Application Layer (1)

Functionality:

- providing applications (e-mail, www, USENET etc)
- providing support protocols to allow the real applications to function properly
  - *security* comprising a large number of concepts and protocols
  - *DNS* (Distributed Name Service) handling naming within the Internet
  - *network management* (SNMP)

Network security nowadays is regarded as one of the biggest problems for upcoming Internet applications

Secure communication in general addresses four different topics:

- *secrecy (Vertraulichkeit)*: keeping information secret, i.e. out of the hands of unauthorized users
- *authentication (Authentifikation)*: determining whom you are talking to before revealing information
- *nonrepudiation (Verbindlichkeit)*: ensuring trustworthiness by means of digital signatures
- *integrity (Datenintegrität)*: ensuring the originality of received messages (not being modified)

Network Security (1)

Solutions to all problem areas are mainly based on **cryptographic** (Greek: secret writing) principles

Historically, mainly 4 human areas have used and contributed to the art of cryptography:

- military
- diplomacy
- diary writing
- private affairs

Today, economy is becoming the most important area where cryptography is used.

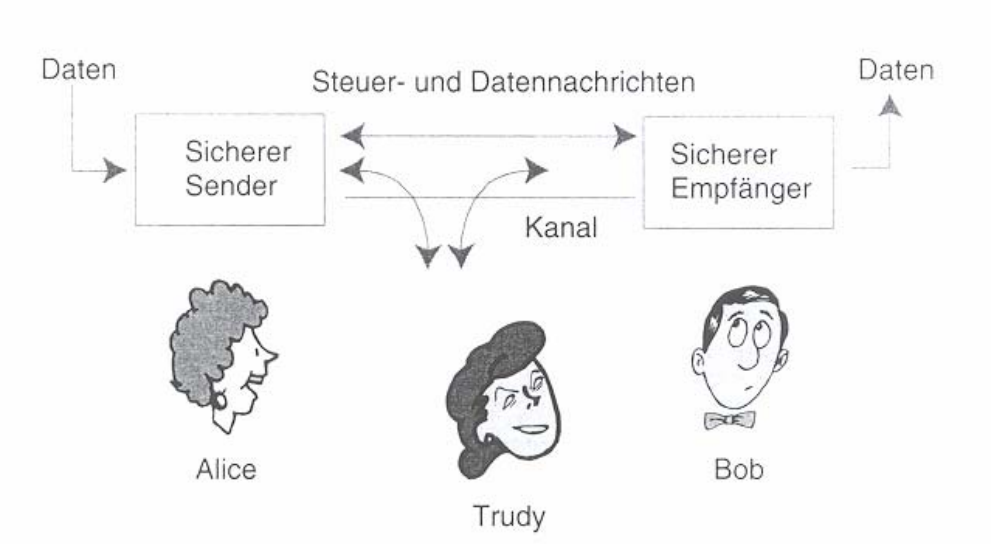
Objective:

providing secure Internet applications like

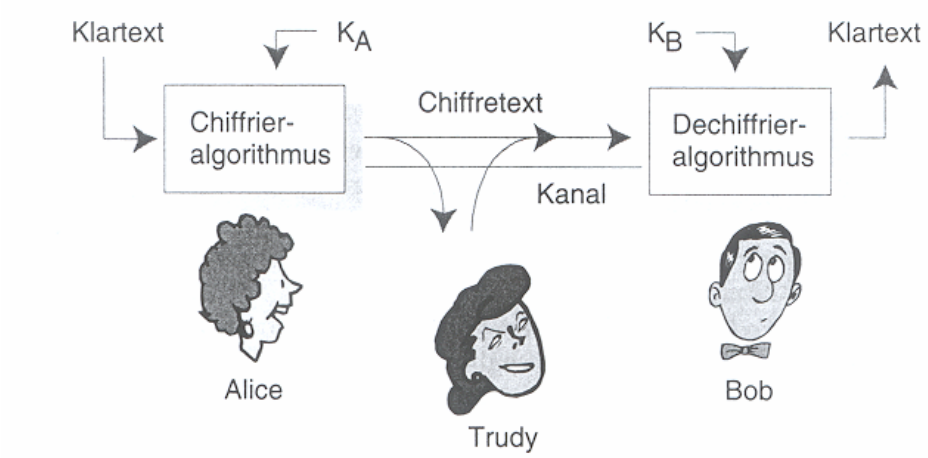
- secure e-mail system
- secure e-commerce - transactions

Network Security (2)

Representatives for sender, recipient, and intruder

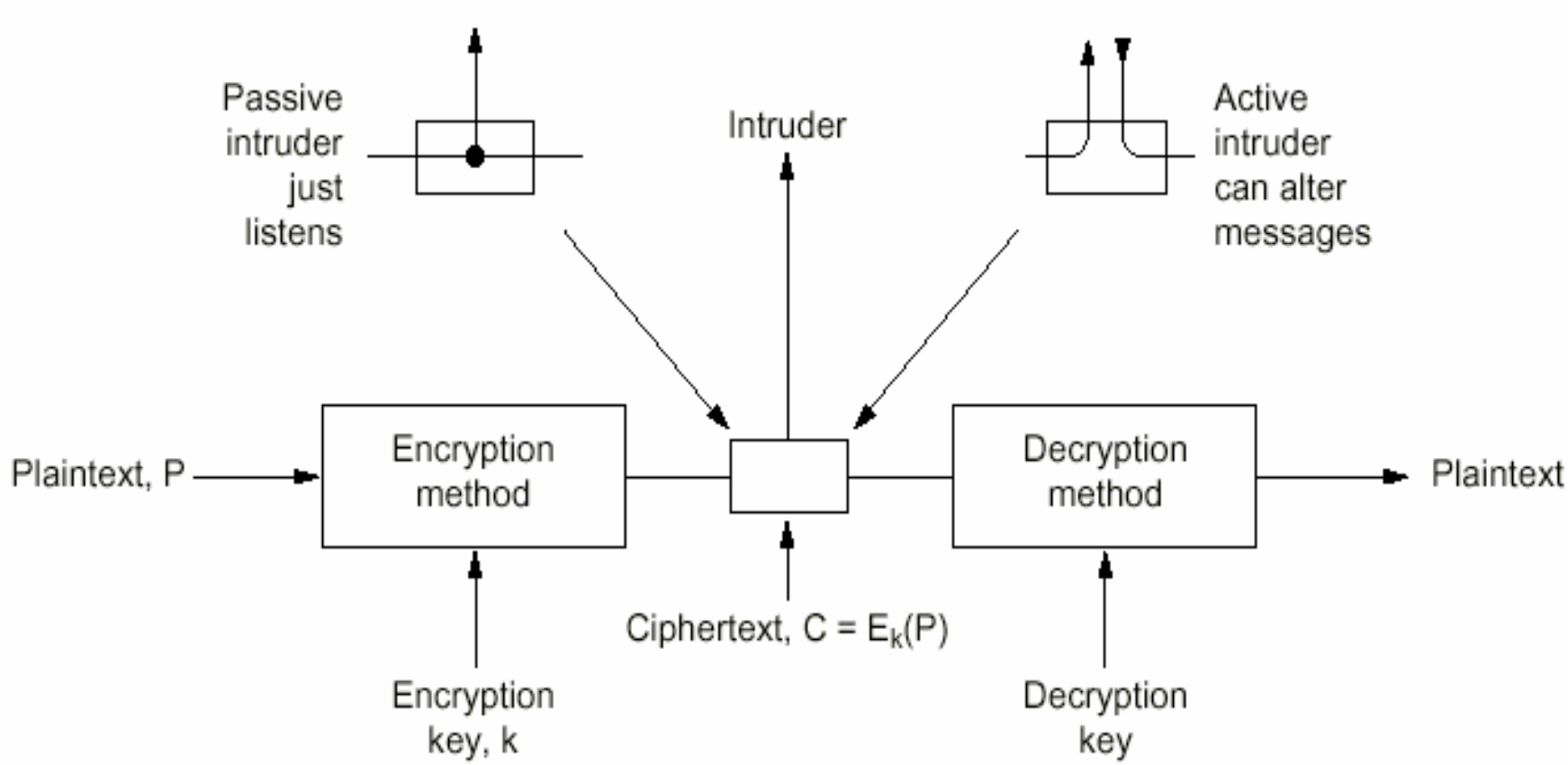


The encryption model (1):



Network Security (3)

The encryption model (2):



Symmetric Key Systems (1)

Symmetric Key System:

Keys of Alice and Bob are identical and secret

Public Key System:

Both, Alice and Bob have a pair of keys, one is public, the other is only known by its holder.

1. Symmetric Key Systems (old)

Traditional encryption methods have been divided historically into two categories:

- substitution ciphers (Ersetzungschiffren) (preserve the order of the plaintext symbols but disguise them)
- transposition ciphers (Umstellungschiffren) (reorders the plaintext symbols but do not disguise them)

Ancient and simple substitution cipher: **Caesar's cipher**

The ciphertext alphabet results from a shift of k letters in the plaintext alphabet (key$:=k$).

