

## *Lecture 5*

# Cryptography II

Information & Communication Security  
(WS 2010/11)

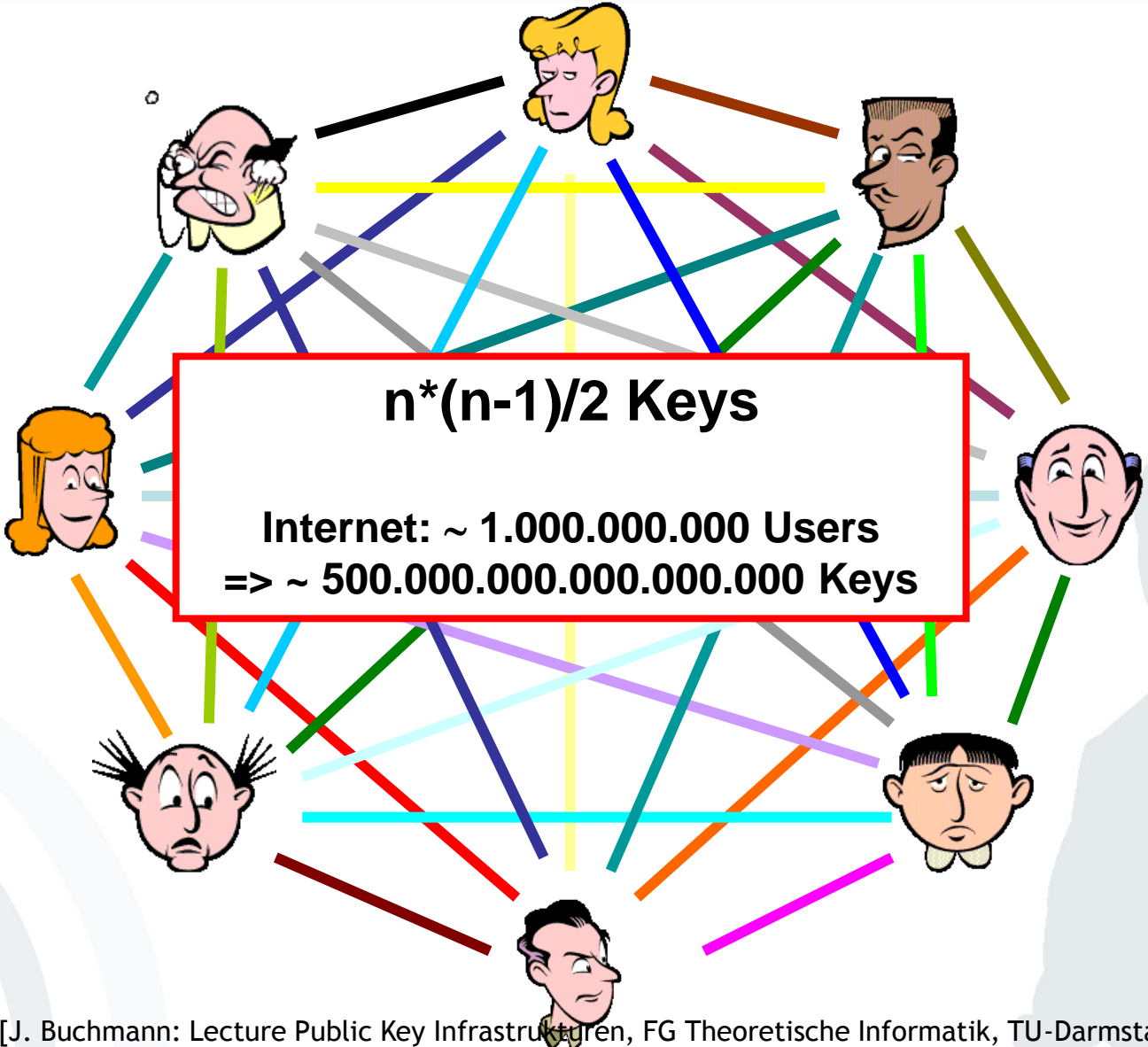
Prof. Dr. Kai Rannenberg

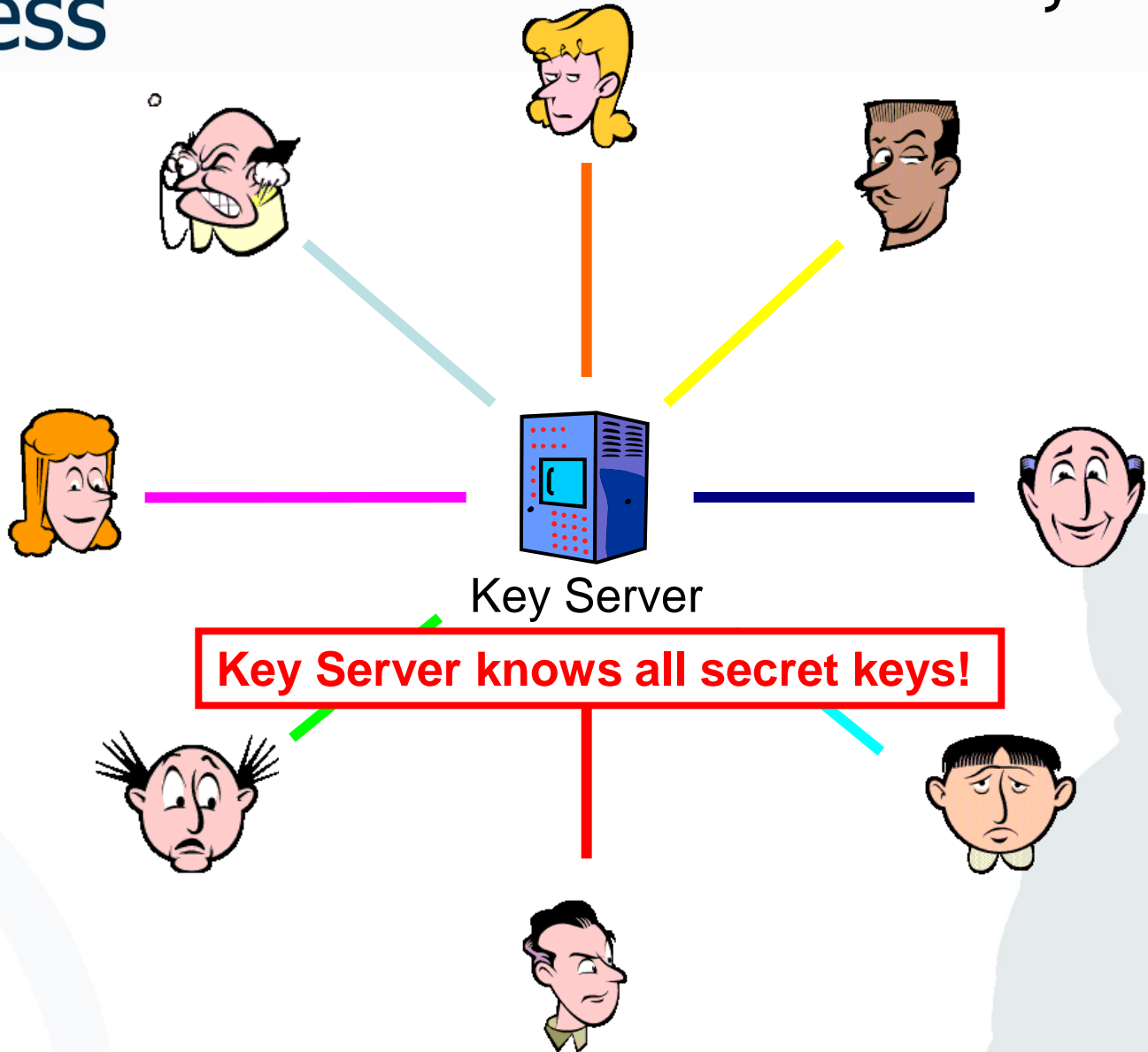
T-Mobile Chair of Mobile Business & Multilateral Security  
Goethe-University Frankfurt a. M.

