Assignment 3 - Cryptography

Information & Communication Security
(SS 2015)

M.Sc. Fatbardh Veseli

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

Agenda

- Wrap up from Exercise 2
- Hash functions and digital signatures
- Caesar cipher
- Symmetric vs. asymmetric ciphers
- Stream ciphers (Vernam code)
- Wrap up: AES

Exercise 4: Role Based Access Control (RBAC)

Consider a simplified scenario in a bank and the concept of RBAC. In order to perform a change (transaction) on an account (to mandate deposits and withdrawals), a customer use his card to “unlock” the account (authorize the transaction). He can do this by being registered in the bank in the role of a “Customer” and bringing his chip-card (bank card) to a card reader. The account of this customer is then authorized (unlocked) during the duration of this session, and authorized subjects can perform changes to this account. In the following, this kind of account “unlocking” will be denoted as “authorization”.

Exercise 4 - RBAC (2)

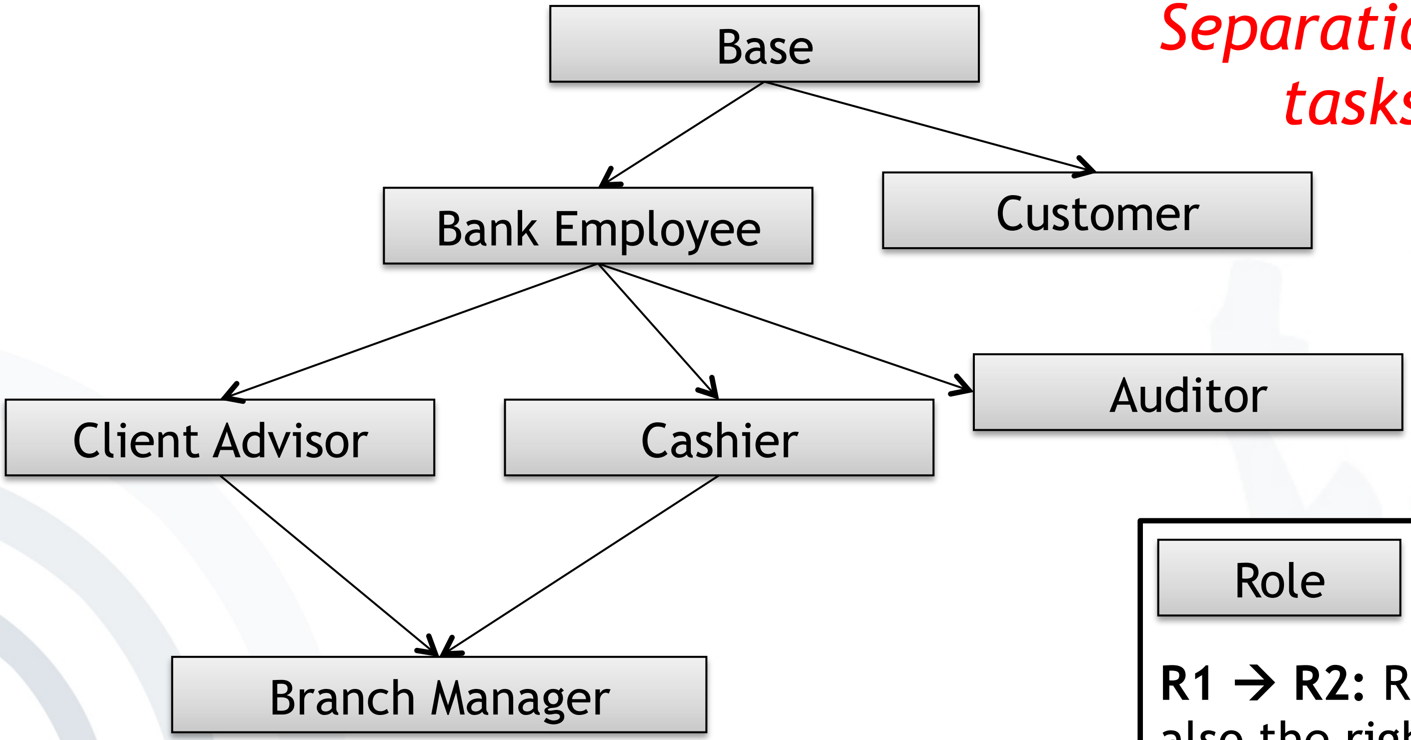
The following roles and their corresponding rights are valid in this scenario:

Role	**Rights**
Bank employee	Read all account data
Base	Read Terms of Use
Auditor	Perform audit
Branch Manager	Open and authorize account(s)' transactions (even without a chip card)
Cashier	Change an authorized account
Client Advisor	Open bank account
Client	Authorize own account

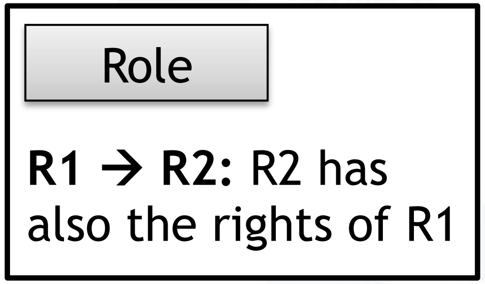
Exercise 4 – RBAC (3)

Roles: Bank employee, Base, Auditor, Branch Manager, Cashier, Client Advisor, Client.

a) draw a role-based access control diagram for this scenario



Separation of tasks



Exercise 4 – RBAC (4)

Roles: Bank employee, Base, Auditor, Branch Manager, Cashier, Client Advisor, Client.

b) The subject Cash machine (ATM) has the role Cashier. Can the ATM from this function perform the following:

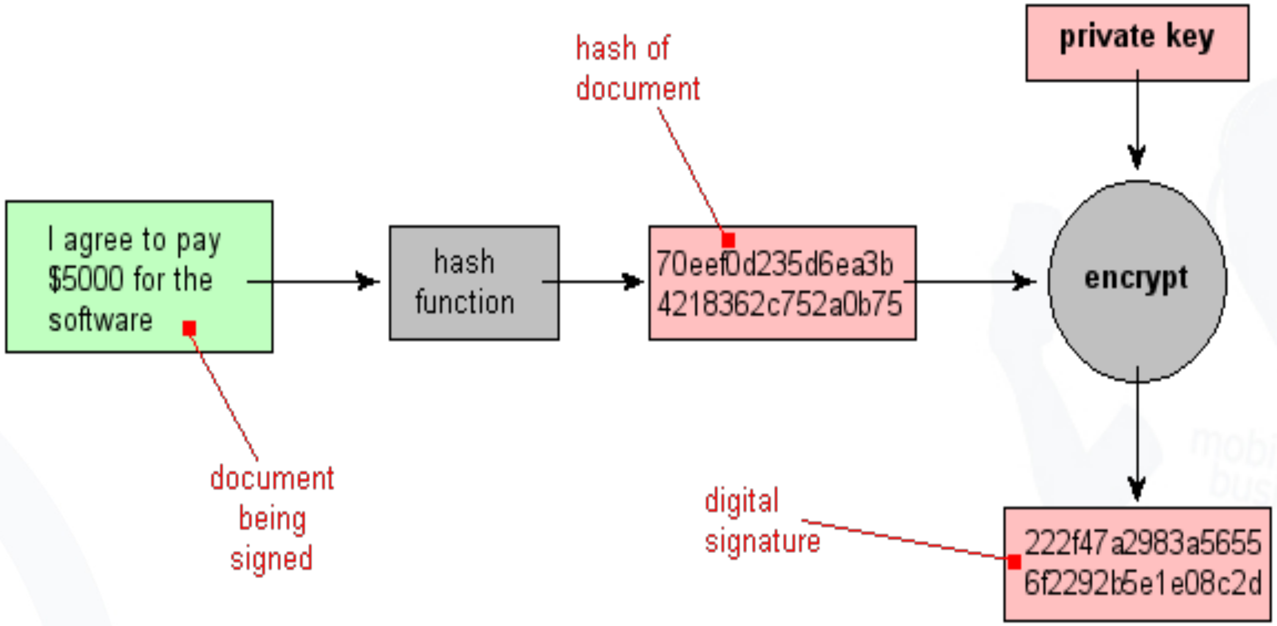
- Withdraw cash from an authorized account: ✔
- Withdraw cash from an unauthorized account: ✖
- Show account balance: ?

Exercise 1: PGP

- Install PGP Email Desktop (trial version) or a similar software for mail encryption on your system. Create a new key pair, and send a signed and encrypted message to fatbardh.veseli@m-chair.de containing your newly created public key and a short summary of your experiences.
- PGP can be downloaded from http://www.symantec.com/business/desktop-email
  - Practical exercise, no solution here, check lecture notes for overview of PGP
  - Be careful to only send your public key
  - You can also send your existing public key, but in this case be extra careful
  - If you haven't done this yet, try it, sending encrypted mail is useful, and we want you to be able to do it.

