote Authentication Dial-In User Service**

# **Zugriffsberechtigung in verteilten Systemen**

**Kernel des Servers überprüft Zugriffsberechtigung des Klienten.**

**Klient und Server müssen gemeinsam Passwort und kryptographischen Schlüssel kennen.**

**Für eine erstmalige Kommunikation muss Schlüssel und Passwort auf einem sicheren Weg an beide Partner übermittelt werden. Public Key Verfahren ist unzureichend, da ein Einbrecher sich als legitimer Benutzer tarnen kann.**

**Authentication Server.**

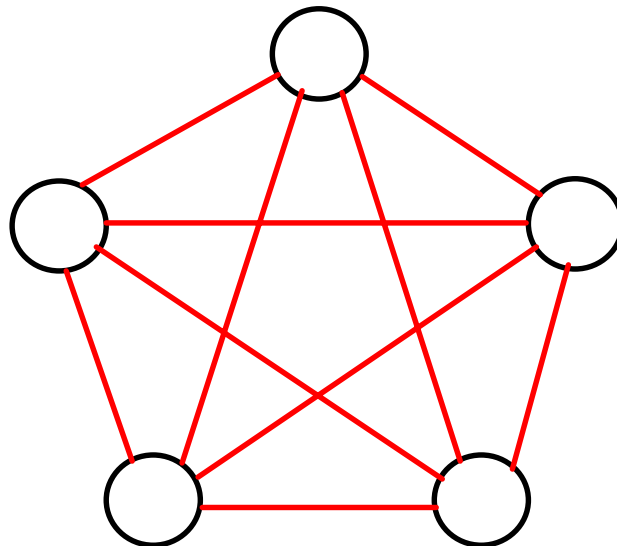
# Problem

In einem Netz mit n Rechnern müssen

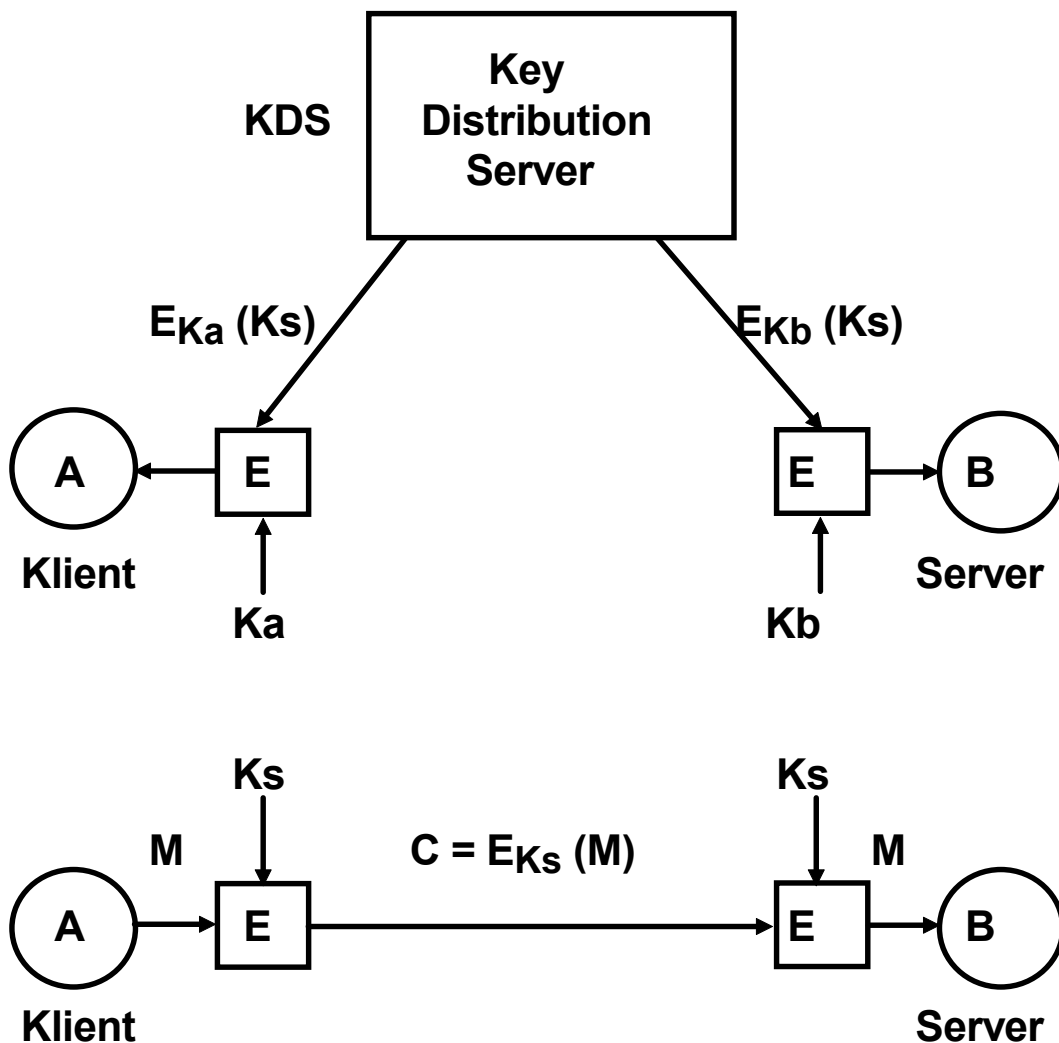
$$(n-1) + (n-2) \dots 1 = 0,5 n (n-1)$$

Schlüssel verteilt werden.

Bei Verwendung eines Key-Distribution-Servers  
genügen n Schlüssel.



für 5 Rechner sind  $0,5 \times 5 \times 4 = 10$  Schlüssel erforderlich



## Schlüsselverteilung

Schlüssel werden durch Schlüsselverteilungsserver KDS verteilt.

Jeder Partner hat einen eigenen Schlüssel  $K_a$  bzw.  $K_b$ , der nur zum Nachrichtenaustausch mit KDS verwendet wird. Zu Beginn einer Nachrichtenübertragung erhalten beide Partner vom KDS einen Sitzungsschlüssel  $K_s$ .

Die Philosophie ist, daß es einem Eindringling nicht möglich ist, genügend viel Geheimtext zu sammeln um  $K_s$  zu entschlüsseln.

# **Schlüsselverteilungsserver**

## **Key Distribution Server**

**Session Key: Schlüssel, der für die Kommunikation zwischen 2 Partnern für eine begrenzte Dauer benutzt wird.**

**Annahme: Teilnehmer A und B haben noch nie miteinander kommuniziert.**

**Beide verfügen über einen (individuellen) privaten Schlüssel, der nur für die Kommunikation mit einem Key Distribution Server (KDS) verwendet wird. Diese Schlüssel sind in dem KDS hinterlegt.**