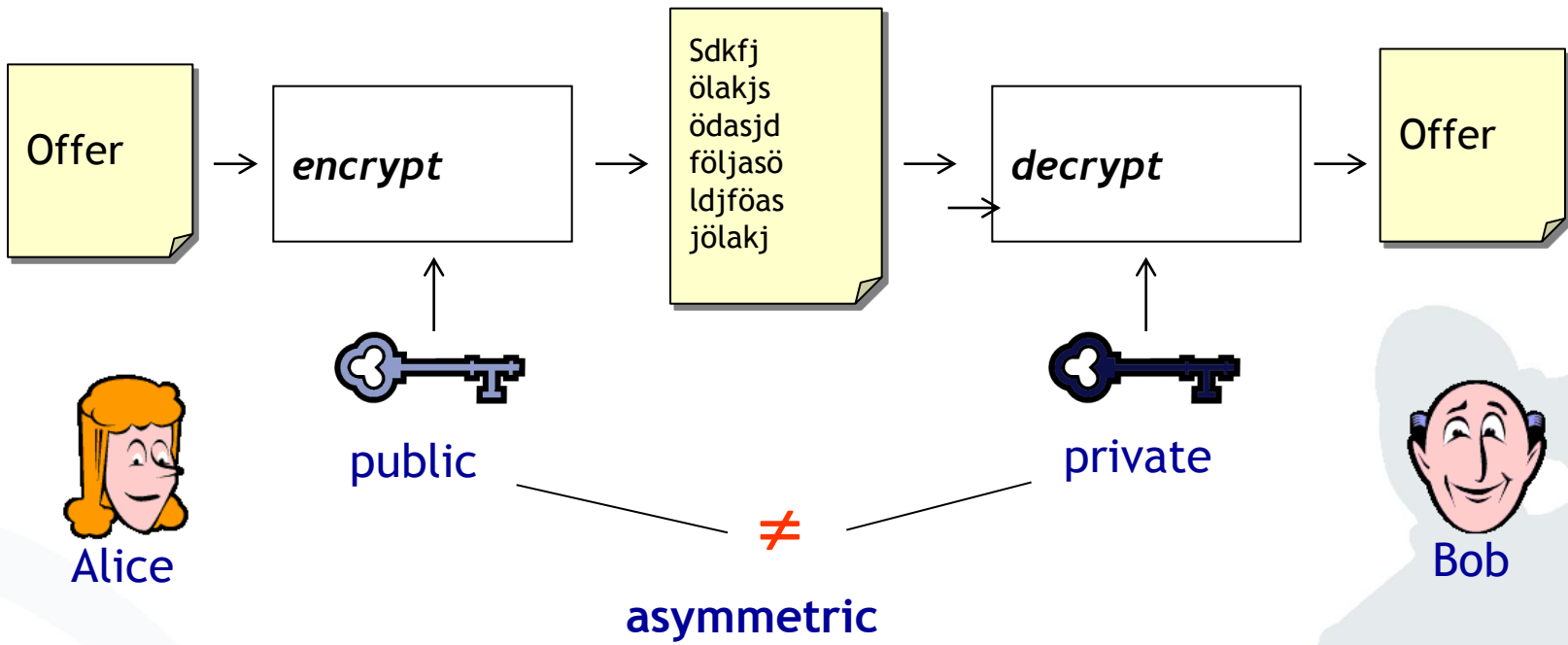


- Introduction
- Classical cryptosystems
- Public key cryptography
  - General concept
  - Algorithms
  - Hybrid systems
  - Key management
  - Example: PGP

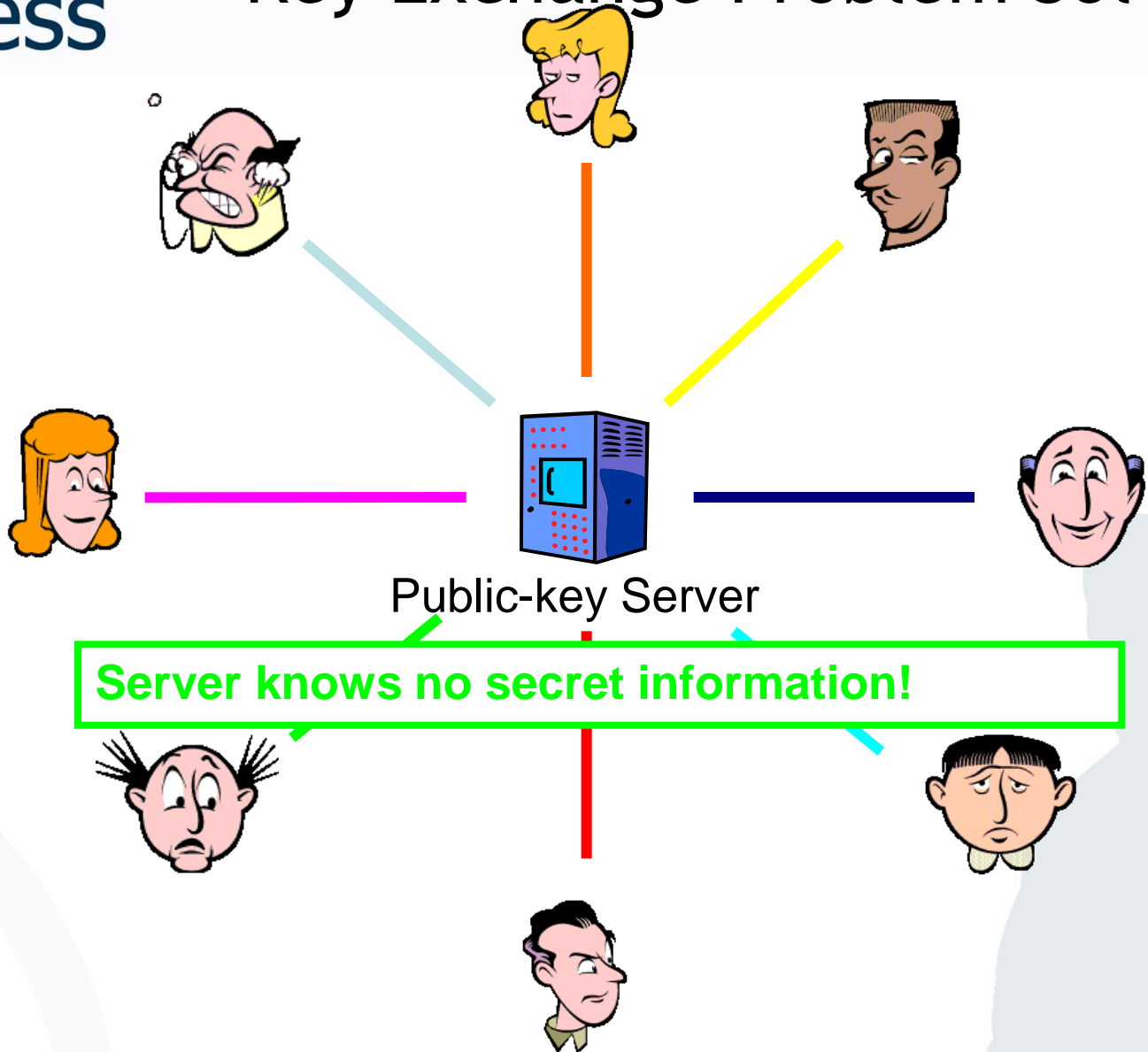
# Problems of Symmetric Cryptosystems: Key Exchange