Generalization of Caesar's chiffre: **monoalphabetic substitution**

Each letter or group of letters is replaced by another letter or group of letters to disguise it

Example for a monoalphabetic substitution

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

Symmetric Key Systems (2)

Three main approaches to break these type of ciphers

- *ciphertext only attack:* exploiting statistical properties of natural languages
- *known plaintext attack:* guessing a probable word or phrase
- *chosen plaintext attack:* the intruder is able to submit his plaintext and gets the corresponding ciphertext

Example for the second approach:

CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ
QJSGS TJQZZ MNQJS VLNSX VSZJU JDSTS JQUUS JUBXJ
DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW

Transposition ciphers

Instead of disguising letters they are reordered

Example for a columnar transposition

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	**w**	o	a	b	c	d

Plaintext

pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

Ciphertext

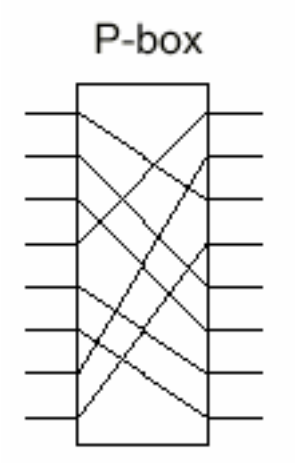
AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

Symmetric Key Systems (3)

2. Symmetric Key Systems (modern)

Idea: Concatenation of standard transposition (permutation) and substitution elements (boxes):

Example for a P(ermutation)**-box** (01234567 ---> 24506713)

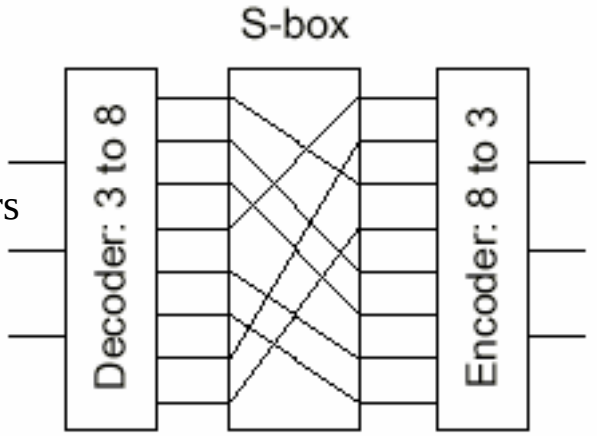


Example for a S(ubstitution)**-box** (3bit plaintext to 3bit ciphertext)

By appropriate wiring of the P-box inside, any substitution can be accomplished.

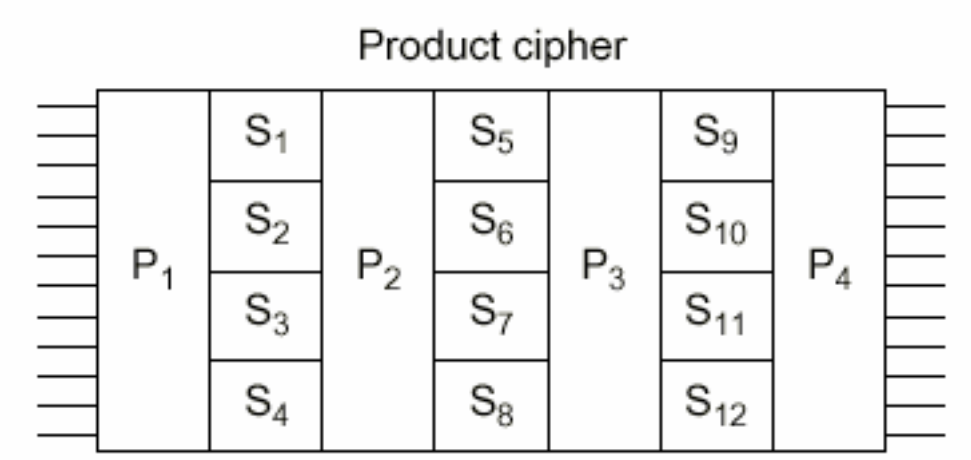
In this example:

Numbers 01234567 (in octal form) each are replaced by the numbers 24506713 (in octal form).



Symmetric Key Systems (4)

Example for a product cipher (concatenation)



Standard: DES (Data Encryption Standard)

- plaintext is encrypted in blocks of 64 bits
- the algorithm has 19 rounds (16 of them are functionally identical with different keys)
- the steps for decryption are done in the reverse order of those for encryption

Public Key Systems (1)

3. Public-Key Systems

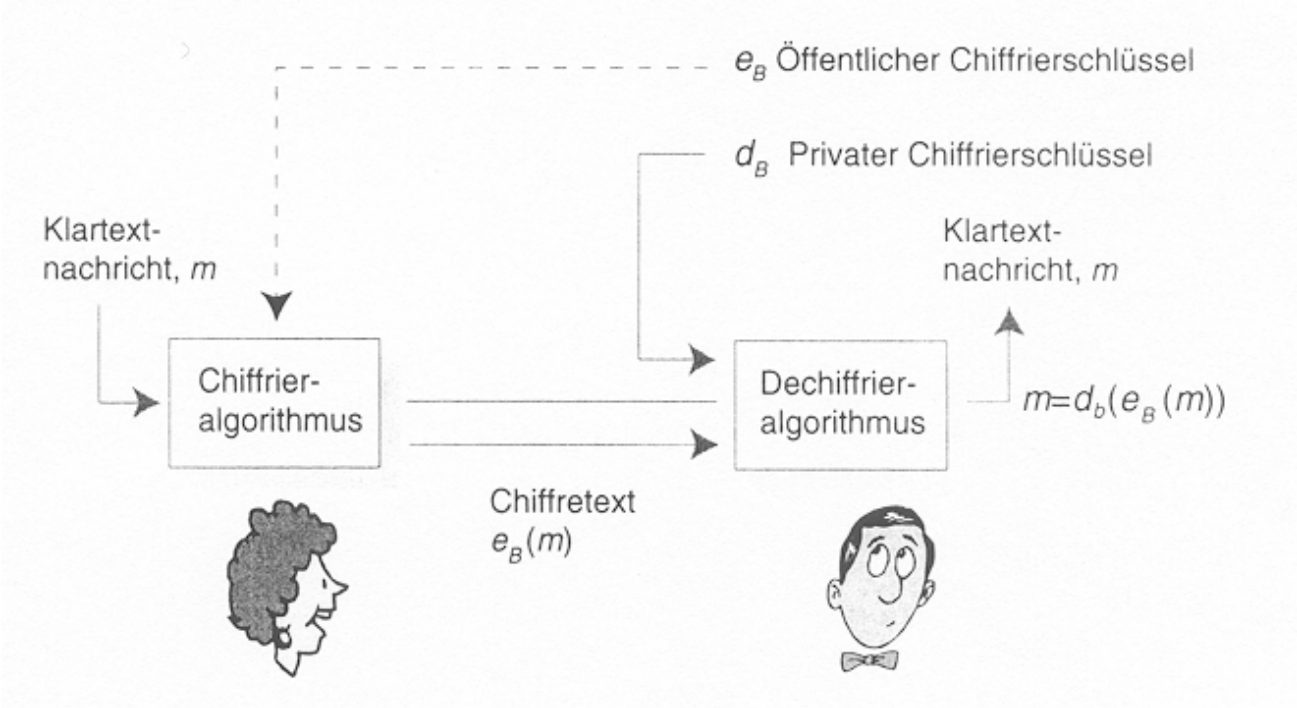
Basic problem behind:

Is it possible that Alice and Bob can communicate by encrypted messages without having exchanged before a common secret key?

Principal solution:

Each party has a pair of keys, a public one (accessible to everybody) and a private one (only known by itself)

The general model



Public Key Systems (2)

The RSA algorithm

Two components:

- Selecting the keys
- Applying the encryption and decryption algorithm

Selecting the keys (by Bob):

1. Choose two large primes, p and q
2. Compute $n = p \times q$ and $z = (p-1) \times (q-1)$.
3. Choose a number relatively prime to z, smaller than n and call it e (e is used for $e$ncryption) .
4. Find d such that $e \times d = 1 \; mod \; z$ (d is used for decryption) .
5. The public key is (n,e), the private key is (n,d).