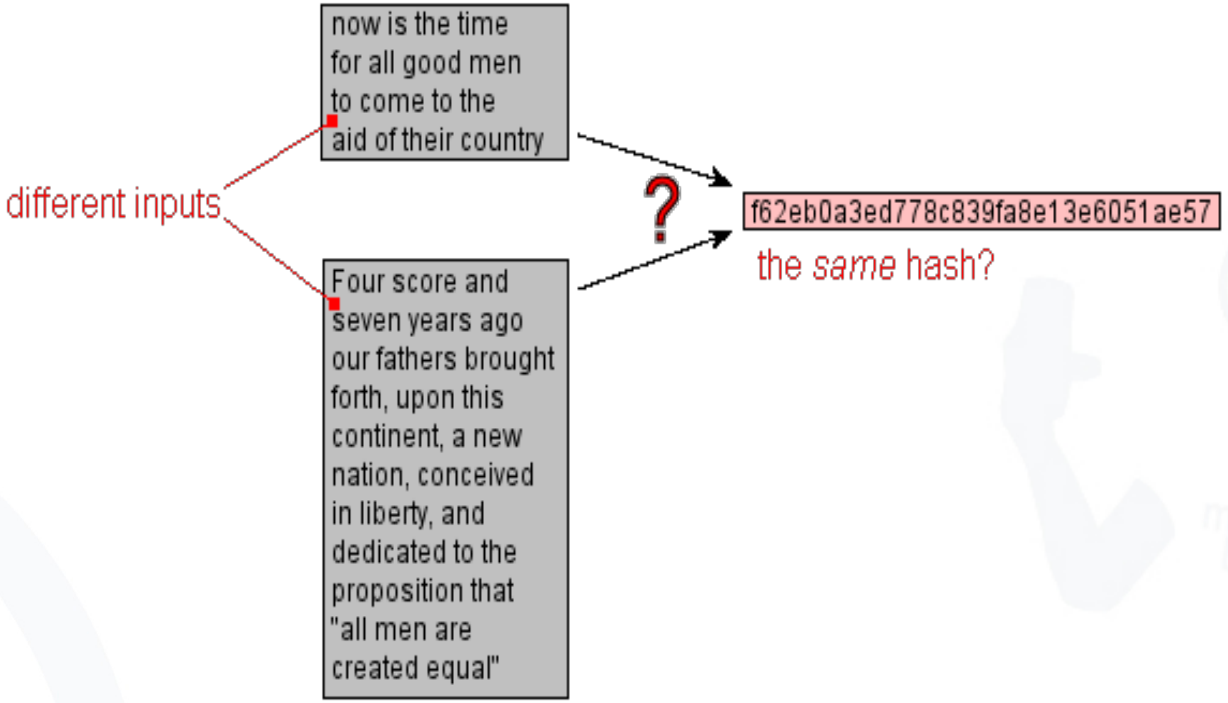
Exercise 2: Hash functions and signature systems

- The image below shows the steps of digitally signing a document. The sender receives the plain document and the digital signature.


hash of document
private key
I agree to pay $5000 for the software
hash function
70eef0d235d6ea3b 4218362c752a0b75
encrypt
document being signed
digital signature
222f47a2983a5655 6f2292b5e1e08c2d

Exercise 2: Hash functions and Signature Systems (2)

- When two different inputs produce the same hash value - collision



now is the time for all good men to come to the aid of their country

Four score and seven years ago our fathers brought forth, upon this continent, a new nation, conceived in liberty, and dedicated to the proposition that "all men are created equal"

Exercise 2: Hash functions and Signature Systems (3)

- Given a fixed message m1, if we cannot find in a practical way a different message m2 such that `hash(m2) = hash(m1)`, then we say that this hash function is *collision-resistant*.

  a. In the digital signature scheme, why do we produce the signature on the hash of the document and not on the document directly?

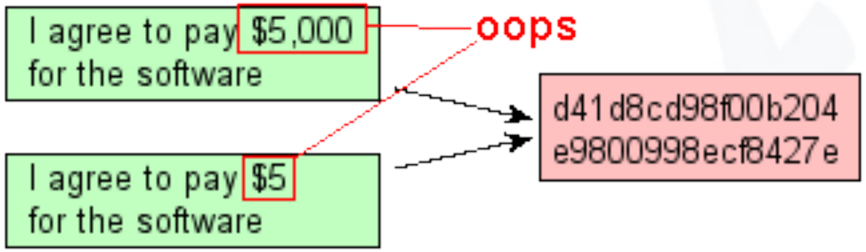
efficiency	integrity
The signature will be much shorter and thus save time since hashing is generally much faster than signing in practice.	Without the hash function, the text "to be signed" may have to be split (separated) in blocks small enough for the signature scheme to act on them directly. However, the receiver of the signed blocks is not able to recognize if all the blocks are present and in the appropriate order.

Exercise 2: Hash functions and Signature Systems (4)

- Given a fixed message `m1`, if we cannot find in a practical way a different message `m2` such that `hash(m2) = hash(m1)`, then we say that this hash function is *collision-resistant*.

b. Why is it important that hash functions are collision-resistant?

- In some digital signature systems, a party attests to a document by publishing a public key signature on a hash of the document.
  - If it is possible to produce two documents with the same hash, an attacker could get a party to attest to one, and then claim that the party had attested to the other.
- Software version comparison. An attacker who could produce two files with the same hash could trick users into believing they had the same version of a file when they in fact did not.


I agree to pay $5,000 for the software
oops
d41d8cd98f00b204 e9800998ecf8427e
I agree to pay $5 for the software

Exercise 3: Caesar Cipher

- Break the following ciphertext, given that the Caesar cipher was used to produce it is:

NZIVSNCZB QA QV OMZUIVG

- (Hint: Start by a permutation of the alphabet by 1, then 2, … until the result makes sense in English)

Caesar Cipher

Ciphertext: **NZIVSNCZB QA QV OMZUIVG**

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

- We assign a number for every character.
- This enables us to calculate with letters as if they were numbers.

Caesar Cipher

- For k ϵ {0..25} we have:
  - An encryption function:
    - e: x -> (x+k) mod 26
  - A decryption function:
    - d: x -> (x-k) mod 26
  - In this case $k_e = k_d$

Caesar Cipher

- Let‘s try:

Key	N	Z	I	V	S	N	C	Z	B		Q	A
1	M	Y	H	U	R	M	B	Y	A		P	Z
2	L	X	G	T	Q	L	A	X	Z		O	Y
3	K	W	F	S	P	K	Z	W	Y		N	X
4	J	V	E	R	O	J	Y	V	X		M	W
5	I	U	D	Q	N	I	X	U	W		L	V
6	H	T	C	P	M	H	W	T	V		K	U
7	G	S	B	O	L	G	V	S	U		J	T
8	**F**	**R**	**A**	**N**	**K**	**F**	**U**	**R**	**T**		**I**	**S**

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

The plaintext

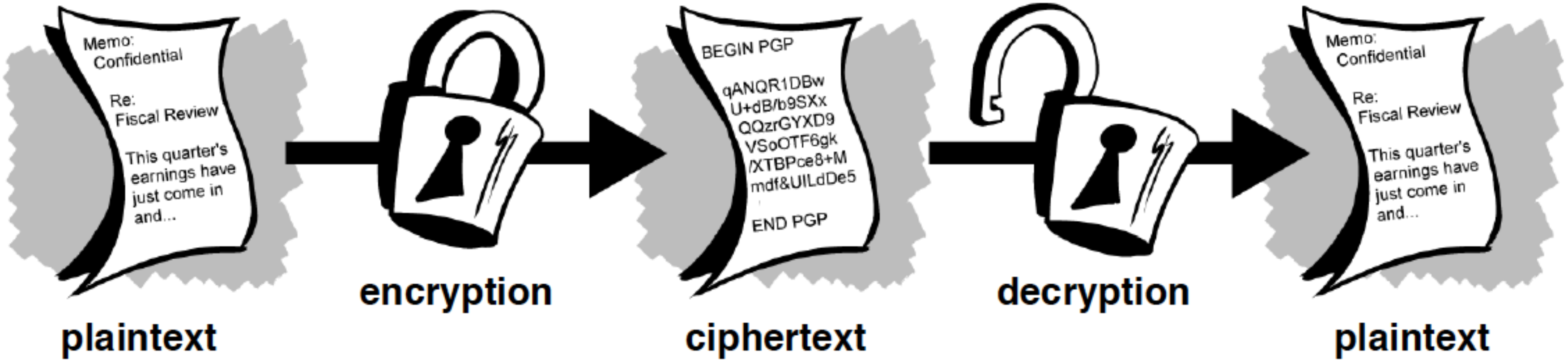
- The key is 8
- The plain text is:

FRANKFURT IS IN GERMANY

Assessment of Caesar Cipher

- Very simple form of encryption.
- The encryption and decryption algorithms are very easy and fast to compute.
- It uses a very limited key space (n=26)
- Therefore, the encryption is very easy and fast to compromise.