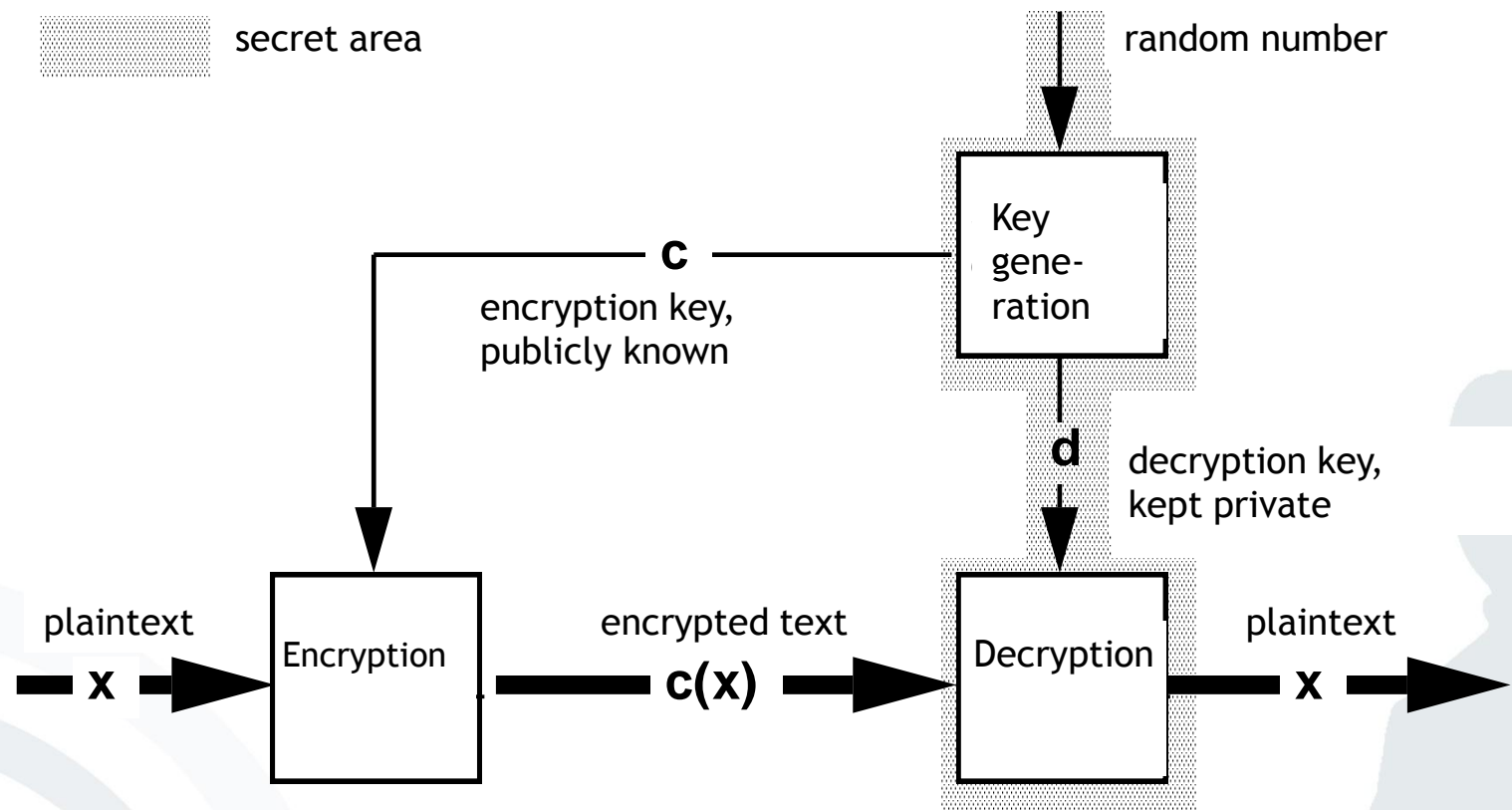
# Key Exchange Problem Solved!



- Introduction
- Classical cryptosystems
- Public key cryptography
  - General concept
  - Algorithms
  - Hybrid systems
  - Key management
  - Example: PGP

- Use of key pairs instead of one key:
  - **Public key** is solely for encryption.
  - Key text can only be decrypted with the corresponding **private (undisclosed) key**.
- Private key cannot be calculated from the public key.
- The public key can be distributed freely, even via insecure ways (e.g. directory (*public key crypto system*)).
- Messages are encoded via the public key of the addressee.
- Only the addressee holds the private key for decoding.

# Asymmetric Encryption Systems



*Box with slot, access to messages only with a key*

- Introduction
- Classical cryptosystems
- Public key cryptography
  - General concept
  - Algorithms
  - Hybrid systems
  - Key management
  - Example: PGP

- RSA
  - Rivest, Shamir, Adleman, 1978
  - Based on the assumption that the factorization of the product of two (big) prime numbers ( $p \cdot q$ ) is “difficult” (product is the public key)
  - Key lengths often 1024 bit; recommended 2048 or 4096 bit
- Diffie-Hellman
  - Diffie, Hellman, 1976
  - First patented algorithm with public keys
  - Allows the exchange of a secret key
  - Based on the “difficulty” of calculating discrete logarithms in a finite field

- Introduction
- Classical cryptosystems
- Public key cryptography
  - General concept
  - Algorithms
    - RSA
  - Hybrid systems
  - Key management
  - Example: PGP

- To encrypt a message  $M$ , using a public key  $(e, n)$ , proceed as follows ( $e$  and  $n$  are a pair of positive integers):
  - First represent the message as an integer between 0 and  $n-1$  (break long messages into a series of blocks, and represent each block as such an integer).
  - Then encrypt the message by raising it to the  $e^{\text{th}}$  power modulo  $n$ .
  - The result (the ciphertext  $C$ ) is the remainder of  $M^e$  divided by  $n$ .
  - The encryption key is thus the pair of positive integers  $(e, n)$ .

- To decrypt the ciphertext, raise it to another power  $d$ , again modulo  $n$ .
- The decryption key is the pair of positive integers  $(d, n)$ .
- Each user makes his encryption key public, and keeps the corresponding decryption key private.

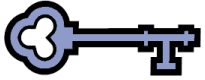
## RSA Encryption/Decryption Summary

- $C \equiv E(M) \equiv M^e \pmod{n}$ ,  
for a message  $M$
- $M \equiv D(C) \equiv C^d \pmod{n}$ ,  
for a ciphertext  $C$

- You first compute  $n$  as the product of two primes  $p$  and  $q$ .
- $n=p*q$
- These primes are very large “random” primes.
- Although you will make  $n$  public, the factors  $p$  and  $q$  will be effectively hidden from everyone else due to the enormous difficulty of factoring  $n$ .
- This also hides the way, how  $d$  can be derived from  $e$ .

