Universität Stuttgart

Institute of Parallel and Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

Net-based Applications: Security

Adnan Tariq

(Based on the slides of Dr. Frank Dürr)

Network Security

- Objectives
- Attacks
- Cryptographic Techniques
  - Symmetric Encryption
  - Asymmetric Encryption
  - Secure Hash Functions
  - Digital Signatures
  - Certificates
- Java Security API

Objectives

1. Confidentiality
Data must be protected from unauthorized reading

2. Integrity
Data must be protected from unauthorized writing
“Writing” means: Insertion, modification, deletion, etc.

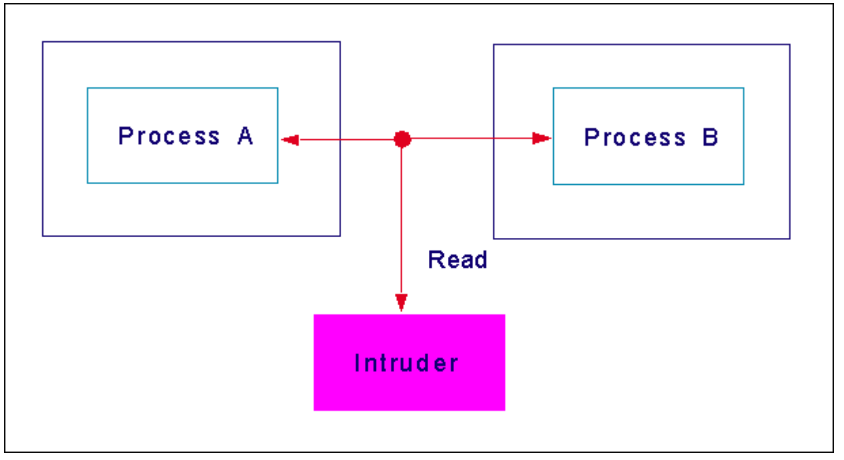
3. Authenticity
The authenticity of data must be guaranteed
i.e. the reported source of the data must be confirmed to be the real one

4. Non-repudiation

Either sender or receiver is prevented from denying a transmitted message

5. Availability
Protection of system resources form non-authorized access, so their availability to authorized users can be guaranteed
System resources: Processors, memory, communication channels, programs etc.

Passive Attacks

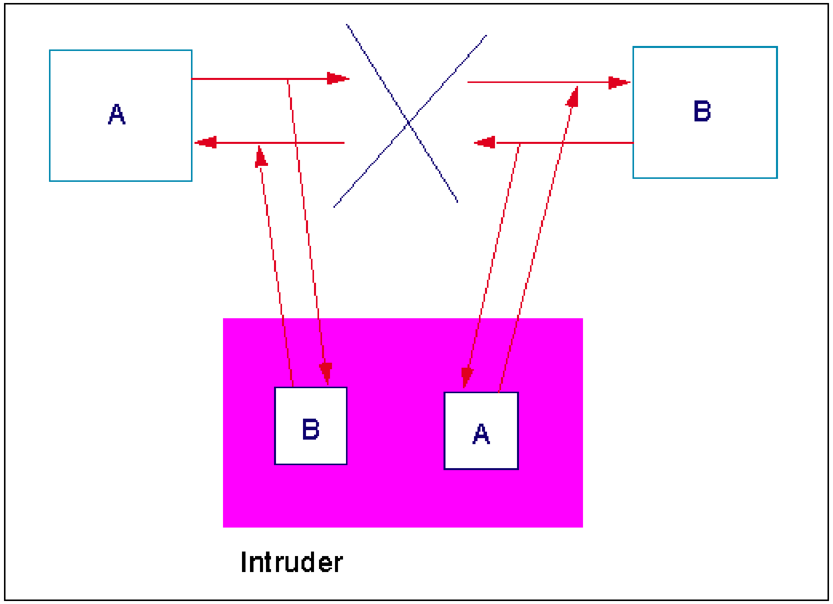


Passive Attacks:

- Reading of messages
  - e.g. by eavesdropping on the communication channel
- Analysis of the message traffic
  - Who communicates with whom?
  - How high is the message traffic at what times?
  - Length and number of messages?

⇨ Can draw conclusions about the message contents.

Active Attacks



Active Attacks:

- Editing the message stream
  - Changes to the messages, generation and deletion of messages
  - Changes to the order of messages
  - Recording and replaying of messages
  - Delay of messages
- Masquerade
  - Intruder disguises herself as an authorised user or program

Basic Design Approach

1. Identify **subjects** and **objects**
    - Subjects access objects
    - Subjects are: users, processes, etc.
    - Objects are: computers, programs, data, etc.