Encryption - Decryption

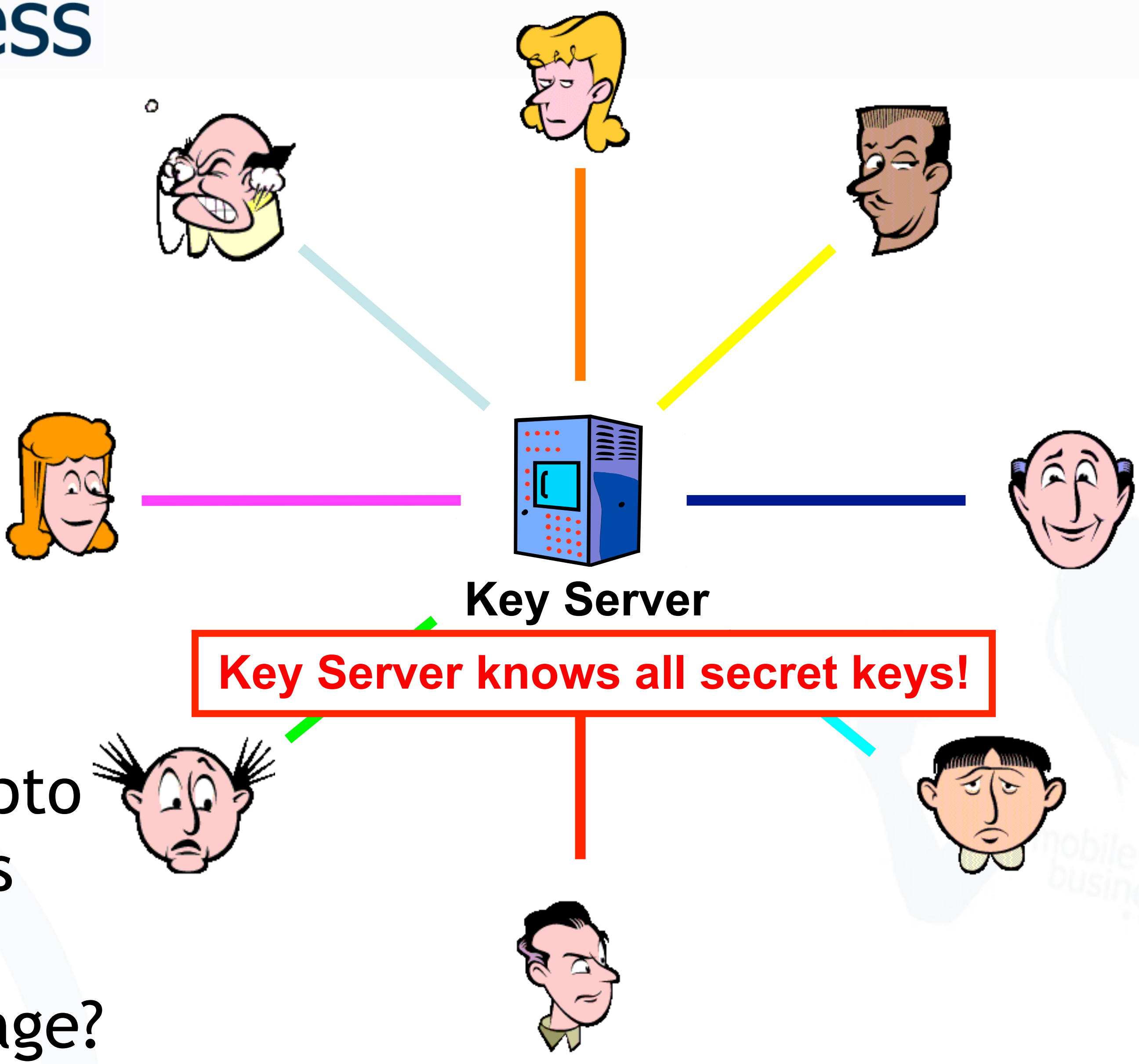


http://www.pgpi.org/doc/guide/6.5/en/intro/

Exercise 4: Misc

a. What is the difference between symmetric and asymmetric crypto systems?

Exercise 4: Misc (a4)

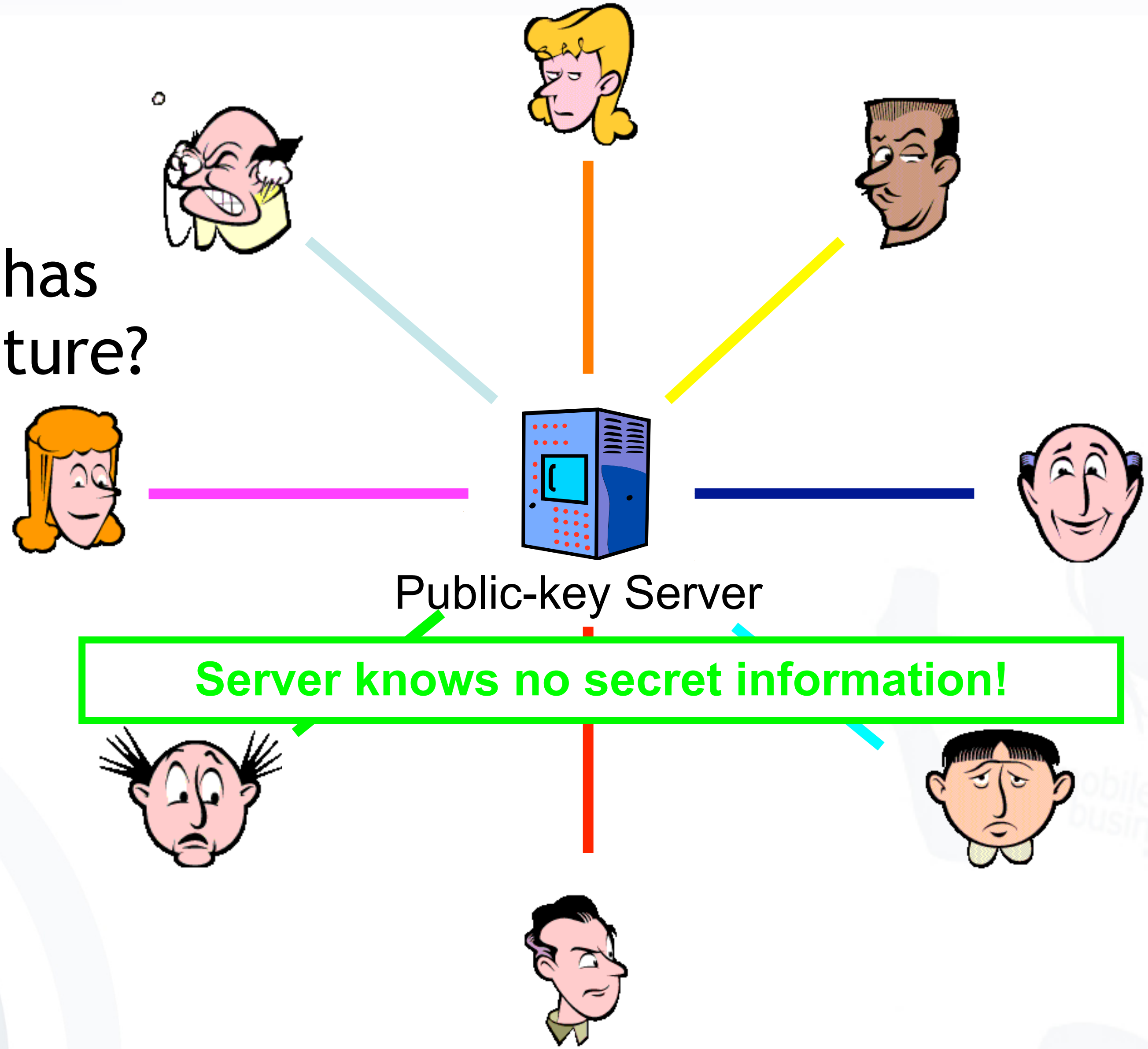


Key Server knows all secret keys!

Which crypto system has this disadvantage?