Encryption (by Alice) of a bit pattern (number) m such that $m < n$ by means of Bob's public key (n,e).
The resulting cipher c is:

$c = m^e \; mod \; n$

Decryption (by Bob) of c by means of his private key (n,d) in order to get the plaintext m:

$m = c^d mod \; n$

Public Key Systems (3)

Example of the RSA algorithm

p=5, q=7 ---> n = 35, z=24

Further, Bob selects e=5, d=29 (5*29 - 1 can be divided by 24)

Alice wants to send the message “LOVE” to Bob by encrypting each letter separately and interpreting each letter as the corresponding number (a maps to 1, ..., z maps to 26)

Tabelle 7.1 Die RSA-Verschlüsselung von Alice, $e = 5$, $n = 35$

Klartextbuchstabe	m: numerische Darstellung	m^e	Chiffretext $c = m^e \bmod n$
L	12	248832	17
O	15	759375	15
V	22	5153632	22
E	5	3125	10

Tabelle 7.2 Die RSA-Verschlüsselung von Bob, e = 29, n = 35

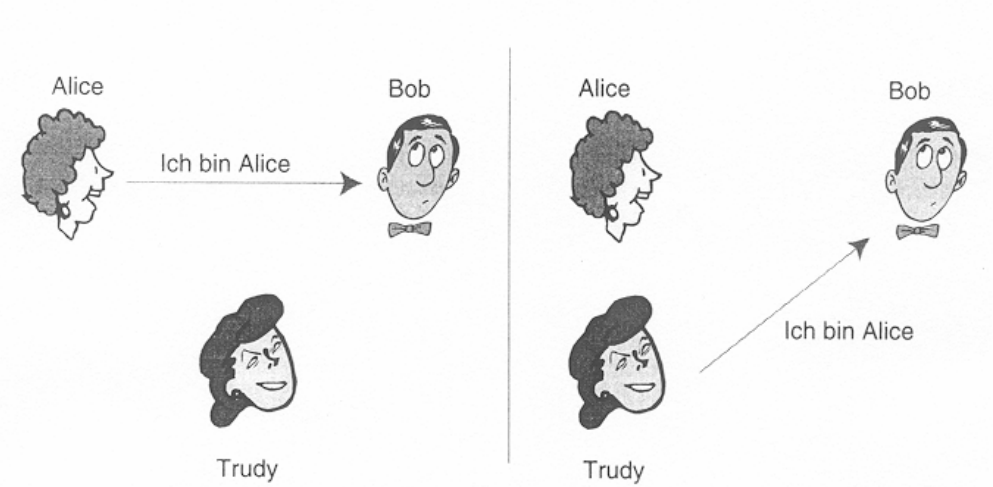
Chiffre-text c	c^d	Chiff-retext $m = c^d \bmod n$	Klartext-buch-stabe
17	481968572106750915091411825223072000	12	l
15	12783403948858939111232757568359400	15	o
22	8.516433190865377011956194499721 11e+38	22	v
10	10000000000000000000000000000	5	e

Authentication (1)

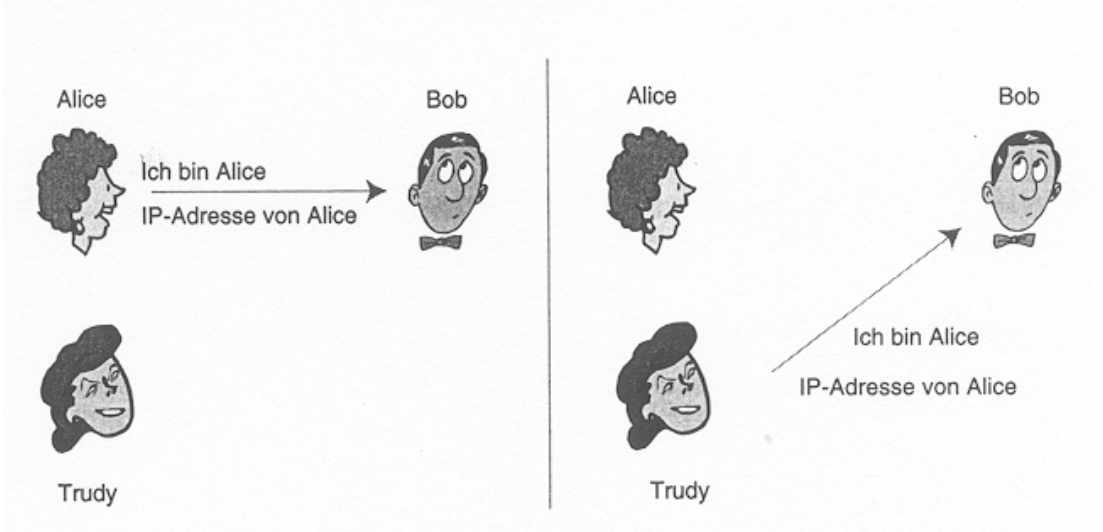
Authentication Protocols

- technique by which a process verifies that its *actual* communication partner is who it is supposed to be
- normally done before the partners start to exchange data messages, e.g. e-mails

Version ap 1.0

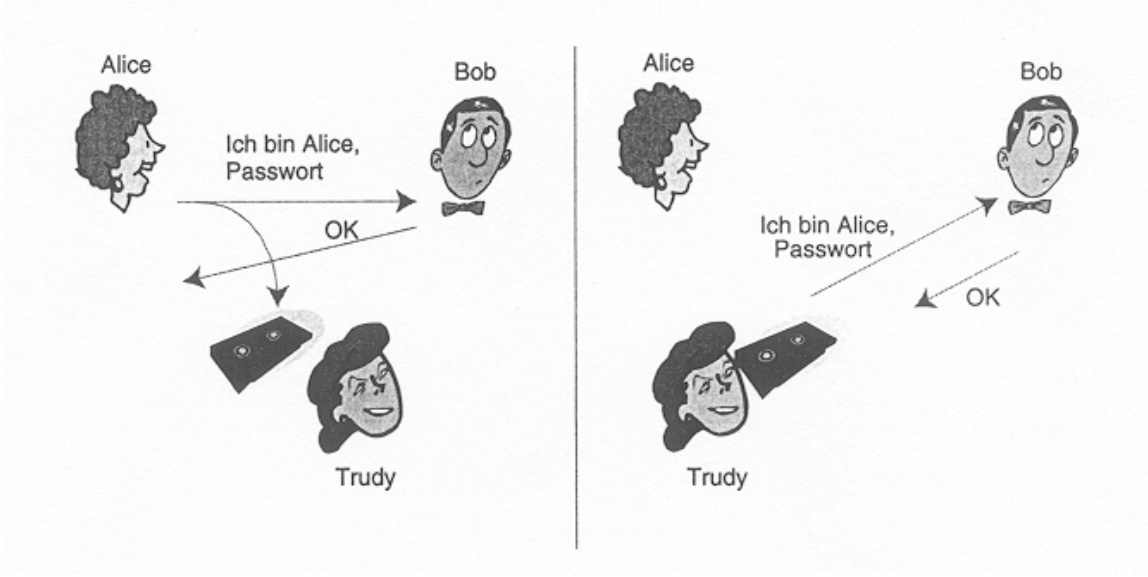


Version ap 2.0



Authentication (2)

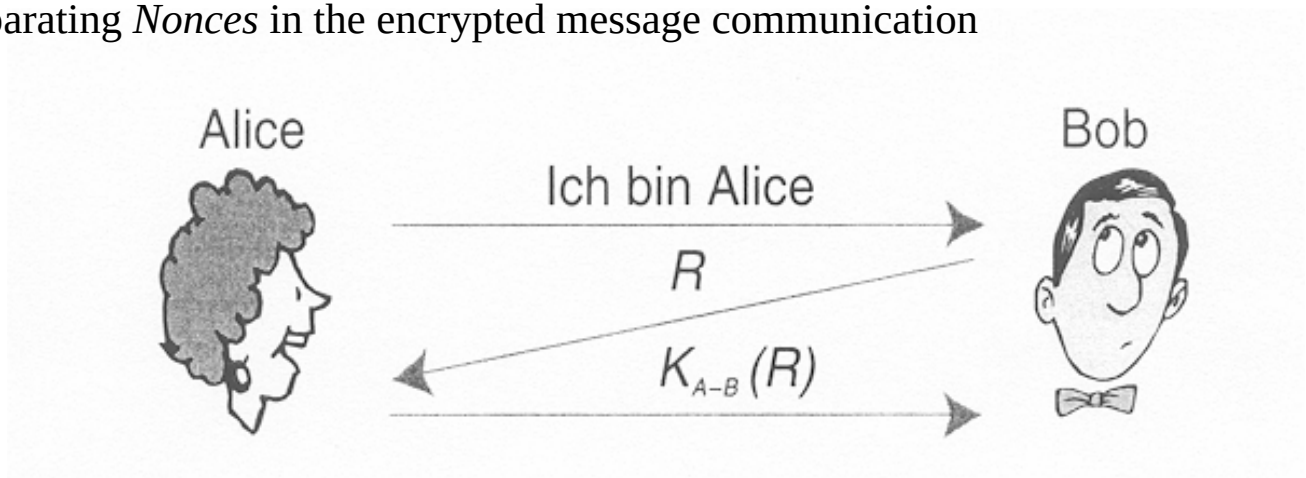
Version ap 3.0



Enryption of the password does hardly improve the security level!!