- You then choose an integer  $d$  to be a large, random integer which is relatively prime to  $(p-1)*(q-1)$ .
- That is, check that  $d$  satisfies:
  - The greatest common divisor of  $d$  and  $(p-1)*(q-1)$  is 1.
  - $\text{gcd}(d, (p-1)*(q-1))=1$

- The integer  $e$  is finally computed from  $p, q$ , and  $d$  to be the “multiplicative inverse” of  $d$ , modulo  $(p-1)*(q-1)$ .
- Thus we have
$$e*d \equiv 1 \pmod{(p-1)*(q-1)}.$$



Public  
(e,n)



Private  
(d,n)



Alice

- Let  $p=7$  and  $q=11$ .
- Then  $n=77$ .
- Alice chooses  $d=53$ , so  $e=17$ .
- $\gcd(d, (p-1)*(q-1)) = \gcd(53, (7-1)*(11-1)) = \gcd(53, 60) = 1$
- $e*d \bmod (p-1)*(q-1) = 901 \bmod 60 = 1$

- Bob wants to send Alice the message „Hello World“
- Each plaintext character is represented by a number between 00(A) and 25 (Z).
- Therefore, we have the plaintext as:  
07 04 11 11 14 26 22 14 17 11 03

Hello  
World



Bob

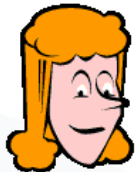
- Using Alice's public key the ciphertext is:
  - $07^{17} \bmod 77 = 28$
  - $04^{17} \bmod 77 = 16$
  - $11^{17} \bmod 77 = 44$
  - ...
  - $03^{17} \bmod 77 = 75$
- Or 28 16 44 44 42 38 22 42 19  
44 75

Hello  
World



Bob

28 16 44 44  
42 38 22  
42 19 44 75



Alice

- Alice decrypts the ciphertext by calculating:
  - $28^{53} \bmod 77 = 07$
  - $16^{53} \bmod 77 = 04$
  - $44^{53} \bmod 77 = 11$
  - ...
  - $75^{53} \bmod 77 = 03$
- Or: 07 04 11 11 14 26 22 14  
17 11 03 = “Hello World”

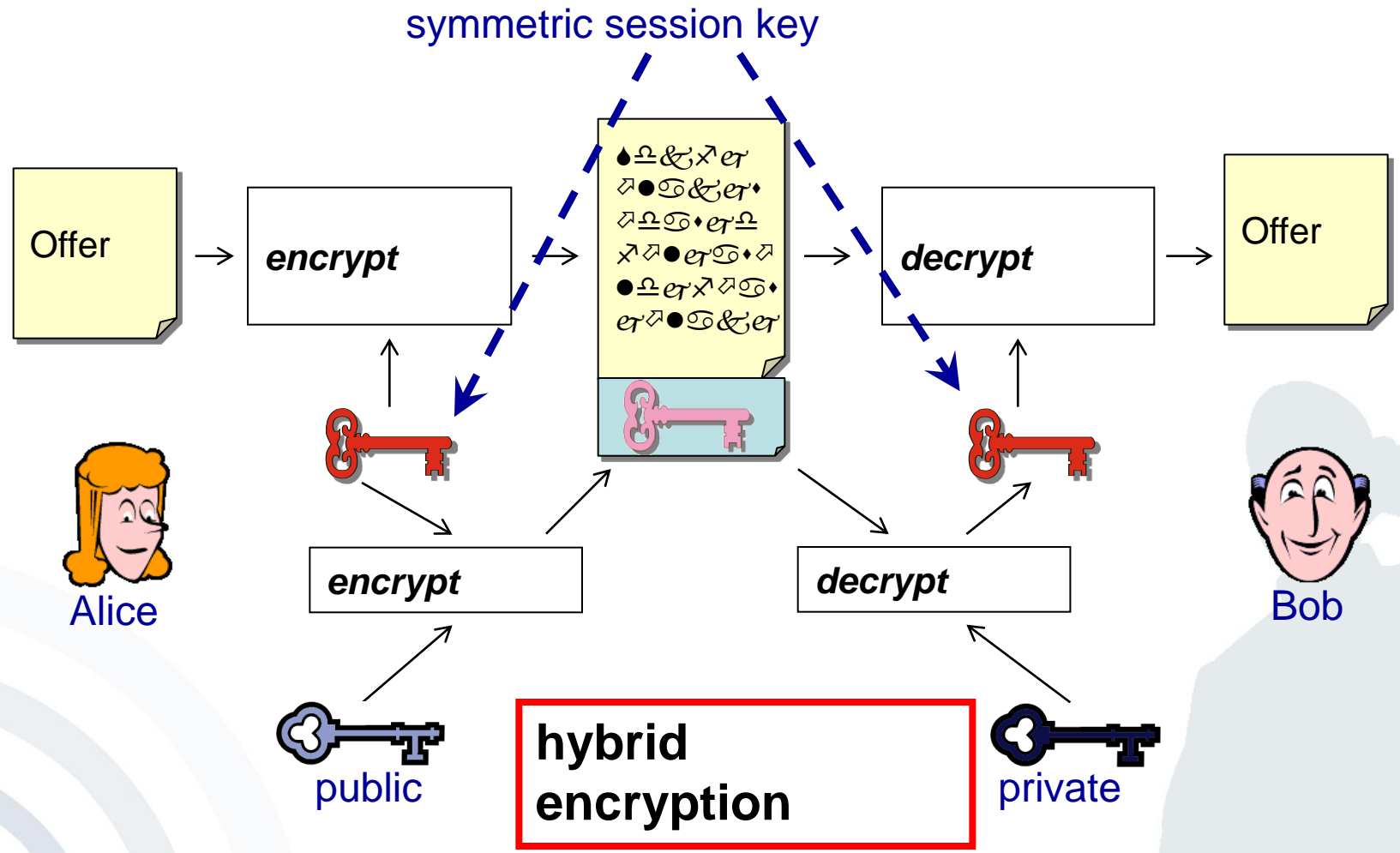
- Introduction
- Classical cryptosystems
- Public key cryptography
  - General concept
  - Algorithms
  - Hybrid systems
  - Key management
  - Example: PGP