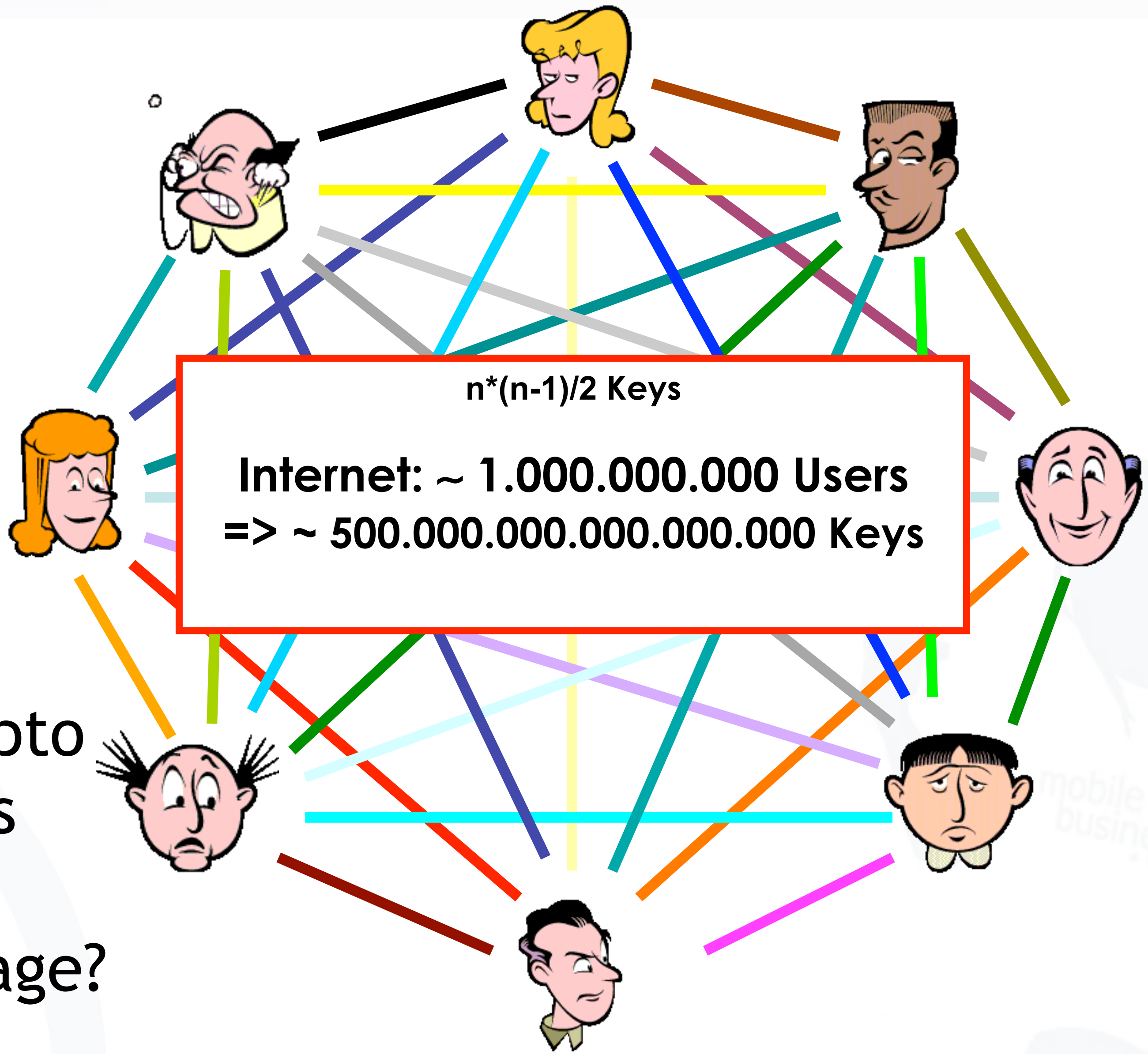
Exercise 4: Misc (a3)

Which crypto system has this feature?



Server knows no secret information!

Exercise 4: Misc (a4)

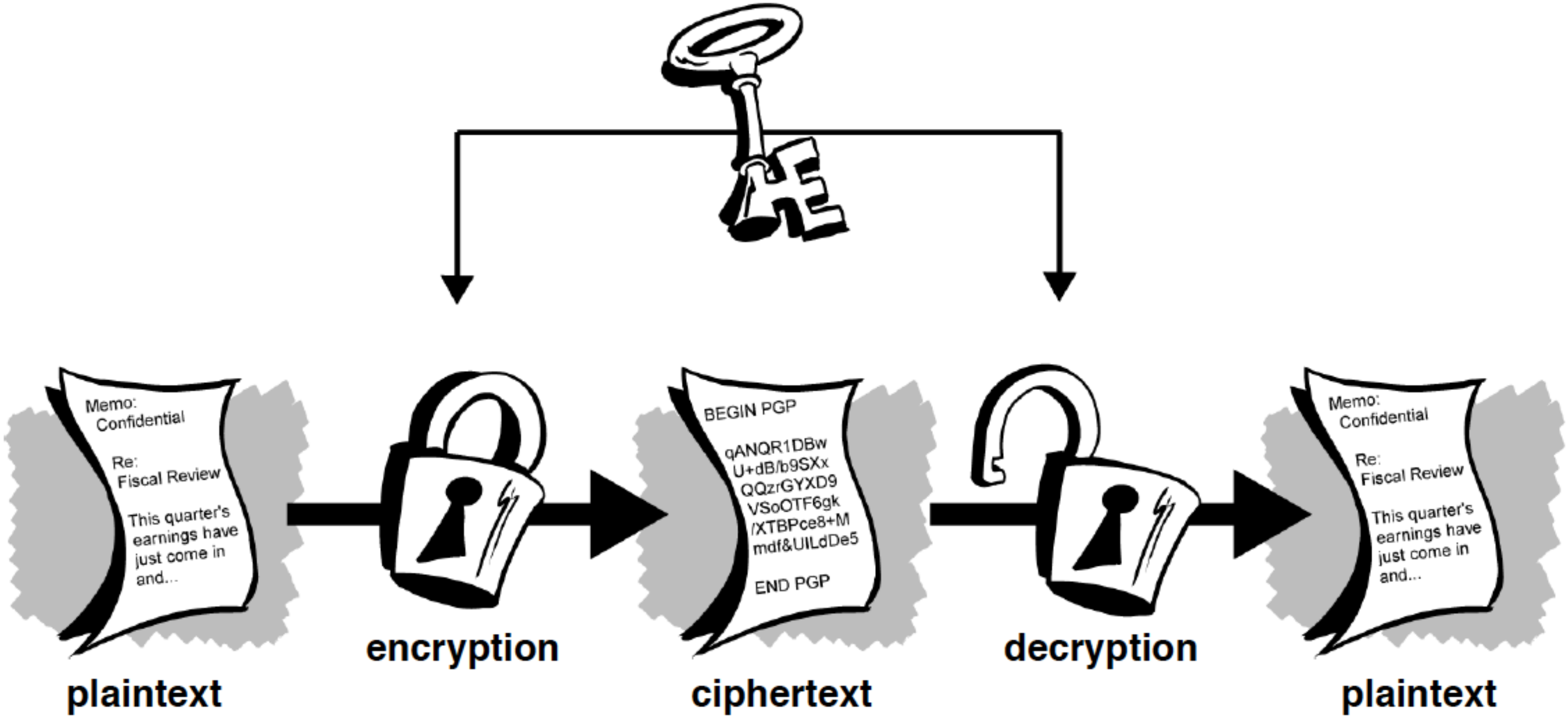


n*(n-1)/2 Keys

Internet: ~ 1.000.000.000 Users
=> ~ 500.000.000.000.000.000 Keys

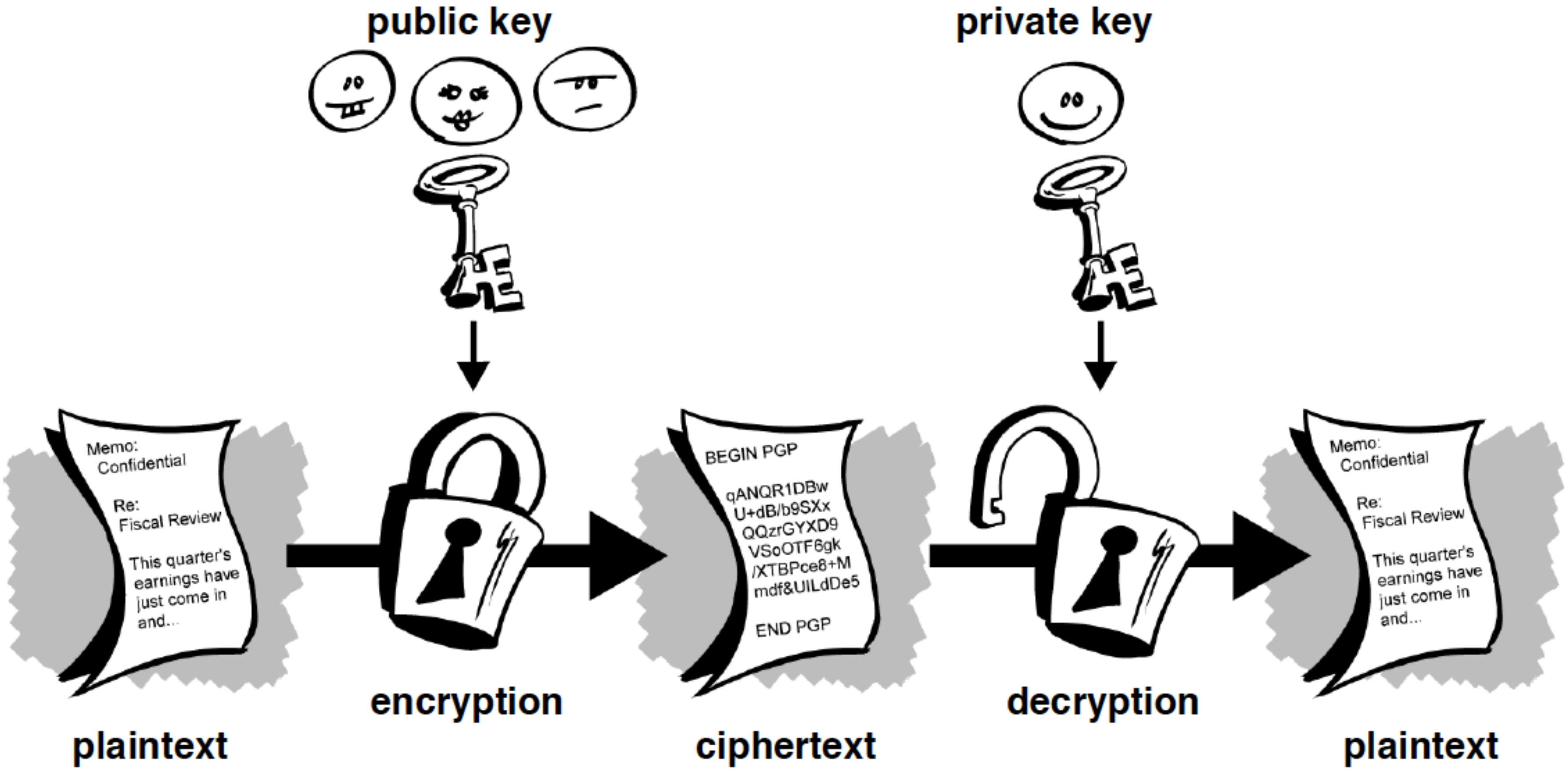
Which crypto system has this disadvantage?