**Teilnehmer A erhält von KDS eine Kopie des Session Key, der mit dem Privaten Key von A verschlüsselt ist.**

**Teilnehmer B erhält von KDS eine Kopie des Session Key, der mit dem Privaten Key von B verschlüsselt ist.**

**Wird von der USA Regierung eingesetzt.**

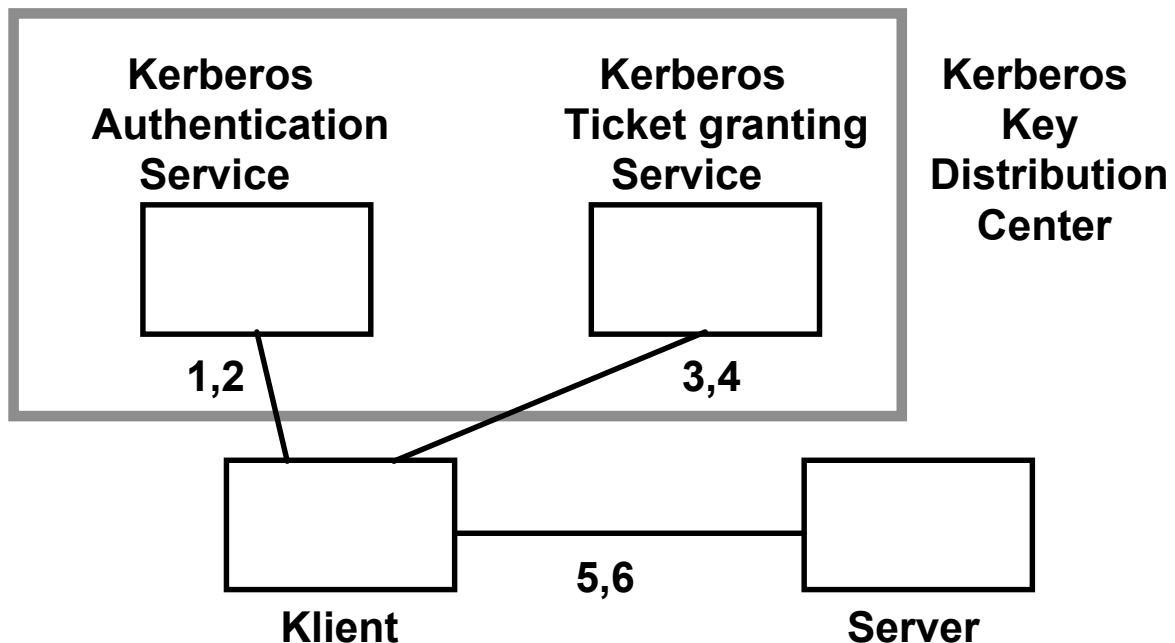
**Problem (tatsächlich vorgekommen): Mitarbeiter verkauft private Schlüssel, die in dem KDC hinterlegt sind.**