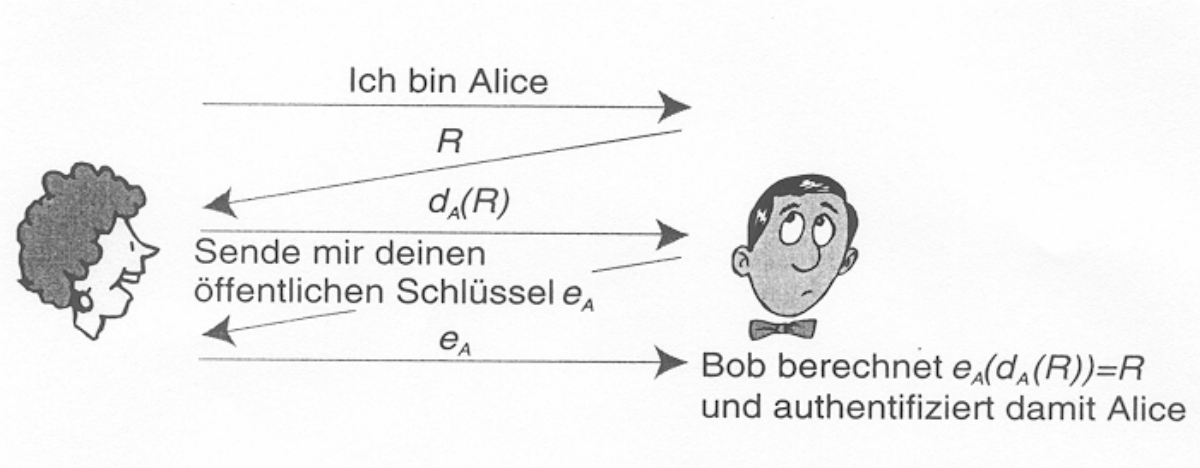
General problem behind: Impossibility to distinguish between “communication live“ and “playback“
Solution: Incorparating *Nonces* in the encrypted message communication

Version ap 4.0



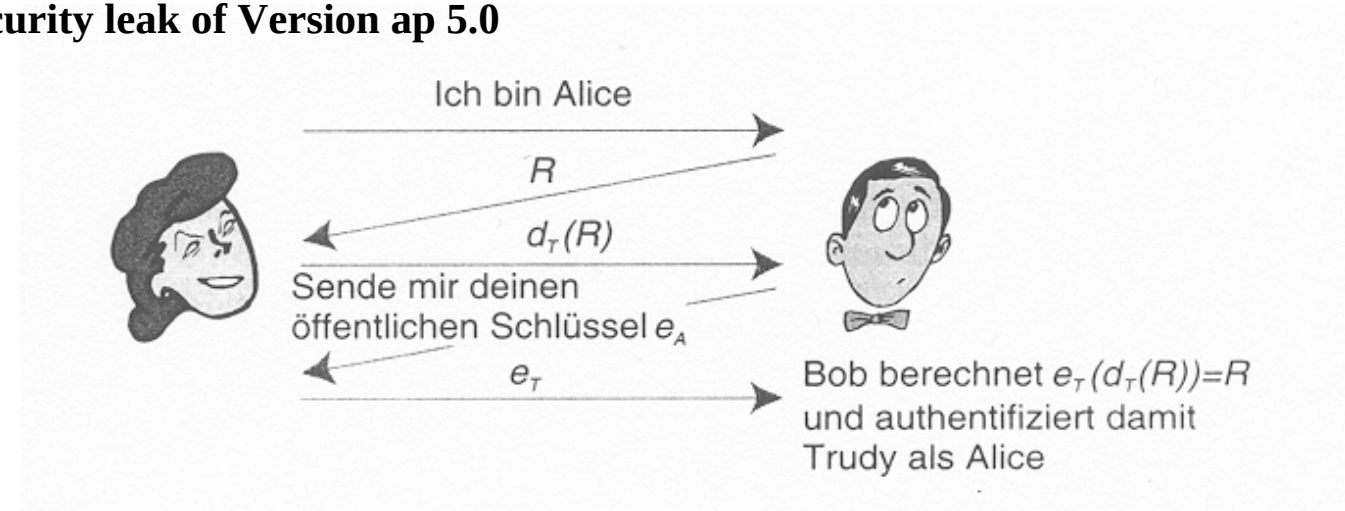
Authentication (3)

Version ap 5.0 (use of public keys)



General problem behind: ap. 5.0 is only as secure as the distribution of public keys is

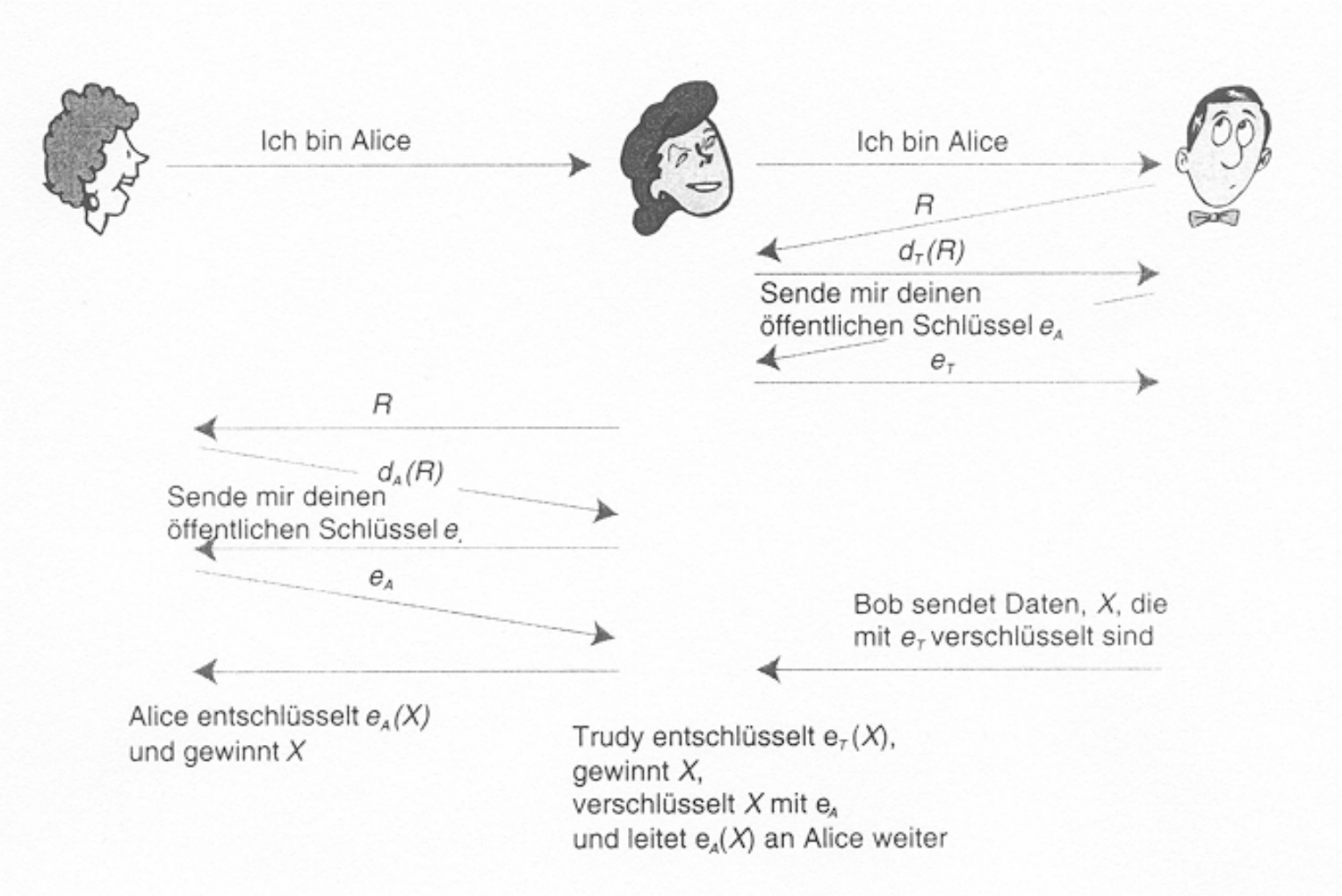
Security leak of Version ap 5.0



Authentication (4)

A more sophisticated and successful attack against ap. 5.0 , which even hardly can be detected by the communication partners Alice and Bob is called