Guess which crypto system this is



Symmetric or Asymmetric?

This crypto system is…?



Symmetric or Asymmetric?

Symmetric Encryption

Advantage: Algorithms are very fast

Algorithm	Performance*
RC6	78 ms
SERPENT	95 ms
IDEA	170 ms
MARS	80 ms
TWOFISH	100 ms
DES-ede	250 ms
RIJNDEAL (AES)	65 ms

*** Encryption of 1 MB on a Pentium 2.8 GHz, using the FlexiProvider Java)**

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

Performance of Public Key Algorithms

Algorithm	Performance*	Performance compared to Symmetric encryption (AES)
RSA (1024 bits)	6.6 s	Factor 100 slower
RSA (2048 bits)	11.8 s	Factor 180 slower

Disadvantage: Complex operations with very big numbers

⇒ Algorithms are very slow

*** Encryption of 1 MB on a Pentium 2.8 GHz, using the FlexiProvider (Java)**

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

Summary 4: Misc (a)

a. Differences between symmetric and asymmetric cryptosystems.

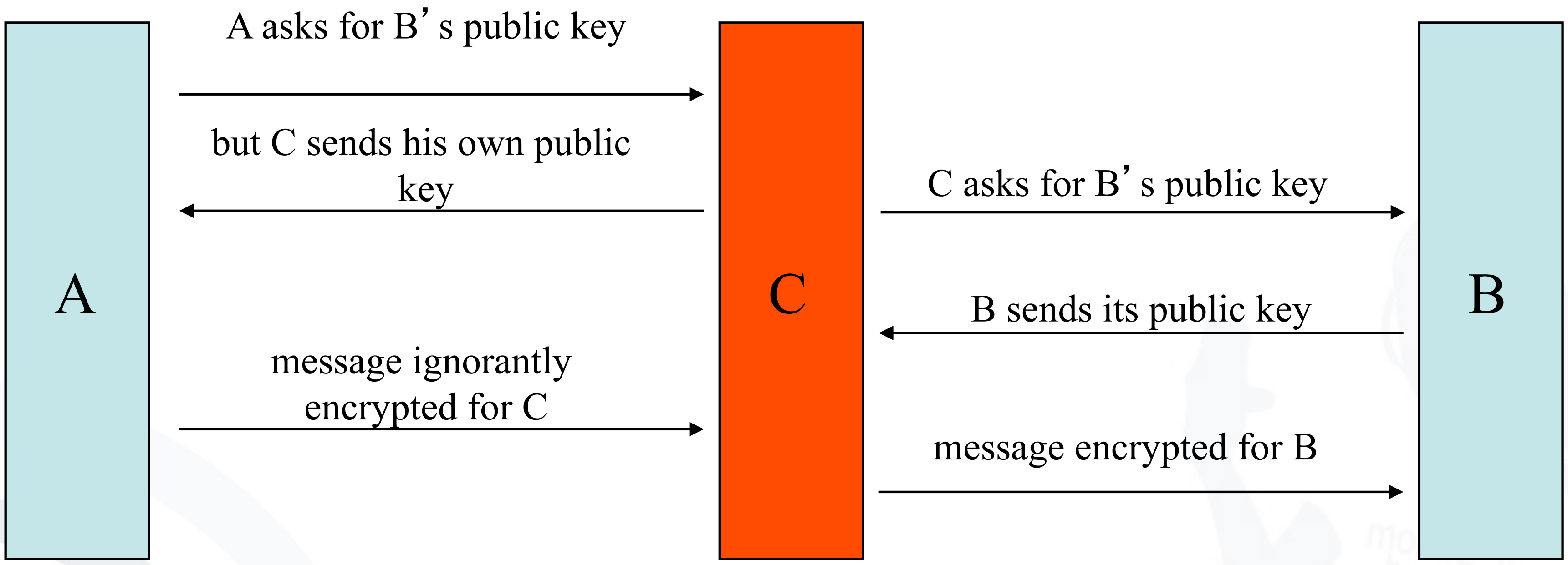
Symmetric	Asymmetric
Both encryption and decryption is done with the same key.	Encryption with public key, decryption with private key.
One key per communication pair is necessary.	Does not require a secure communication channel. Public key can be freely distributed.
Efficient in terms of performance	Less efficient
Keys have to be kept secret	Only keep own private key secret
Secure agreement and transfer are necessary.	Does not require agreement on a shared key.
A center for key distribution is possible but this party then knows all secret keys!	A center for key distribution is possible and this party does not know the secret keys.

Exercise 4: Misc (b)

b. Why is certification of public key necessary? Name an attack that is possible if keys are not certified.

Exercise 4: Misc (b1)

What is the name of this attack?


A
C
B
A asks for B's public key
but C sends his own public key
message ignorantly encrypted for C
C asks for B's public key
B sends its public key
message encrypted for B

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

Exercise 4: Misc

What are advantages and disadvantages of asymmetric crypto systems?

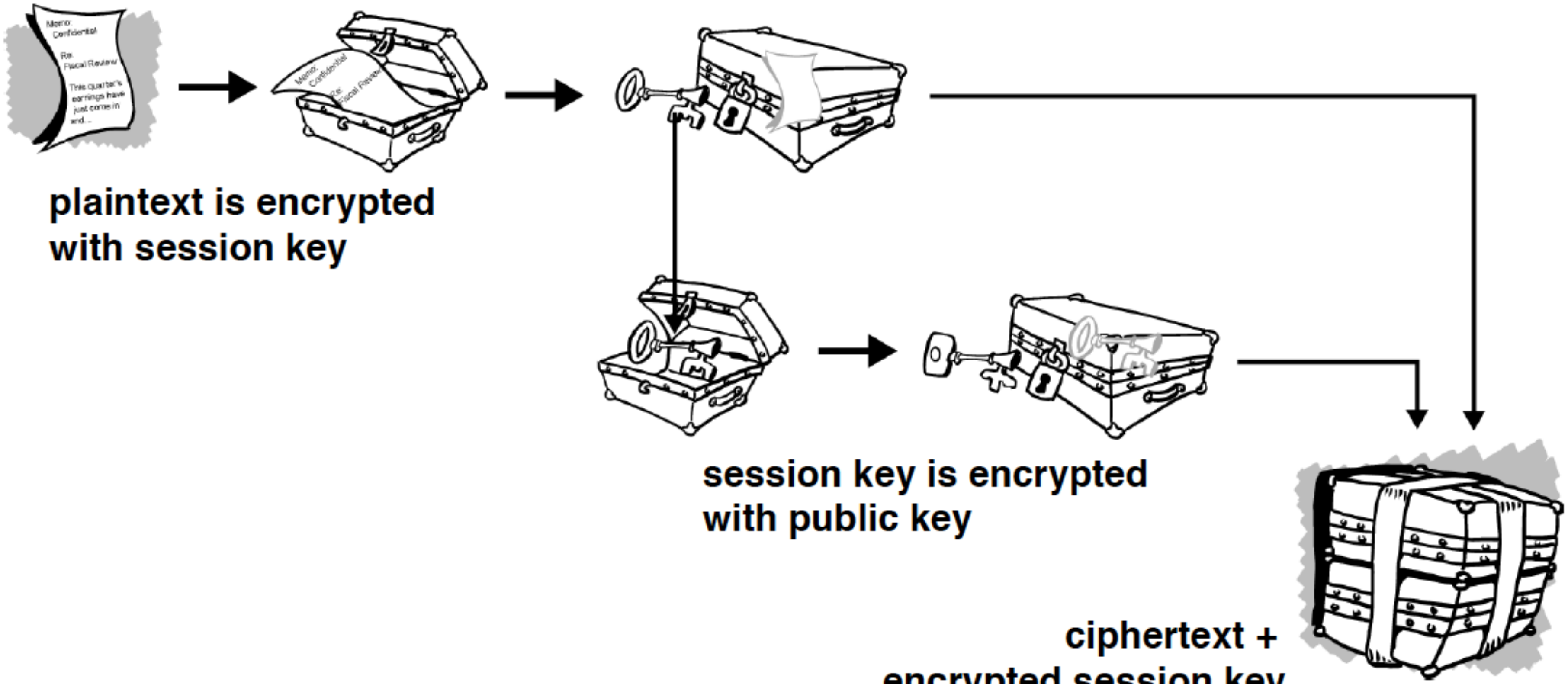
Advantages:

- No secret must be shared
- Only one key per endpoint

Disadvantages:

- Algorithms are very slow
- Man-in-the-middle-attack

Hybrid encryption: PGP Example Encryption

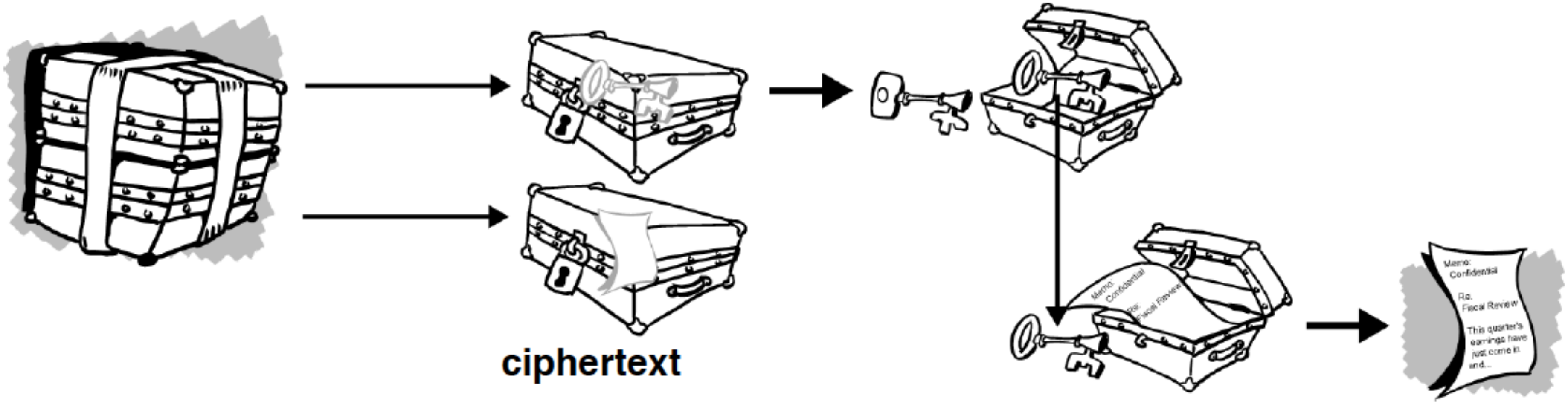


plaintext is encrypted with session key

session key is encrypted with public key

ciphertext + encrypted session key

PGP Decryption



encrypted message

encrypted session key

ciphertext

recipient's private key used to decrypt session key

session key used to decrypt ciphertext

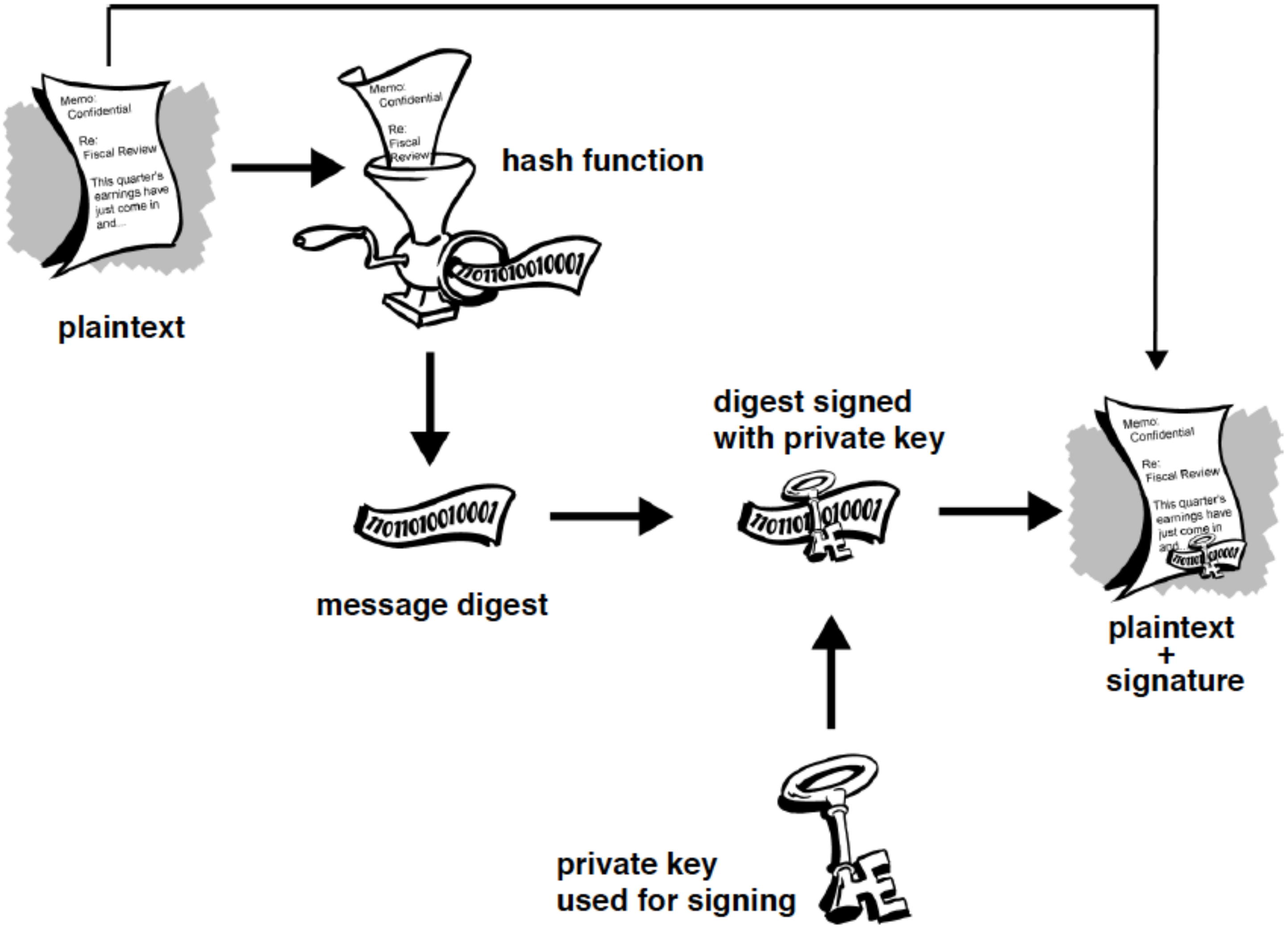
original plaintext

PGP example continued…

- Encryption offers
  - Confidentiality

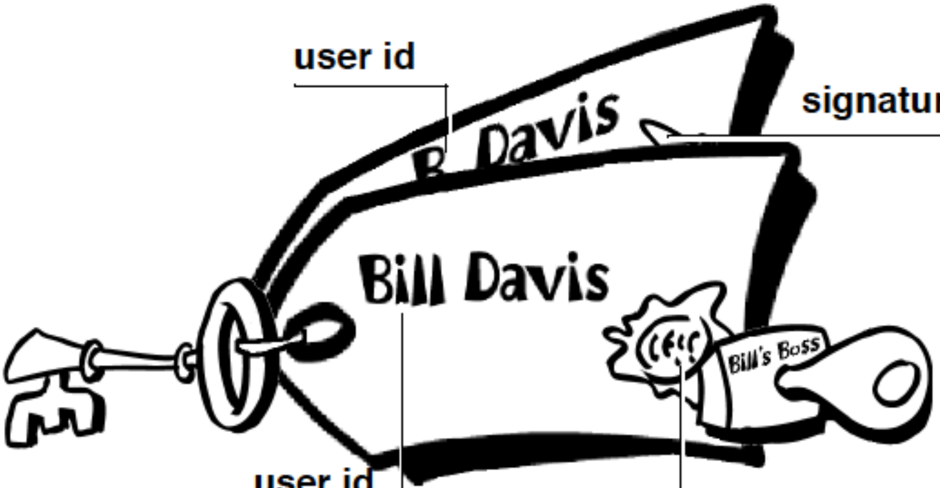
- Digital Signatures offer
  - Authentication
  - Integrity