Algorithm	Performance*
El Gamal	1826 s
RSA	16 s

**Disadvantage:** Complex operations  
with very big numbers

⇒ **Algorithms are very slow**

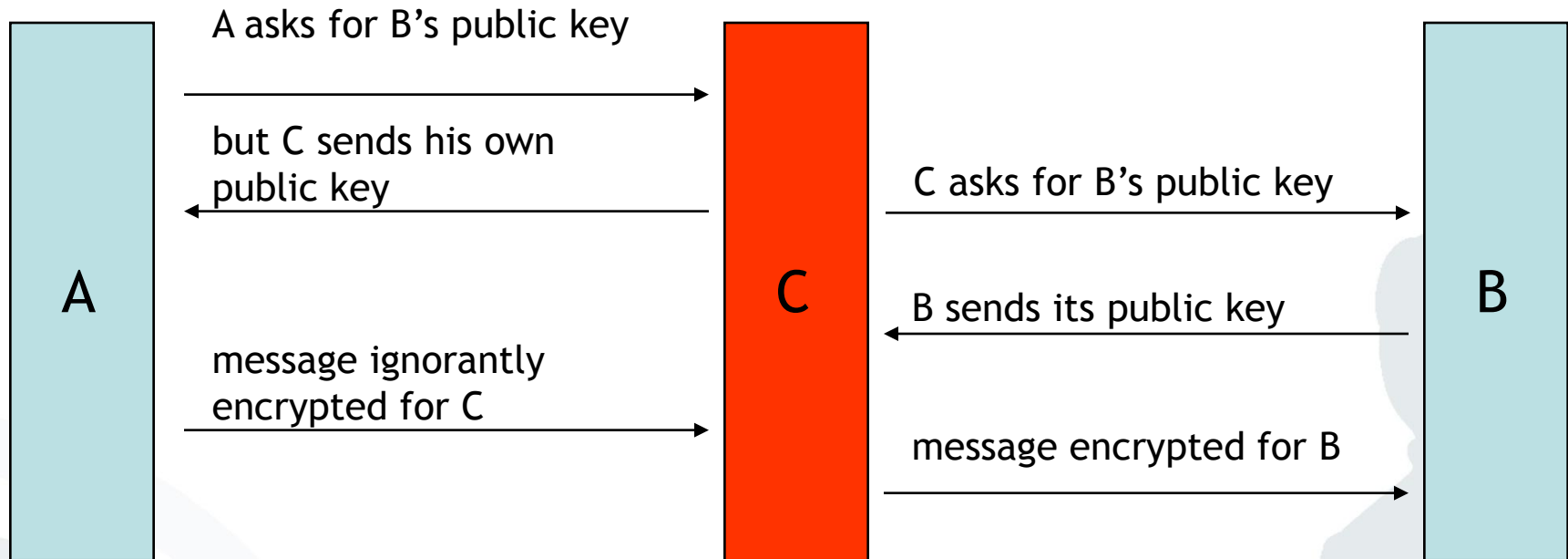
\*) Encryption of 1 MB-blocks with an Athlon 1GHz processor



[based on: J. Buchmann 2005: Lecture Public Key Infrastrukturen, FG Theoretische Informatik, TU-Darmstadt]

- Introduction
- Classical cryptosystems
- Public key cryptography
  - General concept
  - Algorithms
  - Hybrid systems
  - Key management
  - Example: PGP

## “Man in the middle attack”



- Keys are certified: a 3<sup>rd</sup> person/institution confirms (with its digital signature) the affiliation of the public key to a person.

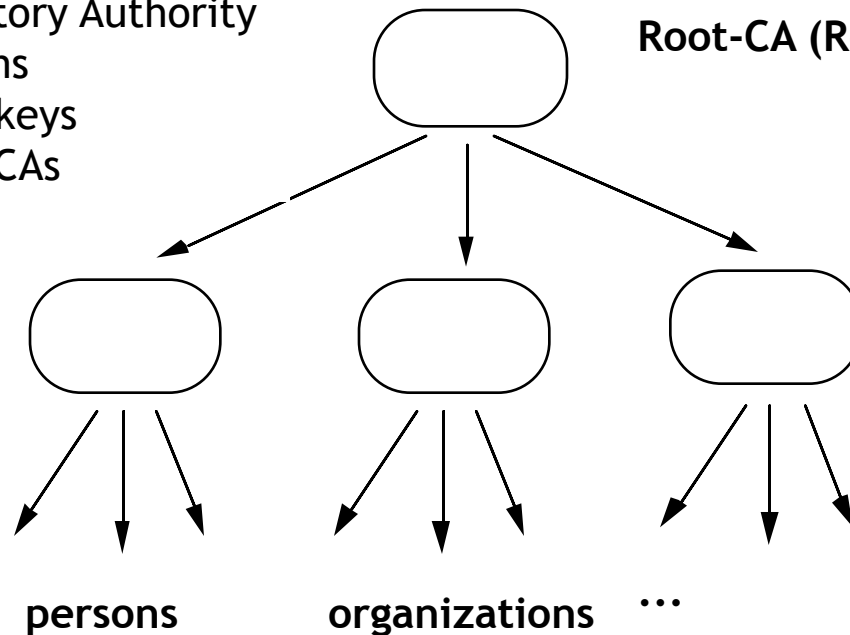
- B can freely distribute his own public key.
- But: Everybody (e.g. C) could distribute a public key and claim that this one belongs to B.
- If A uses this key to send a message to B, C will be able to read this message!
- Thus:  
How can A decide if a public key was really created and distributed by B without asking B directly?
- ➔ Keys get **certified**, i.e. a third person/institution confirms with its (digital) signature the **affiliation of a public key to entity B**.
- ➔ Public Key Infrastructures (PKIs)

Three types of organization for certification systems (PKIs?):

- **Central Certification Authority (CA)**
  - A single CA, keys often integrated in checking software
  - eExample: older versions of Netscape (CA = Verisign)
- **Hierarchical certification system**
  - CAs which in turn are certified by “higher” CA
  - Examples: PEM, TeleTrust, infrastructure according to Signature Law
- **Web of Trust**
  - Each owner of a key may serve as a CA.
  - Users have to assess certificates on their own.
  - Example: PGP (but with hierarchical overlay system)

Regulatory Authority  
confirms  
public keys  
of the CAs

Root-CA (Regulatory Authority)