Man-in-the-Middle (Bucket Brigade) - Attack

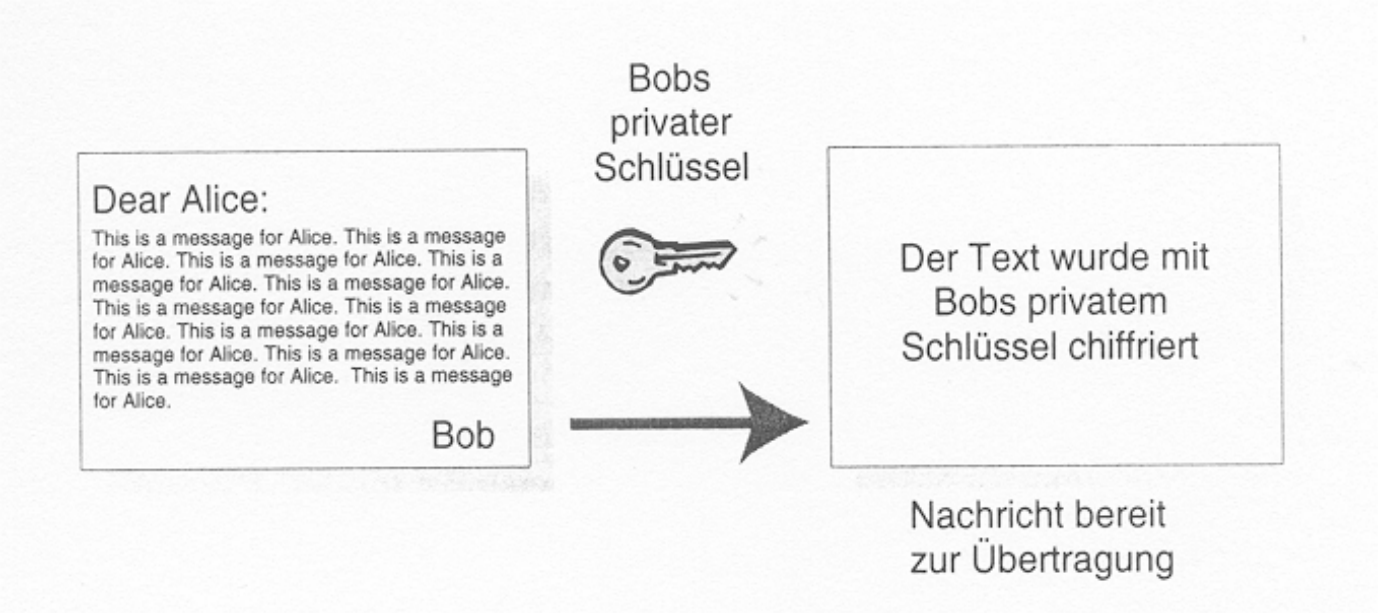


Digital Signatures (1)

Problem:
Finding an electronic adequate for the handwritten signature such that one party can send a signed message to another party in such a way that the following conditions hold:

- The receiver can verify the claimed identity of the sender (authentication)
- The sender later cannot repudiate having sent his message (nonrepudiation)
- The contents of the message cannot have been modified, e.g. by the receiver himself (integrity)

Solution 1: Creation of digital signatures by means of public keys



Drawback:
It couples secrecy on the one side with the triple (authentication, nonrepudiation, integrity) on the other side ---> it needs, often unnecissarily, too much computational overhead for encrypting/decrypting

Digital Signatures (2)

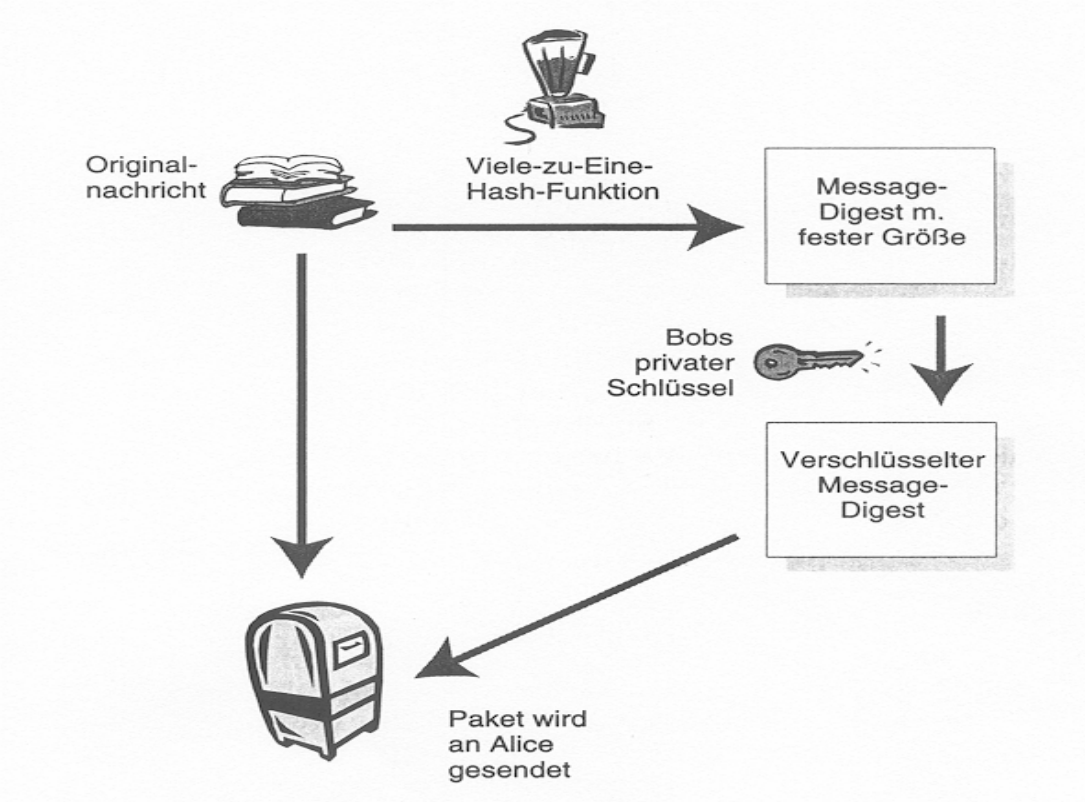
Solution 2: Creation of digital signatures by means of Message Digests, without encrypting the whole text.
Idea: Using a so-called *hash function* to create a "fingerprint" from any plaintext.
Hash function: a message *m* of any length is mapped to a bit string *H(m)* of fixed length such that

- H(m) is computed much easier (faster) than encrypting m
- it is almost impossible to find $m' \neq m$ and $H(m) = H(m')$ (ensuring data integrity)

Other examples using hash functions: checksum, CRC

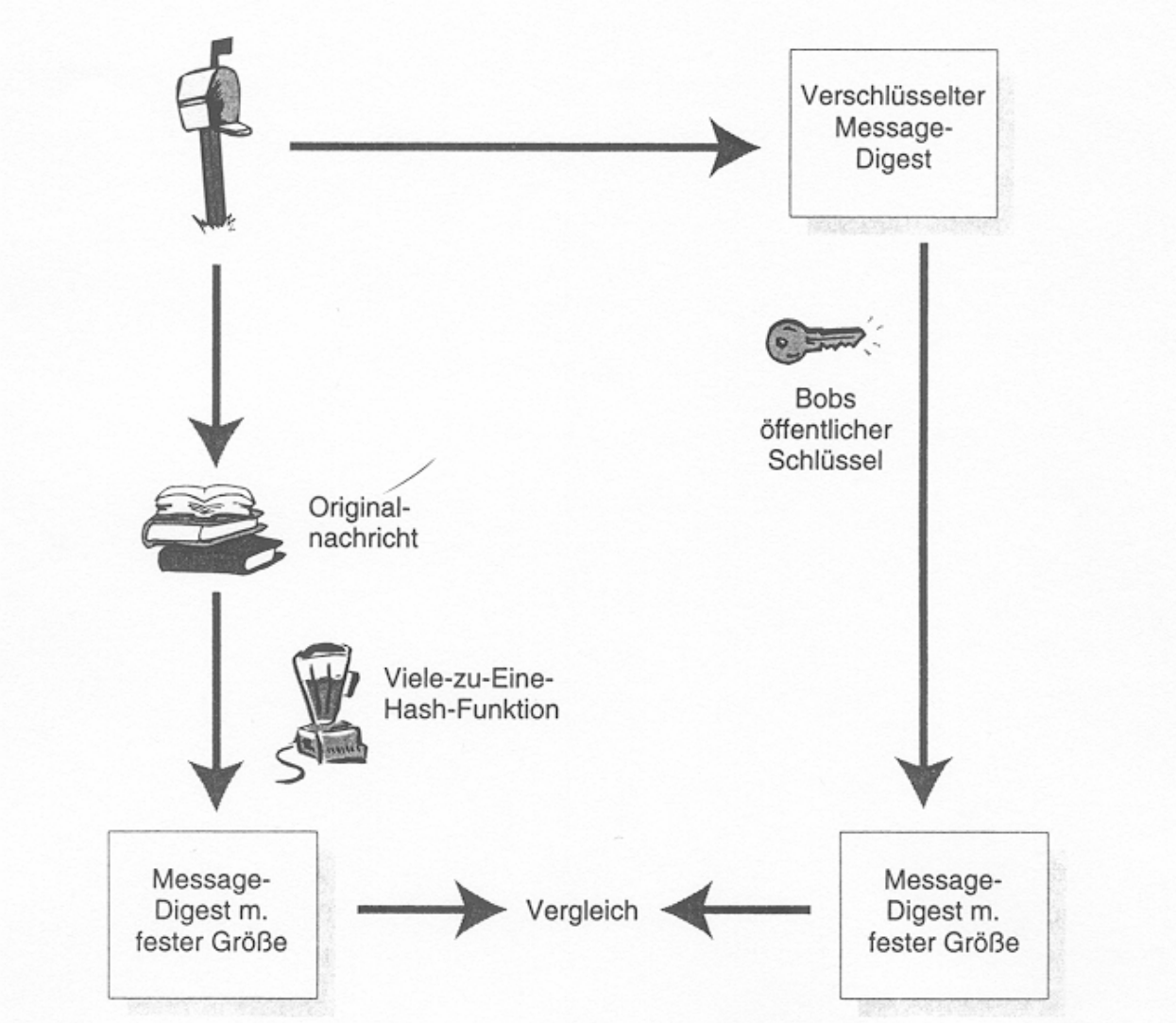
Now, in order to get the effect of a digital signature, we only have to encrypt (sign) the digest of a message.