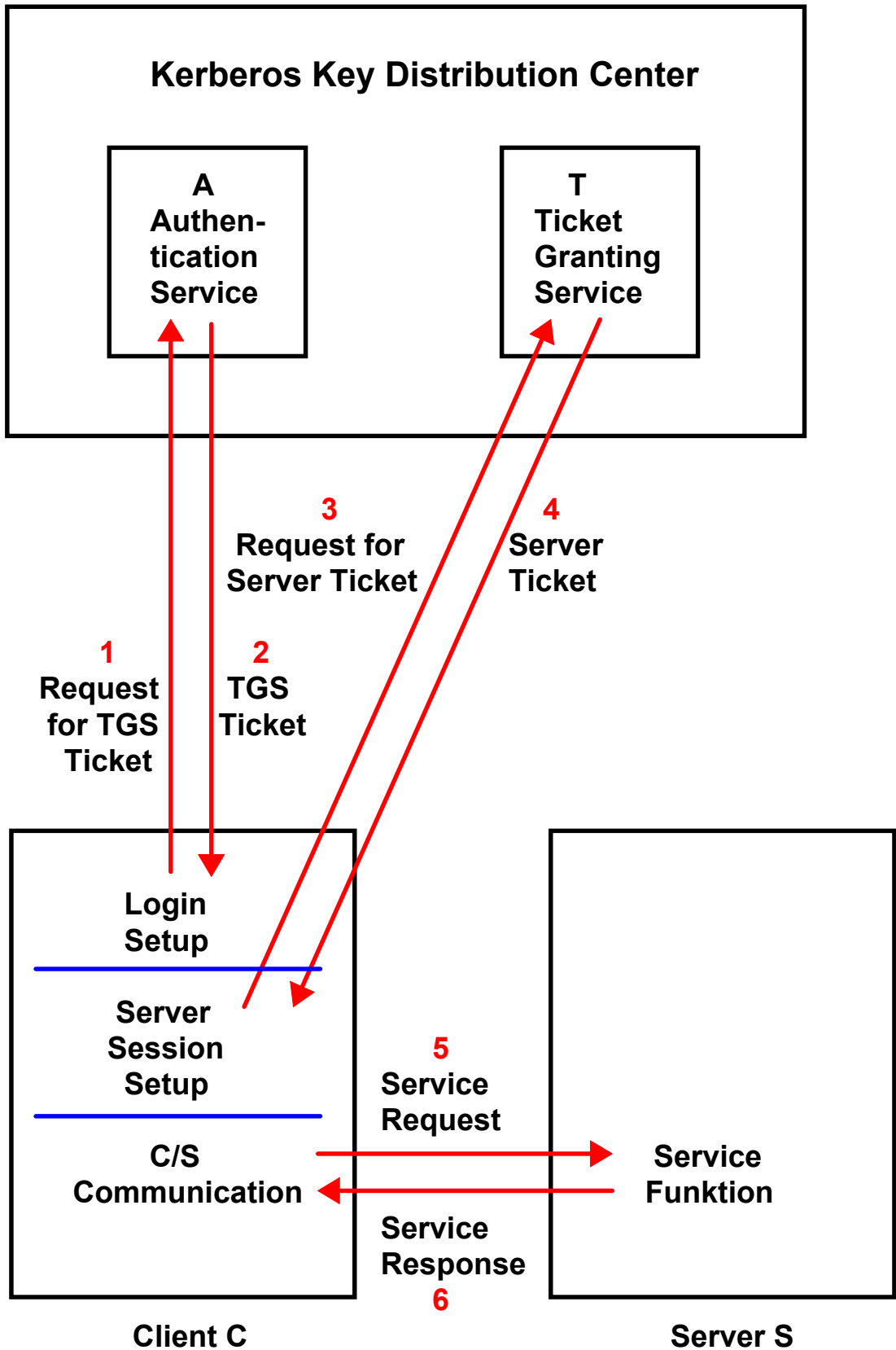
# Kerberos

**Authentifizierungs- und Schlüsselverteildienste für partizipierende Teilnehmer.**

**Nicht jeder Knotentechniker in einem Netz muß die Kerberos Dienste in Anspruch nehmen.**

**Jeder partizipierende Teilnehmer hat einen individuellen Schlüssel, der nur ihm und Kerberos bekannt ist, und der nur für den Nachrichtenaustausch mit Kerberos verwendet wird.**





## Kerberos System Architektur

# Ticket

Ein von Kerberos ausgegebenes Kennwort, welches bestätigt, daß ein spezifischer Benutzer kürzlich authentifiziert wurde.

# Nonce

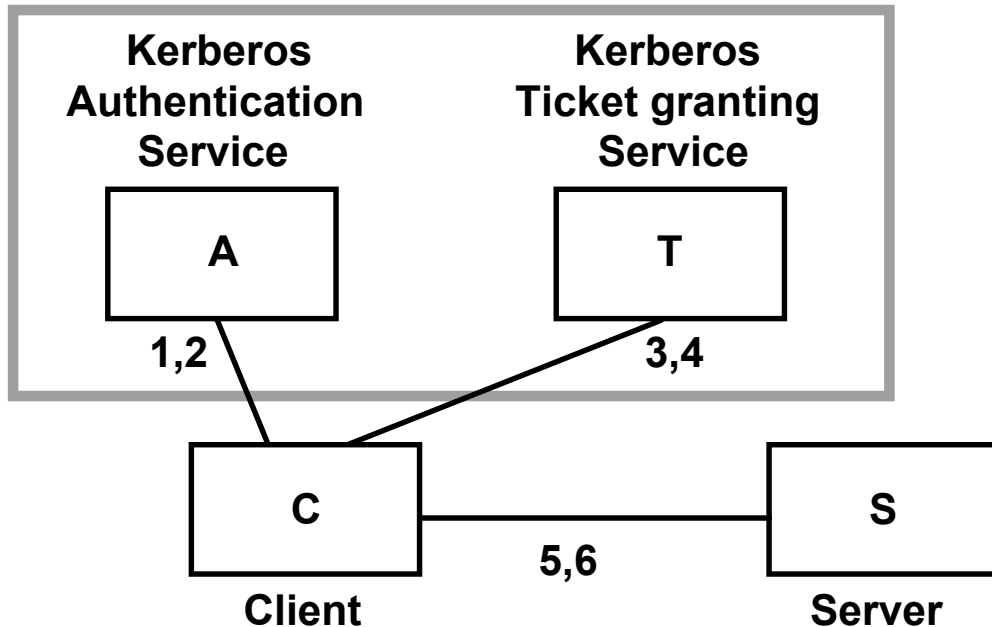
Ein "Nonce" (not but once) ist ein Integer, der die Frische einer Nachricht bestätigt.