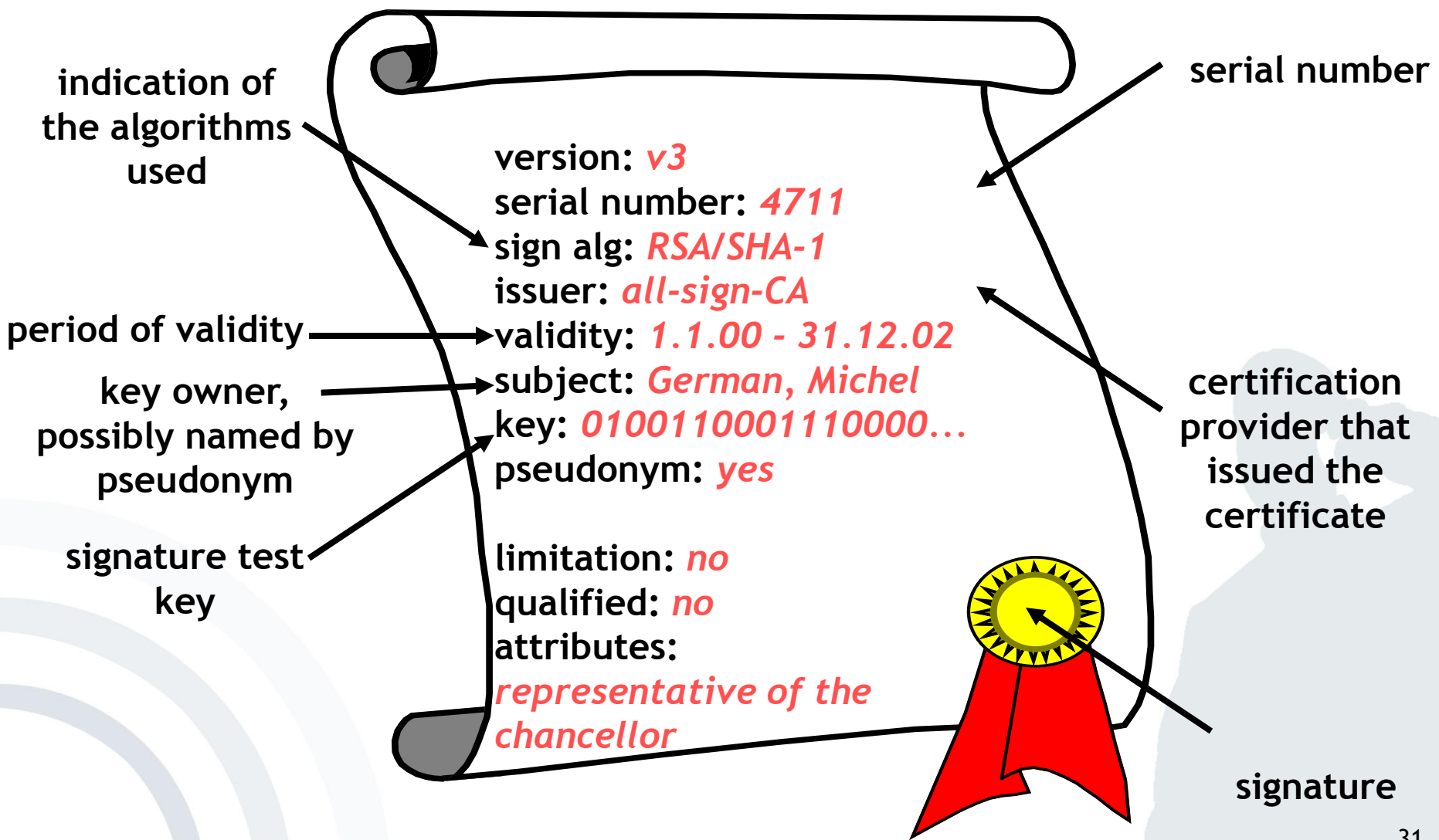


Certification  
Authorities (CA)

TeleSec, D-Trust,  
TC TrustCenter, ...

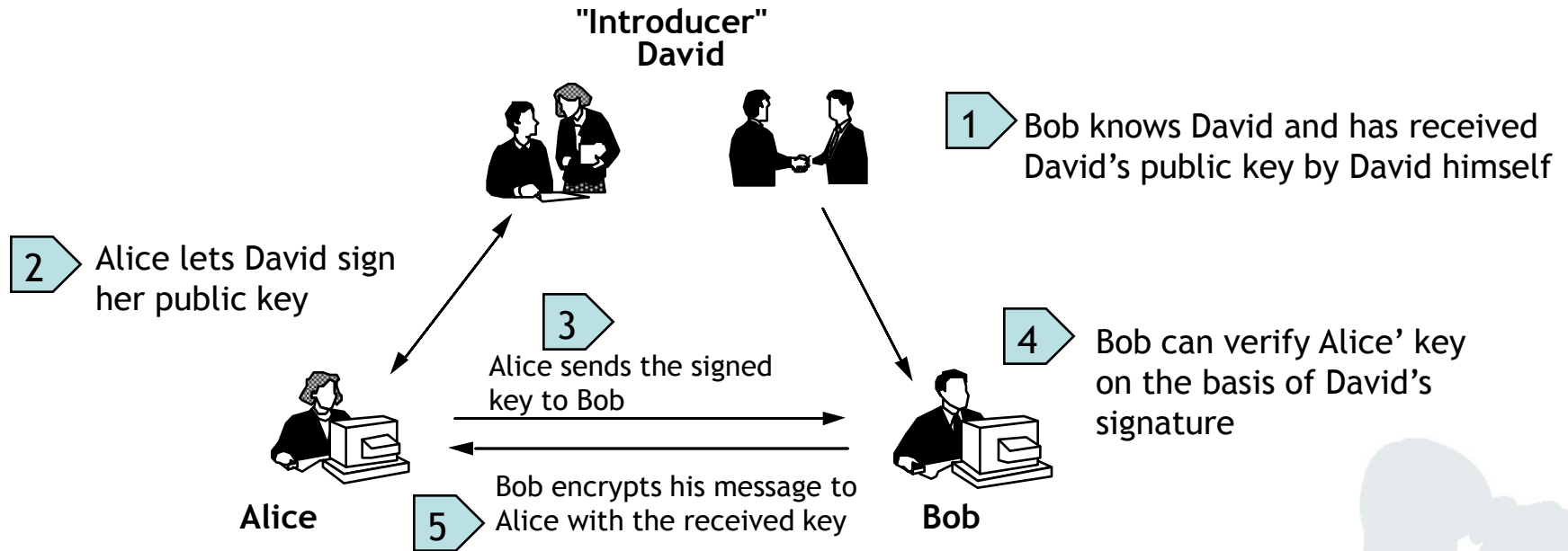
- The actual checking of the identity of the key owner takes place at so called Registration Authorities (e.g. notaries, bank branches, T-Points, ...)
- Security of the infrastructure depends on the reliability of the CAs.

# Content of a Key Certificate (according to German Signature Law and Regulation)



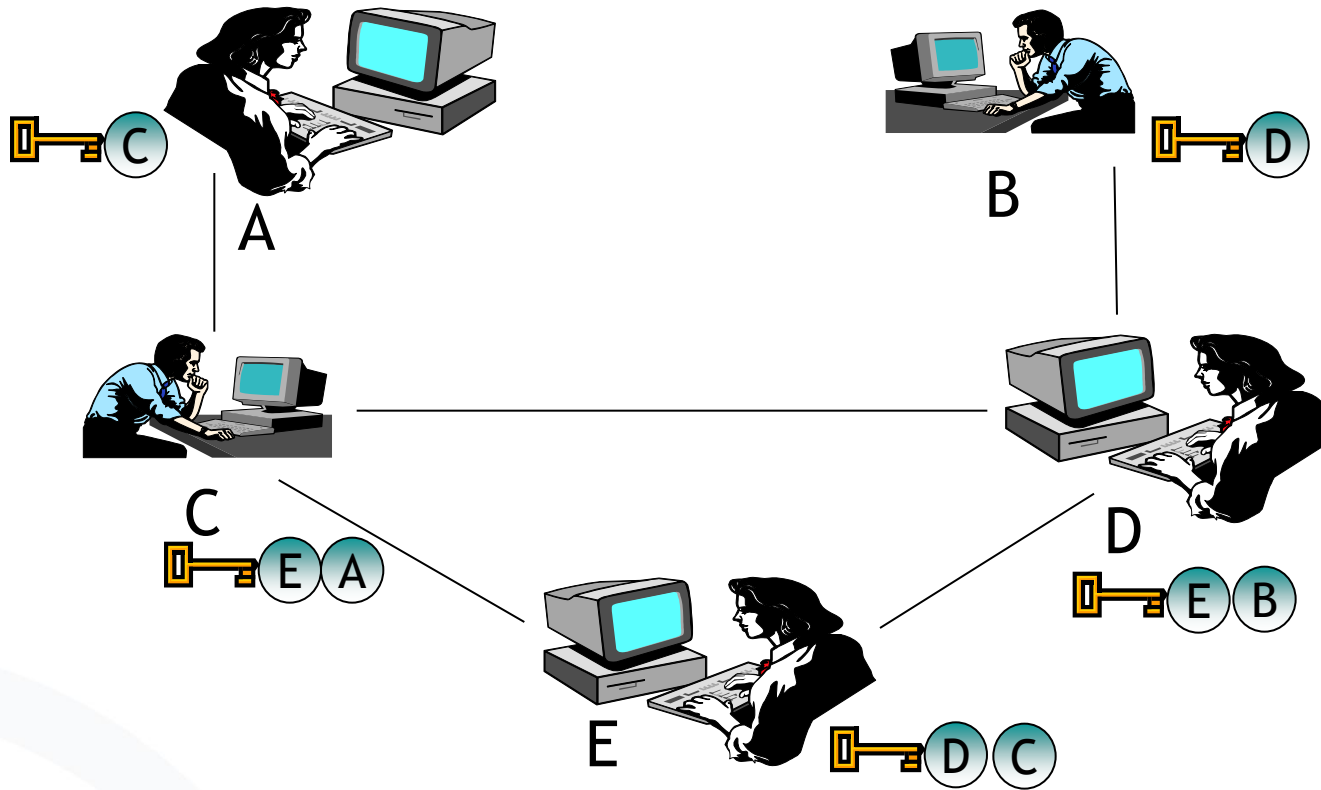
- Reliable identification of persons who apply for a certificate
- Information on necessary methods for fraud resistant creation of a signature
- Provision for secure storage of the private key
  - at least Smartcard (protected by PIN)
- Publication of the certificate (if wanted)
- Barring of certificates
- If necessary issuing of time stamps
  - for a fraud resistant proof that an electronic document has been at hand at a specific time