Sending a digitally signed message



Digital Signatures (3)

Checking a digitally signed message



The most widely used message digest functions are MD5 and SHA-1. They operate by mangling bits in a sufficiently complicated way such that every output bit (bit of the digest) is affected by (dependent on) every input bit (bit of the message).

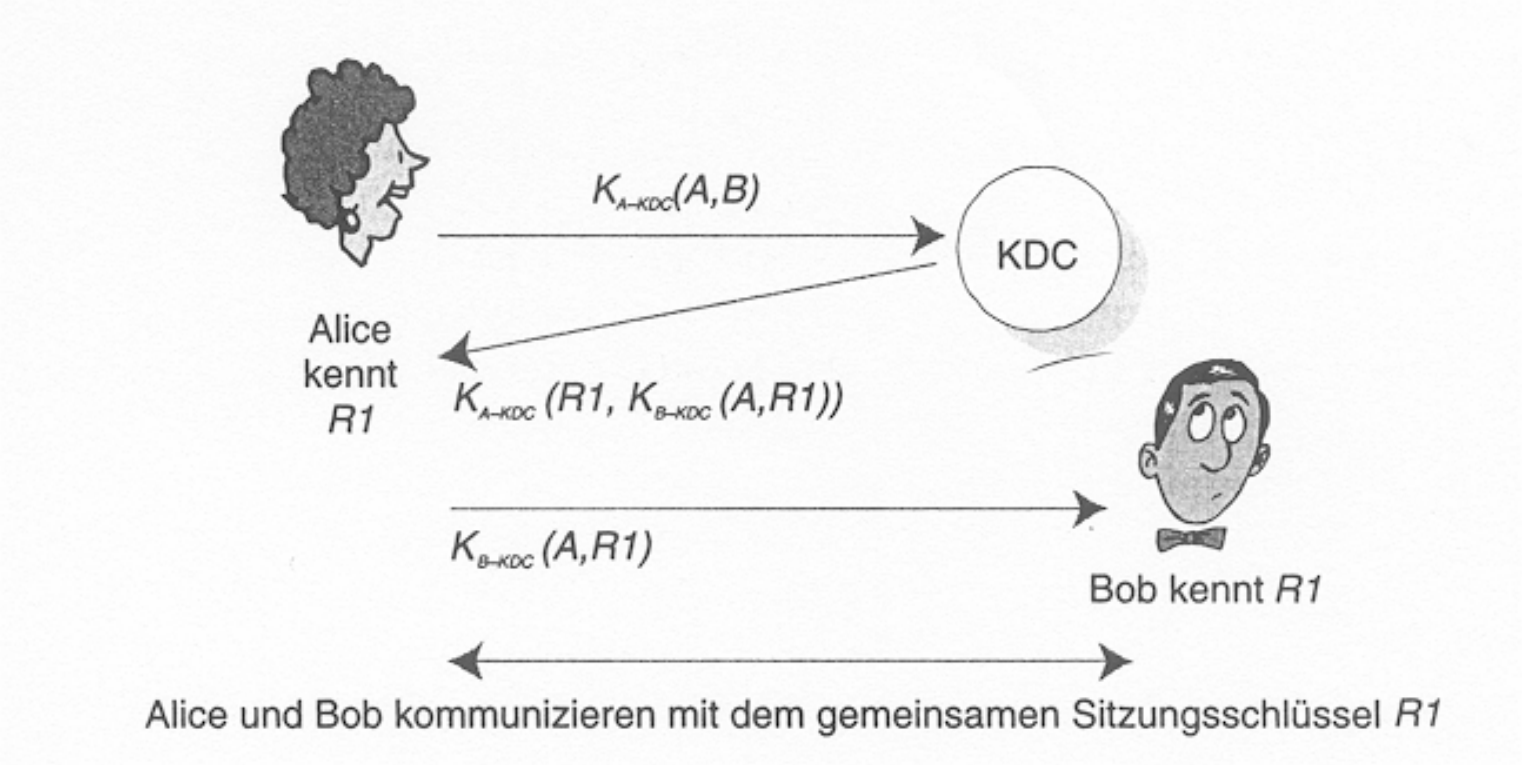
Key Management (Distribution and Certification) (1)

Remaining problem of the symmetric key approach:
How to agree a priori, i.e. before the secure communication between two partners starts, on their secret key?
Solution: Using a trustworthy third person, the so-called

Key Distribution Center (KDC)

Idea: KDC is a server having a secret key with each registered user of the system. Key distribution (session key management) and authentication now goes through the KDC.

Creating and distributing a unique session key between Alice and Bob via the KDC



Key Management (Distribution and Certification) (2)

Kerberos

- named after a multiheaded dog in Greek mythology guarding the entrance to Hades
- designed at MIT to allow workstation (end) users to access network resources (servers) in a secure way, i.e. when accessing users are authenticated and checked whether he has adequate access rights
- the so-called Authentication Server (AS) takes the role of the KDC
- used in many real operating systems (Windows 2000, Unix)

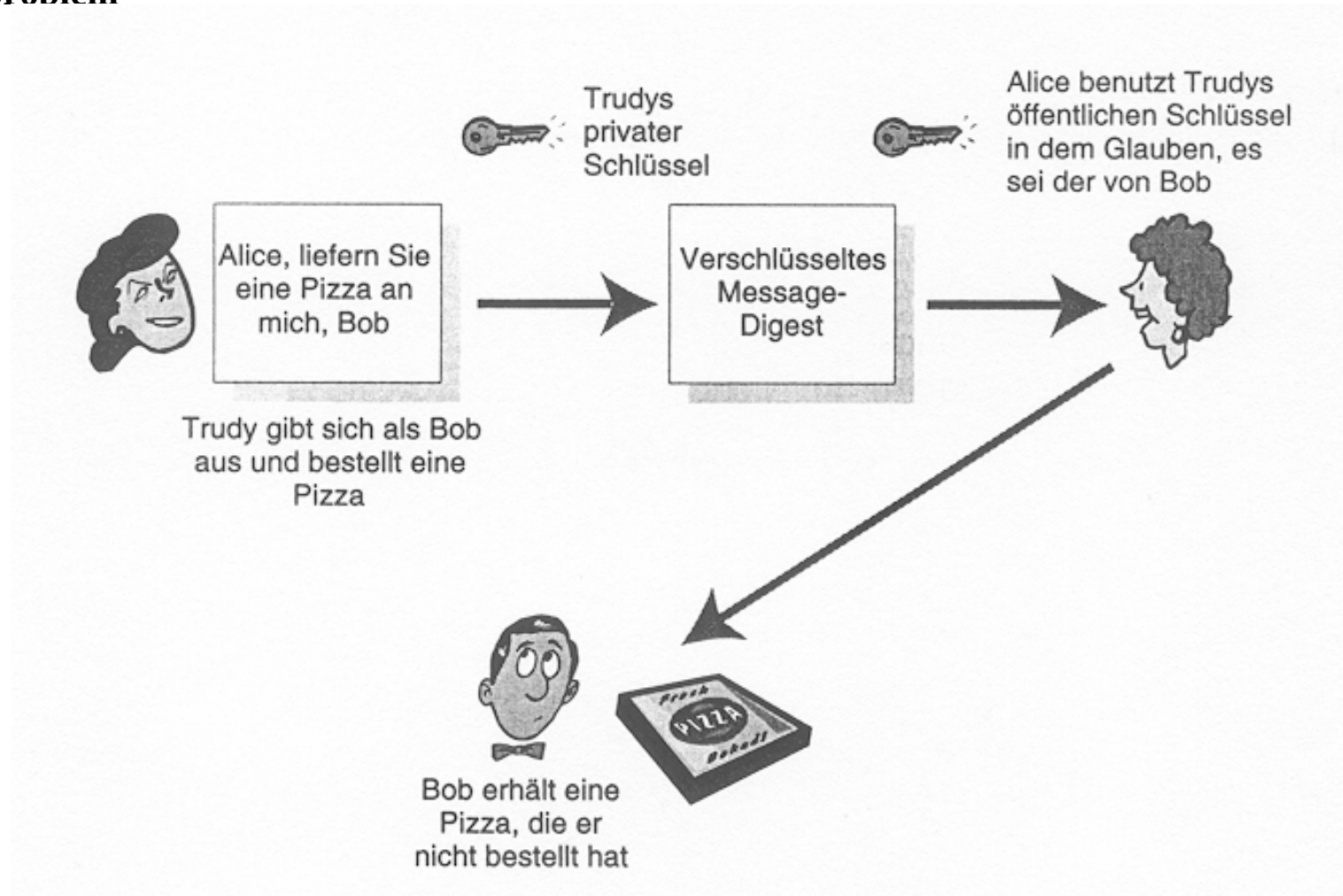
User Alice wants to access the file server Bob:

- Alice types her name and password in an arbitrary user station which is passed to the AS. She gets back the session key for this secure communication between Alice and AS encrypted by Alice's secret key.
- Alice tells AS that she wants to access Bob. AS authenticates Alice (using the session key), checks her access rights to Bob, and, if o.k., sends back the session key R1 for the communication between Alice and Bob together with a so-called *ticket* containing Alice's name, R1, and a timestamp marking a time-out, all this encrypted with the secret key between AS and Bob.
- Alice sends the ticket to Bob together with a Nonce encrypted with R1.
- Bob sends back to Alice the incremented Nonce encrypted with R1 authenticating himself to Alice.