Digital Signatures



Memo: Confidential
Re: Fiscal Review
This quarter's earnings have just come in and...
plaintext
hash function
11011010010001
message digest
digest signed with private key
private key used for signing
plaintext + signature

PGP Digital Certificates



public key

- PGP version number
- time when key created
- how long key is valid
- key type (DH, RSA)
- the key material itself

user id

signature

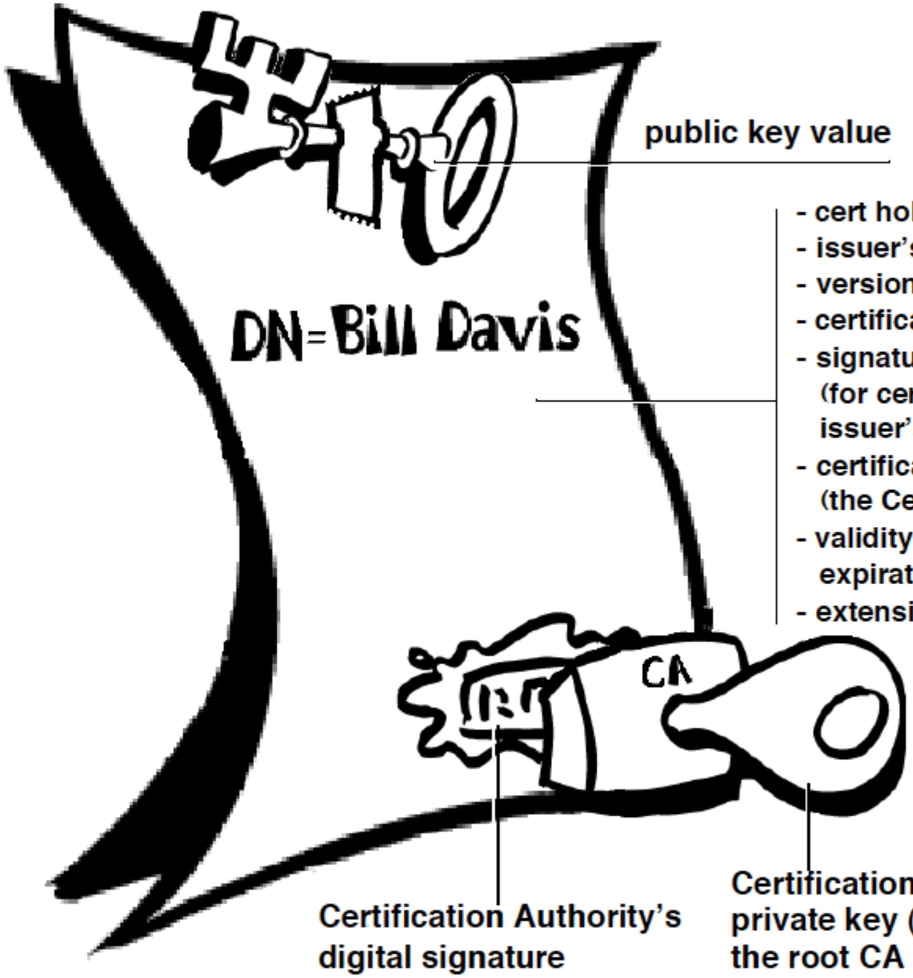
user id

- string identifying the key's owner

signature

- certification that the userid and key go together
- version number
- message digest algorithm
- message digest calculation
- signed message digest
- signer key id

X.509 Digital Certificates



public key value

- cert holder’s unique name (DN)
- issuer’s unique name
- version of cert. format
- certificate serial number
- signature algorithm identifier (for certificate issuer’s signature)
- certificate issuer’s name (the Certification Authority)
- validity period (start/ expiration dates/times)
- extensions

Certification Authority’s digital signature

Certification Authority’s private key (also called the root CA certificate)

Exercise 5: Stream ciphers

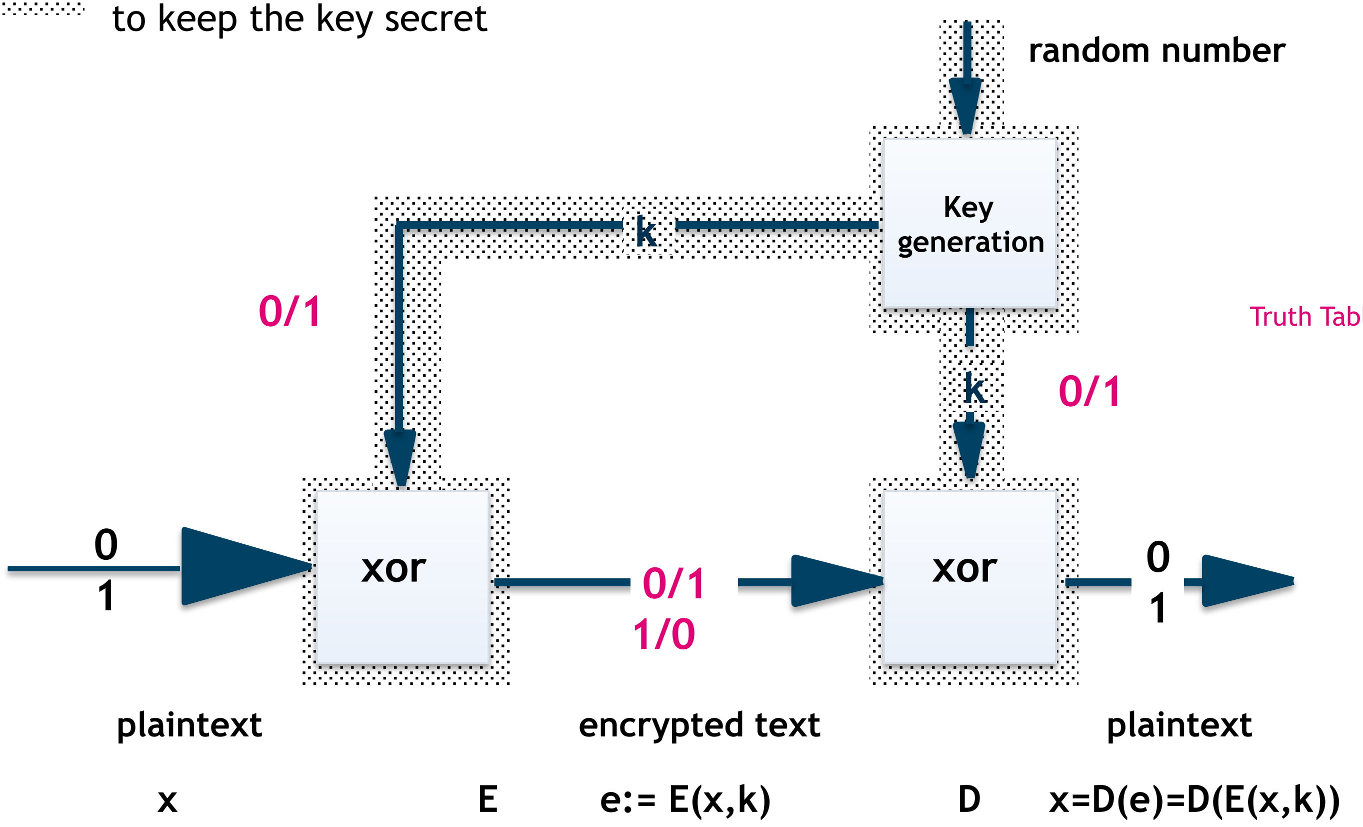
a) What is a one-time pad (Vernam-code)?

One Time Pad

- Invented by Gilbert Vernam
- The one-time pad is basically a Vigenére cipher.
- The length of the key is as long as the length of the plaintext.
- Therefore, there are no periodic reoccurrences.
- The key is randomly chosen and only used once.
- Every key has the same probability.

Example One Time Pad

▨ area that needs to be protected to keep the key secret



random number

Key generation

k

k

0/1

0/1

0

1

xor

0/1

1/0

xor

0

1

plaintext

encrypted text

plaintext

x

E

e:= E(x,k)

D

x=D(e)=D(E(x,k))

X_i	S_i	Y_i
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table of the XOR operation

[based on Federrath and Pfitzmann 1997]

Example

PT=	0	1	1	0
k=	1	0	1	1

X_i	S_i	Y_i
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table of the XOR operation

a=	1	1	1	1
b=	1	0	1	1
c=	1	1	0	1

Exercise 5: Stream ciphers

- b) Alice wants to encrypt the letter A, where the letter is given in ASCII code. The ASCII value for A is $65_{10} = 1000001_2$. Using Vernam-code, which of the following keys are suitable to encrypt this plaintext:
  - b1) 10100110
  - [✓ b2) 0011111]
  - b3) 101010

X_i	S_i	Y_i
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table of the XOR operation

Exercise 5: Stream ciphers

- c) Encrypt the message using Vernam code and using XOR as an encryption function and the key in b).