- Checking of the following items by certain confirmation centers (BSI, TÜVIT, ...)
  - Concept of operational security
  - Reliability of the executives and of the employees as well as of their know-how
  - Financial power for continuous operation
  - Exclusive usage of licensed technical components according to SigG and SigV
  - Security requirements as to operating premises and their access controls
- Possibly license of the regulation authority



- Each user can act as a “CA”.
- Mapping of the social process of creation of trust
- Keys are “certified” through several signatures.
- Expansion is possible by public key servers and (hierarchical) CAs

# Web of Trust Example



## Web of Trust:

- Certification of the public keys mutually by users
- Level of the mutual trust is adjustable.

- Introduction
- Classical cryptosystems
- Public key cryptography
  - General concept
  - Algorithms
  - Hybrid systems
  - Key management
  - Example: PGP

- PGP = Pretty Good Privacy
- De facto-Standard for freely accessible e-mail encryption systems on the Internet
- First implementation by Phil Zimmermann
- Long trial against Phil Zimmermann because of suspicion of violation of export clauses
- In U.S. free version in cooperation with MIT (agreement with RSA because of then patent)
- Meanwhile commercialized: [www.pgp.com](http://www.pgp.com)
- Gnu Privacy Guard (GPG): non-commercial Open Source variant (OpenPGP, RFC2440)



**Von:** Heiko Rossnagel  
**Betreff:** Klausur MC1  
**An:** Jan Muntermann  
**Cc:**

-----BEGIN PGP MESSAGE-----  
Version: PGP 8.0 - not licensed for commercial use: [www.pgp.com](http://www.pgp.com)

hQCMA5/VPPIP3satAQP+LqxvxFsk4G/TaexpMLX436biwBp6xP8pa89R7ro5Xo.  
uHEs07/tFrJFQJpPbcUWouy47p4sR2FO+IXqJuJyHp5ExMGIdmQCpGXEOs2Ijw.  
B5TXKtUB8YJ|dpPncK61as78RBP1sq8VDrAlYopEaeqMMw2pkBuoxyo3KCiRkhi.  
Ag4DIYlowhVX62wQCAD2L9WAA97xEUBWMET6kR9n5+oafTBF+ROlv6UOz2TO5S.  
Alkh23iQ0LI9Drye/uygpcQpT2HhTtZY1AjjudLvi+GsegOlWmBjY8q8G1Y61C.  
kDP3GEanyDiDU6R9F1XFovxPNMk6Ek8hH6qZ37hhDNDcXkxkSjM3nJ2VuuLvXb.  
uOuXNA9iAC96dhg7NpvzCJI2J7xRMtuBc9BUI8LXODrvGLwnLtaD5+EvgL1xTu.  
dfvQ3NiGrUEQsOHVxwjQdMtr8C09kREYLuAdD7j/05WtsAdbAVMn72PYFOIRfZ.  
i77MitBfAbxXF0gFS7/b2LccbaK8fx6e1VNFnVO7B/9qpdOGg5WZVP2eQA5fbw.  
h2oTOSjWCRp/v5s9Og1aUtcAxd1RAjQPhVsFS2eXXMn9ZzvNIFMh6Ktqnt6Ei.  
m39jRjPE9Ob/HLjMwPAXUHyneh9QrCX1X5qHORNcjIYVrnQyZGIk8t39059FBd.  
cr1rhf6ht7SwGgfgGW2aL8HyiFEVRC6piJaJFmrzifnzliwfuf82Tc42GBd9bP.  
E1IJGt9QLiwMmXormxcOg+WR2Ix4nGFx17Hy1vjKqpn7gfyLxXjgeDCnjxm708J.  
Njwtr+1SkqMCXs+PzcAHDsiuG.  
pE3huhK5cfvulUg7+Oa9SUAy4.  
NZncI3vJgkZeZr1bh+pi4dRjs(  
=hC09  
-----END PGP MESSAGE-----

heiko rossnagel  
frankfurt direkt  
-25306 D-60054 frankfurt

PGPTray - Enter Passphrase

Message was encrypted to the following public key(s):

- Heiko Rossnagel <heiko.rossnagel@m-lehrstuhl.de> (DH/2048)
- Jan Muntermann <munterma@wiwi.uni-frankfurt.de> (RSA/1024)

Enter passphrase for your private key:  Hide Typing

OK Cancel

Text Viewer

Hallo Jan,  
Anbei meine Aufgaben für die MC1 Klausur:

Copy to Clipboard OK

- Certification of public keys by users: “Web of Trust”
- Differentiation between ‘validity’ and ‘trust’
  - ‘Trust’: trust that a person / an institution signs keys only if their authenticity has really been checked
  - ‘Validity’: A key is valid for me if it has been signed by a person / an institution I trust (ideally by myself)
- Support through key servers
  - Collection of keys
  - Allocation of ‘validity’ and ‘trust’ remains task of the users.
- Path server: finding certification paths between keys

Keys	Validity	Trust	Size	Description
Andreas Albers <andreas.albers@m-lehrstuhl.de>	●	▬	2048/1024	DH/DSS public key
Elvira Koch <Elvira.Koch@M-Lehrstuhl.de>	●	▬	3096/1024	DH/DSS public key
fritsch@fsinfo.cs.uni-sb.de	●	▬	1024	RSA legacy public key
<b>Heiko Rossnagel &lt;heiko.rossnagel@m-lehrstuhl.de&gt;</b>	●	▬	2048/1024	DH/DSS key pair
Heiko Rossnagel <heiko.rossnagel@m-lehrstuhl.de>	●	▬	1024/1024	
Jan Muntermann <munterma@wiwi.uni-frankfurt.de>	●	▬	1024	
Kai Rannenbergl <kara@iig.uni-freiburg.de>	●	▬	2048/1024	
Kai R. Rannenbergl 2048 <kara@iig.uni-freiburg.de>	●	▬	2048	
Lothar Fritsch <fritsch@klammeraffe.org>	●	▬	4096/1024	
Lothar Fritsch <fritsch@klammeraffe.org>	●	▬		
Lothar Fritsch <Lothar.Fritsch@M-Lehrstuhl.de>	●	▬		
Lothar Fritsch <fritsch@klammeraffe.org>	●	▬		
fritsch@fsinfo.cs.uni-sb.de	●	▬		
Jan Muntermann <munterma@wiwi.uni-frankfurt.de>	●	▬		
Andreas Albers <andreas.albers@m-lehrstuhl.de>	●	▬		
Lothar Fritsch <Lothar.Fritsch@whatismobile.de>	●	▬		
Stefan Figge <stefan.figge@m-lehrstuhl.de>	●	▬	2048/1024	