Ein Nonce kann als Paar { Datum, Uhrzeit } dargestellt werden.

# Sitzungsschlüssel

Ein von Kerberos per Zufallszahlengenerator herausgegebener (in der Regel symmetrischer) Schlüssel, der von 2 Partnern für eine begrenzte Zeit zum kryptographischen Nachrichtenaustausch verwendet wird.

## Kerberos Protokoll (2)

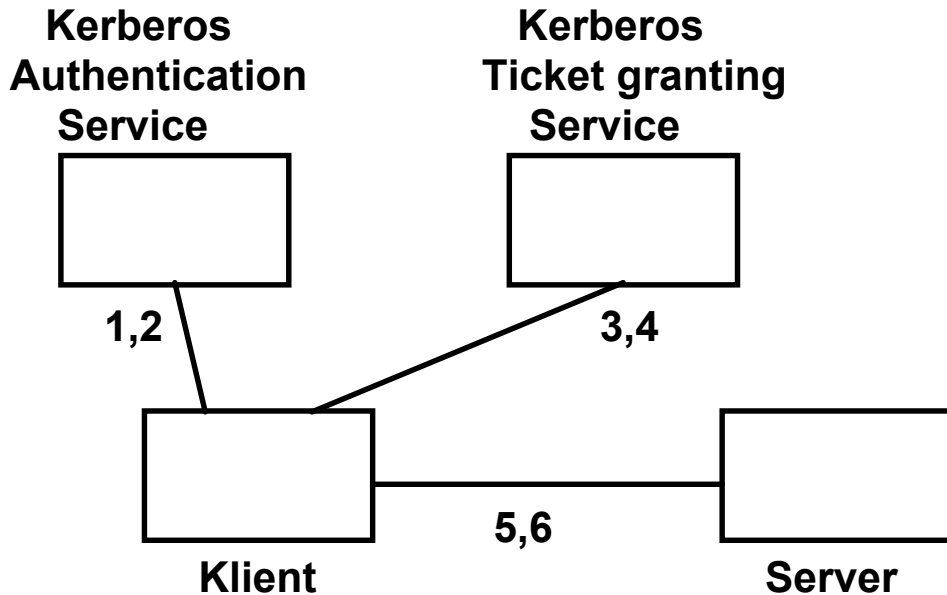


**C** Client  
**A** Kerberos Authentication Service  
**T** Kerberos Ticket Granting Service  
**S** Server

**K<sub>C</sub>** privater Schlüssel von C, geht nie über das Netz  
**K<sub>T</sub>** privater Schlüssel von T, geht nie über das Netz  
**K<sub>S</sub>** privater Schlüssel von S, geht nie über das Netz  
**K<sub>CT</sub>** in der Lebensdauer begrenzter Schlüssel für die Kommunikation zwischen C und T  
**K<sub>CS</sub>** in der Lebensdauer begrenzter Schlüssel für die Kommunikation zwischen C und S