Key Management (Distribution and Certification) (3)

Remaining problem of the public key approach:
How to ensure that the public key received is really the one of the sender?

Illustration of the problem



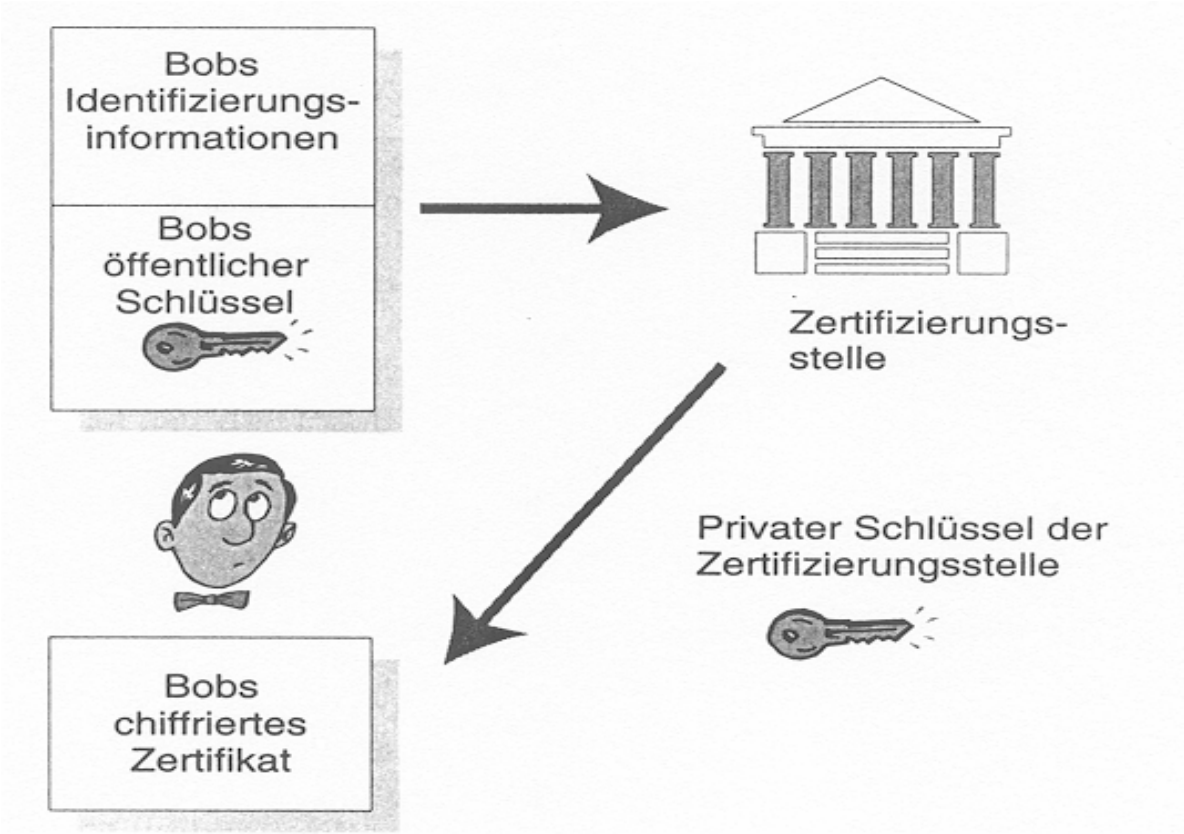
Key Management (Distribution and Certification) (4)

Solution: Using a trustworthy third person, the so-called

Certification Authority (CA)

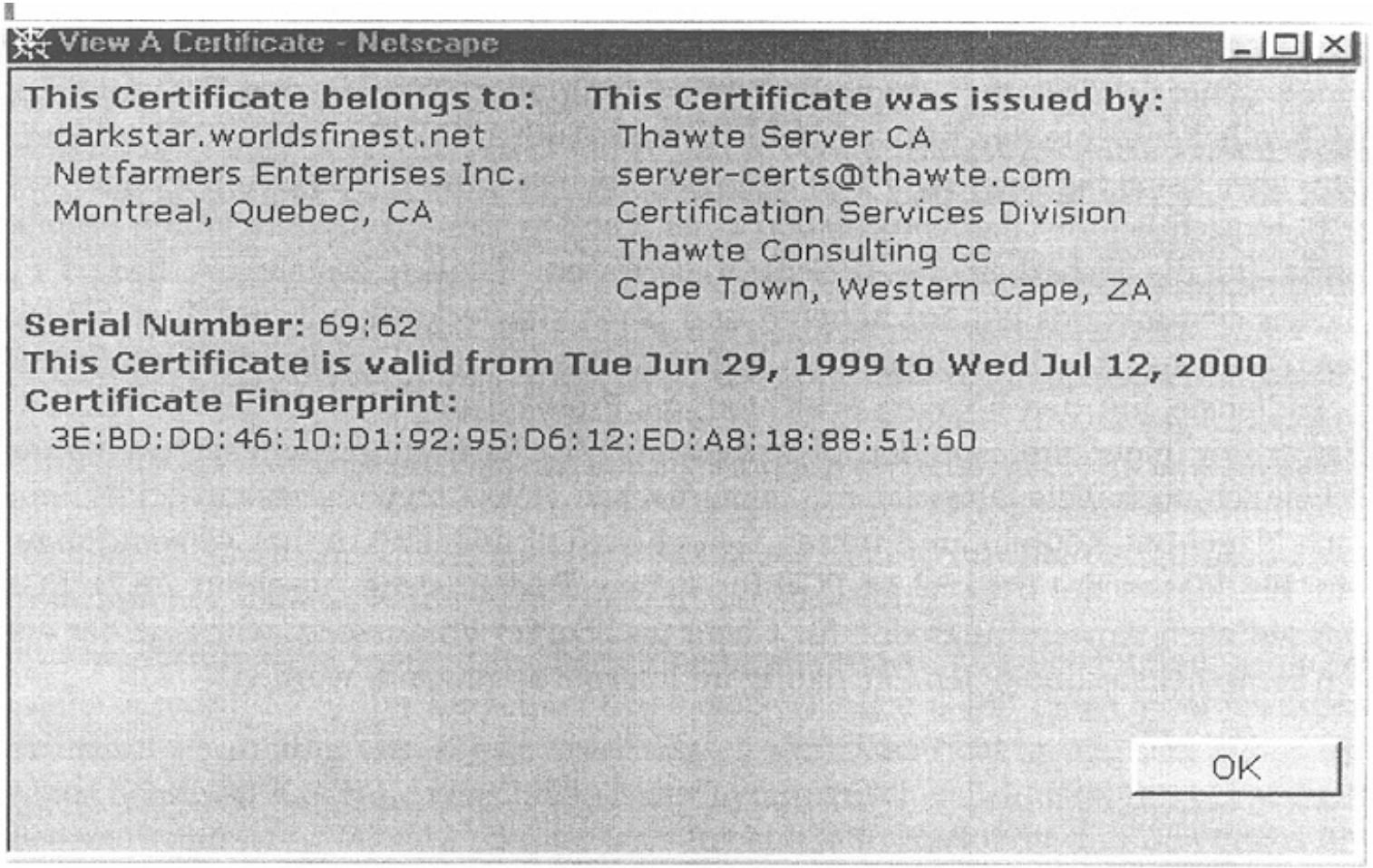
Idea: CA checks the identity of public key holders and creates a *certificate* which binds the key to the correct holder and is digitally signed by the CA.