1 key(s) selected

Lothar Fritsch <fritsch@klammeraffe.org>

General | Subkeys

ID: 0xFED07240  
 Type: DH/DSS  
 Size: 4096/1024  
 Created: 15.01.2004  
 Expires: 15.01.2006  
 Cipher: CAST  
 Enabled

Fingerprint  
 6075 14A6 1248 5A4A 7E18 6187 AE57 9E4D FED0 7240  
 Hexadecimal

Trust Model  
 Invalid  Valid  Untrusted  Trusted







Close Help


PGPkeys Search Window

Search for keys on  where

User ID

Search Pending Area

Keys	Validity	Trust	Size	Description
<input type="checkbox"/>  Kai R. Rannenberg 2048 <kara@iig.uni-freiburg.de>		<input type="text" value=""/>	2048	RSA legacy public key
<input type="checkbox"/>  Kai R. Rannenberg <kara@iig.uni-freiburg.de>		<input type="text" value=""/>	1024	RSA legacy public key
<input type="checkbox"/>  kara <kara@iig.uni-freiburg.de>		<input type="text" value=""/>	2048/1024	DH/DSS public key



The screenshot shows a web browser window titled "Public Key Server -- Verbose Index 'Kai Rannenber'". The address bar contains the URL: <http://blackhole.pca.dfn.de:11371/pks/lookup?op=vindex&search=Kai+Rannenber>. The main content area displays a list of public keys with columns for Type, bits/keyID, Date, and User ID. The keys are listed in a table format with various key IDs and user identifiers.

Type	bits/keyID	Date	User ID
pub	1024/ <a href="#">AF1FDF70</a>	1997/09/18	kara < <a href="mailto:kara@iig.uni-freiburg.de">kara@iig.uni-freiburg.de</a> >
sig	<a href="#">OB6375FD</a>		Matthias Schunter < <a href="mailto:schunter@acm.org">schunter@acm.org</a> >
sig	<a href="#">DSCDE083</a>		Herbert Damker < <a href="mailto:damker@iig.uni-freiburg.de">damker@iig.uni-freiburg.de</a> >
sig	<a href="#">879AC041</a>		Birgit Pfitzmann 1 < <a href="mailto:pfitzb@informatik.uni-hildesheim.de">pfitzb@informatik.uni-hildesheim.de</a> > NO LEGAL RELEVANCE
sig	<a href="#">8128DC75</a>		Gerhard Weck < <a href="mailto:73064.2271@compuserve.com">73064.2271@compuserve.com</a> >
sig	<a href="#">8EFO41F1</a>		Kai R. Rannenber 2048 < <a href="mailto:kara@iig.uni-freiburg.de">kara@iig.uni-freiburg.de</a> >
sig	<a href="#">2F8D5039</a>		Kai Martius < <a href="mailto:Kai@imib.med.tu-dresden.de">Kai@imib.med.tu-dresden.de</a> >
sig	<a href="#">5C3C4FE4</a>		Holger Reif < <a href="mailto:reif@prakinf.tu-ilmeneau.de">reif@prakinf.tu-ilmeneau.de</a> >
sig	<a href="#">AF1FDF70</a>		kara < <a href="mailto:kara@iig.uni-freiburg.de">kara@iig.uni-freiburg.de</a> >
sig	<a href="#">49EF1D84</a>		Hannes Federrath < <a href="mailto:federrath@inf.tu-dresden.de">federrath@inf.tu-dresden.de</a> >
			Kai R. Rannenber < <a href="mailto:kair@microsoft.com">kair@microsoft.com</a> >
sig	<a href="#">OB6375FD</a>		Matthias Schunter < <a href="mailto:schunter@acm.org">schunter@acm.org</a> >
sig	<a href="#">AEB4BCDD</a>		fapp2_AEB4BCDD_HSK < <a href="mailto:fapp2@cam.ac.uk">fapp2@cam.ac.uk</a> >
sig	<a href="#">AF1FDF70</a>		kara < <a href="mailto:kara@iig.uni-freiburg.de">kara@iig.uni-freiburg.de</a> >
sig	<a href="#">044584B5</a>		Douglas Swiggum < <a href="mailto:Swiggum@Waisman.Wisc.Edu">Swiggum@Waisman.Wisc.Edu</a> >
			Kai Rannenber < <a href="mailto:kara@iig.uni-freiburg.de">kara@iig.uni-freiburg.de</a> >
sig	<a href="#">OCB6E63F</a>		Martin Reichenbach < <a href="mailto:marei@iig.uni-freiburg.de">marei@iig.uni-freiburg.de</a> >
sig	<a href="#">AF1FDF70</a>		kara < <a href="mailto:kara@iig.uni-freiburg.de">kara@iig.uni-freiburg.de</a> >
			kara < <a href="mailto:kara@telematik.iig.uni-freiburg.de">kara@telematik.iig.uni-freiburg.de</a> >
sig	<a href="#">OB6375FD</a>		Matthias Schunter < <a href="mailto:schunter@acm.org">schunter@acm.org</a> >
sig	<a href="#">AF1FDF70</a>		kara < <a href="mailto:kara@iig.uni-freiburg.de">kara@iig.uni-freiburg.de</a> >
			Kai R. Rannenber < <a href="mailto:kara@iig.uni-freiburg.de">kara@iig.uni-freiburg.de</a> >
sig	<a href="#">OB6375FD</a>		Matthias Schunter < <a href="mailto:schunter@acm.org">schunter@acm.org</a> >
sig	<a href="#">AF1FDF70</a>		kara < <a href="mailto:kara@iig.uni-freiburg.de">kara@iig.uni-freiburg.de</a> >

- Network of public-key servers:
  - [pgpkeys.pca.dfn.de](http://pgpkeys.pca.dfn.de)
  - [www.cam.ac.uk/pgp.net/pgpnet/email-key-server-info.html](http://www.cam.ac.uk/pgp.net/pgpnet/email-key-server-info.html)
  - ...

- Brute-Force-Attacks on the pass phrase
  - PGPCrack for conventionally encrypted files
- Trojan horses, changed PGP-Code
  - e.g. predictable random numbers, encryption with an additional key
- Attacks on the computer of the user
  - Not physically deleted files
  - Paged memory
  - Keyboard monitoring
- Analysis of electromagnetic radiation
- Non-technical attacks
- Confusion of users [WT99]

- **[Bi05] Bishop, Matt:** *Introduction to Computer Security*. Boston: Addison Wesley, 2005. pp. 113-116.
- **[DH76] Diffie, Whitfield and Hellman, Martin E.:** New Directions in Cryptography, *IEEE Transactions on Information Theory*, 1976, 22(6), pp. 644-654.
- **[RSA78] Rivest, Ron L., Shamir, A. and Adleman, L.:** A Method for Obtaining Digital Signatures and Public Key Cryptosystems, *Communications of the ACM*, February 1978, 21(2), pp. 120-126.
- **[WT99] Whitten, Alma and Tygar, J.D.** *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0*, In: Proceedings of the 9th USENIX Security Symposium, August 1999, [www.gaudior.net/alma/johnny.pdf](http://www.gaudior.net/alma/johnny.pdf)