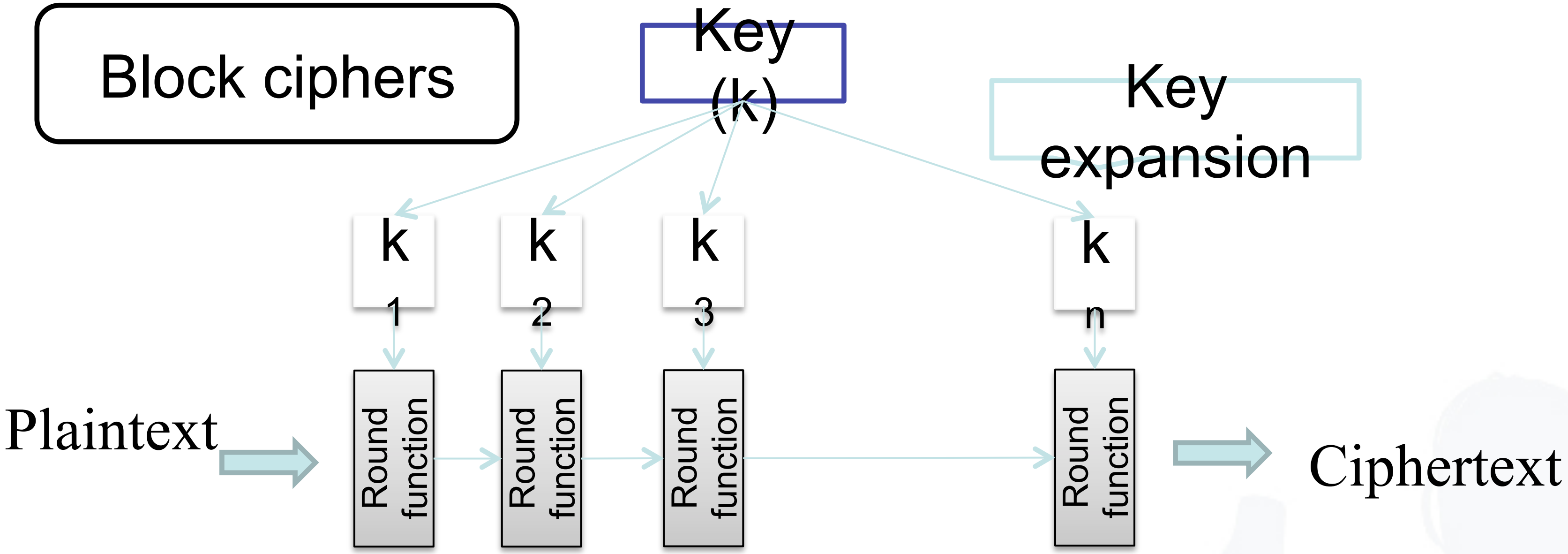
Plaintext (A)	1000001
Key (B)	0011111
Ciphertext (A xor B)	1011110

X_i	S_i	Y_i
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table of the XOR operation

EXAMPLE 2: AES

Advanced Encryption Standard



- Successor of DES, this new cipher is called Advanced Encryption Standard (AES).
- AES has been approved for Secret or even Top Secret information by the NSA.

[Bi2005]

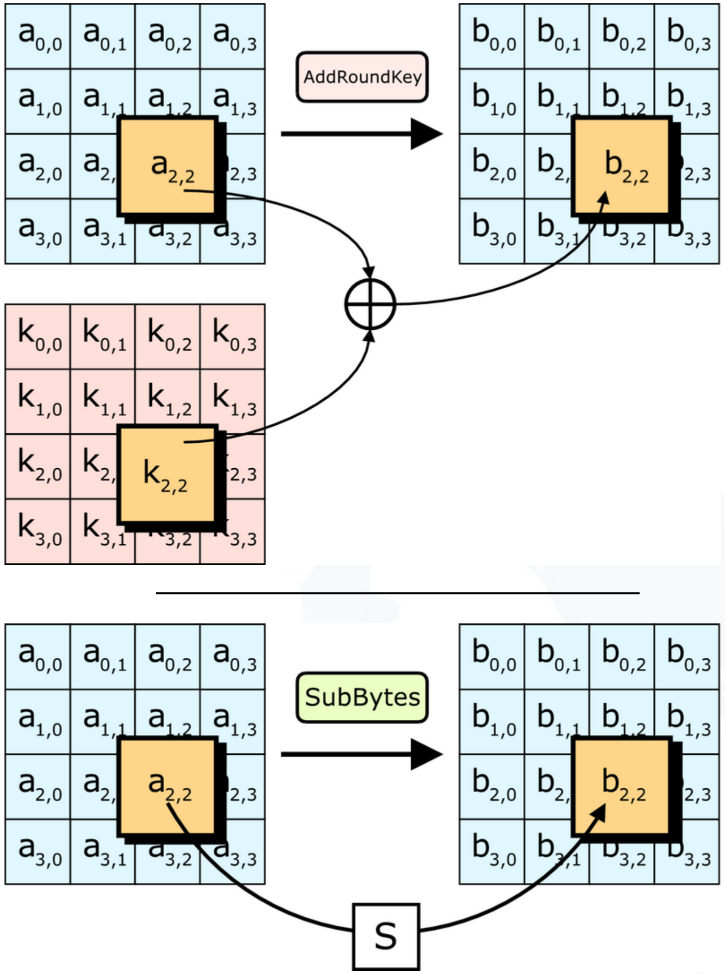
AES Encryption - Overview

- AES encryption
  - has a variable **number of rounds (10, 12, 14)**
  - depending on **key size (128-bit, 192-bit, 256-bit)**.
- To encipher a block of data in AES
  - Initialize (key schedule…)
    - Stretch key data
    - Initialization Round
  - Then several rounds of encryption
    - Shifting and mixing bits
  - Finally, some postprocessing
    - perform a round with the last step omitted

Encryption Round (1)

AES

- AddRoundKey
  - XOR (mix bits of) current state a and round key
  - Round key k derived using key schedule
- SubBytes
  - Substitution using a lookup table (S-Box)

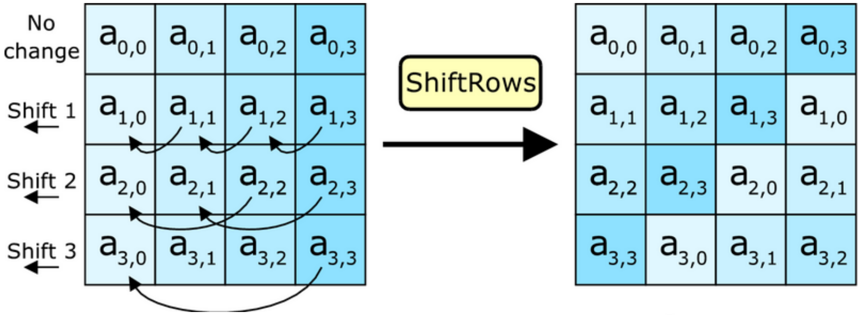


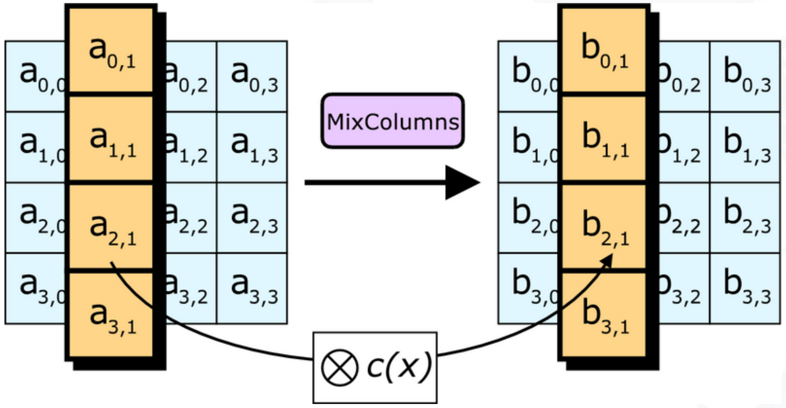
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
AddRoundKey
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$\oplus$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$
SubBytes
S

Encryption Round (2)

AES

- ShiftRows
  - Shift each row by row index
- MixColumns
  - 4 key bytes combined into each column using polynomial multiplication modulo 2^8 [in GF(2^8)]


No change
Shift 1
Shift 2
Shift 3
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
MixColumns
$\otimes c(x)$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

AES Decryption

- Works in the same way, but steps are performed in reverse order.
- Enables an easier implementation in hardware.

Thank you!

- Questions: sec@m-chair.de