Job of the CA



Key Management (Distribution and Certification) (4a)

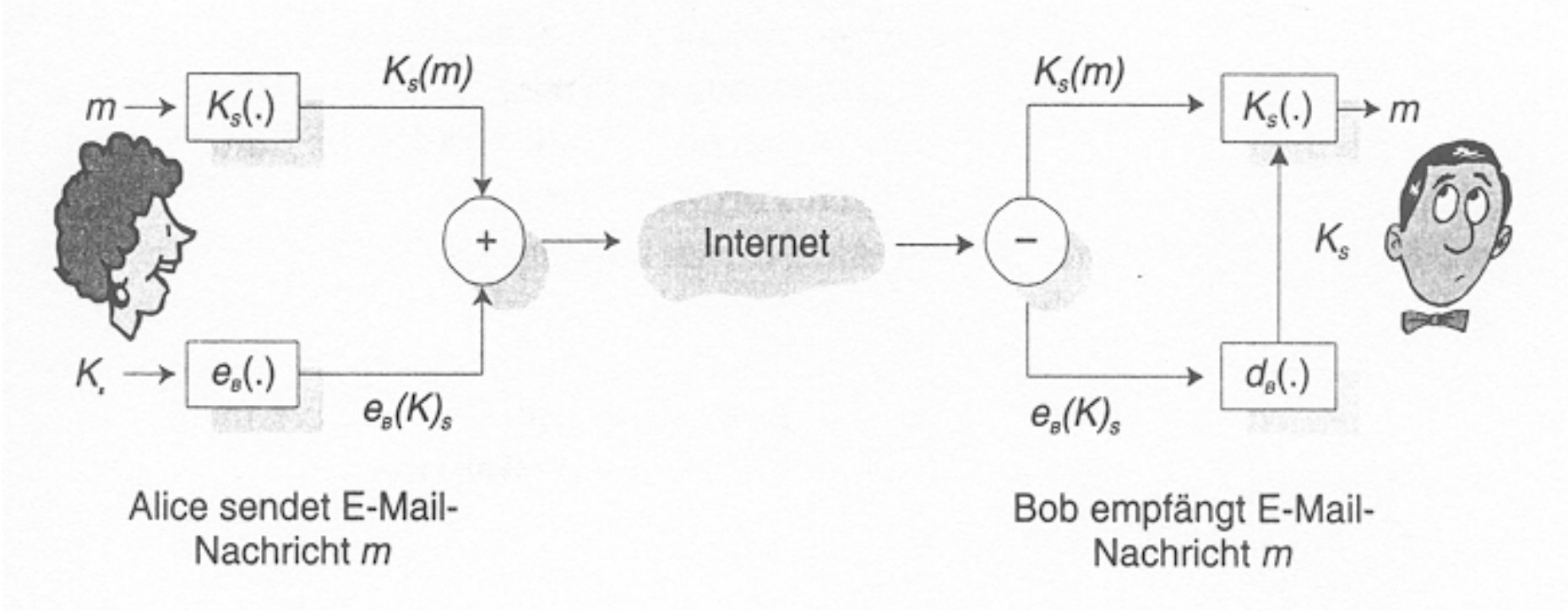
Example of a Certificate



Application layer: Secure E-Mail (1)

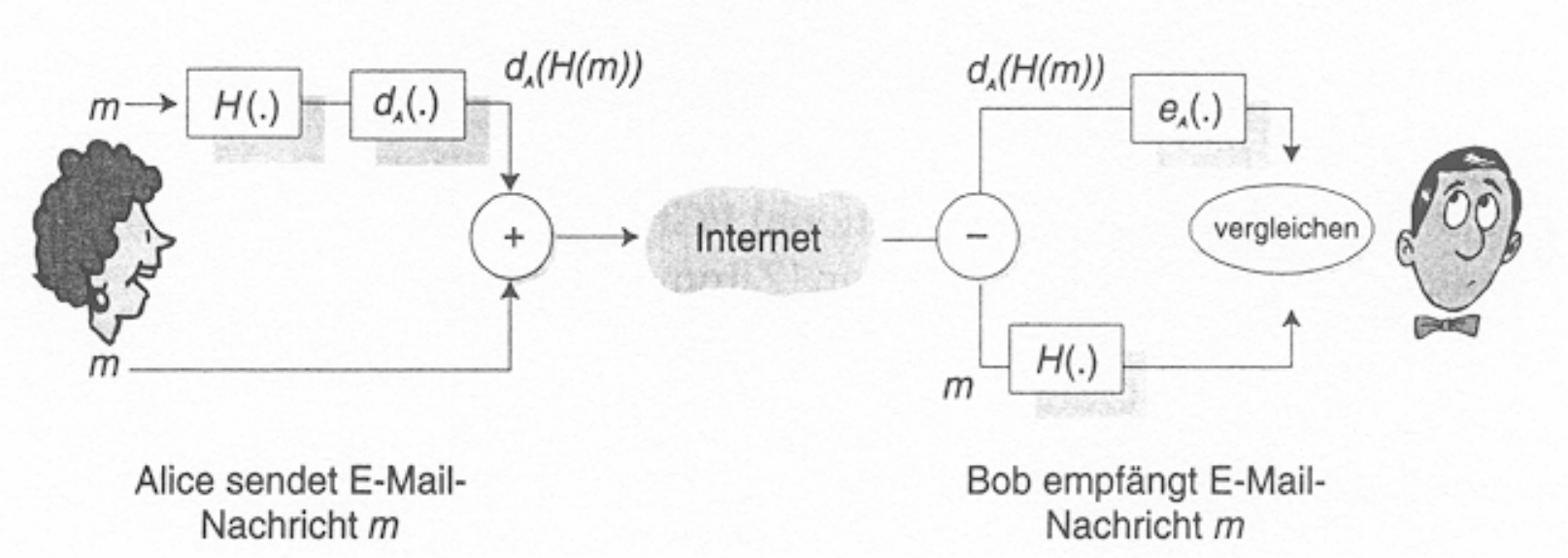
Desired security functionality: secrecy, authentication, message integrity

Secrecy approach: Symmetric session key encrypted by RSA (public key algorithm)

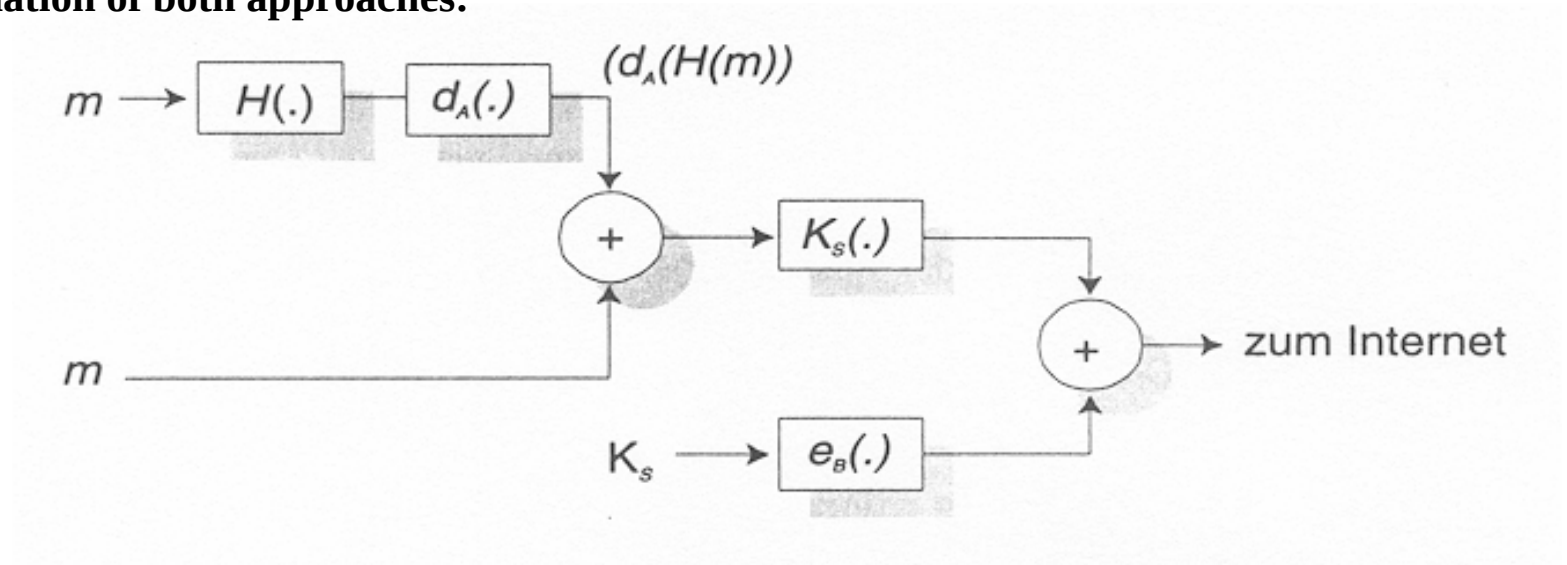


Application layer: Secure E-Mail (2)

Approach to sender authentication and integrity: Message Digests and digital signature



Combination of both approaches:



Application layer: Secure E-Mail (3)

De facto Standard: PGP (Pretty Good Privacy)
reflects in principle the approach just described

A message signed with PGP

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
Bob:
My husband is out of town tonight.
Passionately yours, Alice
-----BEGIN PGP SIGNATURE-----
Version: PGP for Personal Privacy 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZP9t6n7G6m5Gw2
-----END PGP SIGNATURE-----
```

A secret PGP message:

```
-----BEGIN PGP MESSAGE-----
Version: PGP for Personal Privacy 5.0
u2R4d+/jKmn8Bc5+hgDsqAewsDfrGdszX68liKm5F6Gc4sDfcXyt
RfdS1OjuHgbcfDssWe7/K=1KhnMikLoO+1/BvcX4t==Ujk9PbcD4
Thdf2awQfgHbnmKlok8iy6gThlp
-----END PGP MESSAGE-----
```