**R** Authentication Request von C  
**ticket (x,y)** Authentication Bestätigung für die Kommunikation von x nach y  
**n** nonce

## Kerberos Protokoll (3)



1.  $C \Rightarrow A$        $C, T, n$
2.  $A \Rightarrow C$        $\{ K_{CT}, n \}_{K_C}, \{ \text{ticket}(C,T) \}_{K_T}$   
 $A \Rightarrow T$        $K_{CT}$
3.  $C \Rightarrow T$        $\{ R \}_{K_{CT}}, \{ \text{ticket}(C,T) \}_{K_T}, S$
4.  $T \Rightarrow C$        $\{ K_{CS} \}_{K_{CT}}, \{ \text{ticket}(C,S) \}_{K_S}$   
 $T \Rightarrow S$        $\{ K_{CS} \}_{K_S}$
5.  $C \Rightarrow S$        $\{ R \}_{K_{CS}}, \{ \text{ticket}(C,S) \}_{K_S}, \text{Request}, n$
6.  $S \Rightarrow C$        $\{ n \}_{K_{CS}}, \text{Response}$

# Kerberos Protokoll

1. C ⇒ A

C, T, n

A besitzt private Schlüssel von C und T. Gehen nie über das Netz.

2. A ⇒ C

$K_{CT}, n$

ticket (C,T)

A ⇒ T

$K_{CT}$

3. C ⇒ T

R

ticket (C,T)

S, n

4. T ⇒ C

$K_{CS}, n$

ticket (C,S)

T ⇒ S

$K_{CS}$

5. C ⇒ S

R

ticket (C,S)

Request, n

6. S ⇒ C

n

Response

$K_C$



privater Schlüssel von C

$K_T$



privater Schlüssel von T

$K_S$



privater Schlüssel von S

$K_{CT}$



Schlüssel für die Kommunikation zwischen C und T

$K_{CS}$



Schlüssel für die Kommunikation zwischen C und S

# Kerberos Arbeitsweise

## Schritt 1

Klient sendet unverschlüsselt seinen Benutzernamen plus Nonce an den Authentication Service A des Kerberos Key Distribution Center KDC

## Schritt 2

KDC sendet an Klient Nachricht bestehend aus 3 Teilen:

1. Schlüssel  $K_{CT}$  für Kommunikation mit Ticket Granting Service T
2. Nonce Bestätigung
3. Ticket welches der Authentifizierung des Benutzers gegenüber T dient

Teil 1 und 2 sind mit dem Paßwort des Benutzers verschlüsselt. Teil 3 ist mit einem geheimen, dem Benutzer nicht bekannten Paßwort von T verschlüsselt.

Die Nachricht von T wird als „Challenge“ bezeichnet, weil der Klient nur bei Kenntnis des Benutzer Paßwortes damit etwas anfangen kann. Das Benutzer Paßwort selbst wird nie über das Netz übertragen.

## Schritt 3

Klient fordert von T ein Ticket für die Nachrichtenverbindung mit S an. Diese Nachricht ist mit  $K_{CT}$  verschlüsselt.

## Schritt 4

T erstellt Schlüssel  $K_{CS}$  für Kommunikation zwischen C und S.  $K_{CS}$  wird nach einer begrenzten Zeit ungültig.

## **Probleme mit Kerberos**

**Kerberos schützt nicht gegen Änderungen im Betriebssystem des Arbeitsplatzrechners.**

**Ein Einbrecher kann die System Software abändern, so daß jede Benutzername/Paßwort Kombination automatisch aufgezeichnet oder an eine dritte Maschine gesendet wird. Der Einbrecher kann sich so als ein legitimer Benutzer tarnen (Trojanisches Pferd).**

**Kerberos benötigt einen sicheren Kerberos Server. Auf dem Kerberos Server sollte keine andere Anwendung laufen. Der Server muß unter Verschuß in einem physikalisch gesicherten Raum untergebracht werden.**