2. Create mechanisms that make it possible to assign access rights for certain objects to certain subjects
    - Access control (e.g. ACL, Capabilities)

3. Ensure that mechanisms cannot be circumvented by anyone
    - Cryptography

Cryptographic Techniques

Many Applications:

- Confidentiality and integrity of messages
- Authentication
  - of messages (digital signatures)
  - of users
- and more ...

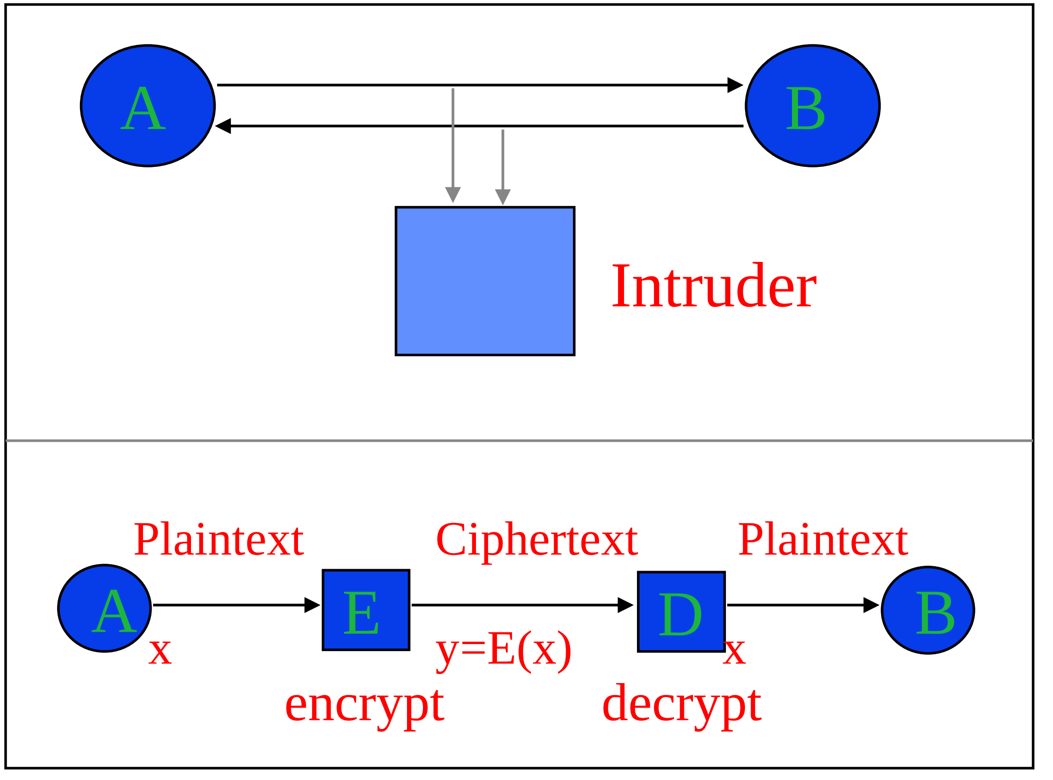
Two classes of algorithms:

- Symmetric Algorithms
  - Data Encryption Standard (DES)
  - Advanced Encryption Standard (AES)
- Asymmetric Algorithms
  - e.g. RSA (Rivest-Shamir-Adleman)

Cryptographic Techniques

- Attack

- Solution

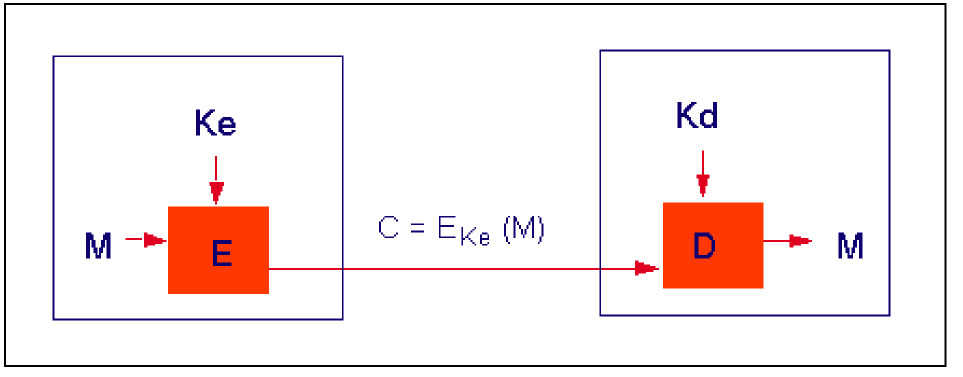


➢ Ensures confidentiality

Principle

Let

M	= message
K_e	= encryption key
K_d	= decryption key
E	= encryption algorithm
D	= decryption algorithm
C	= encrypted message



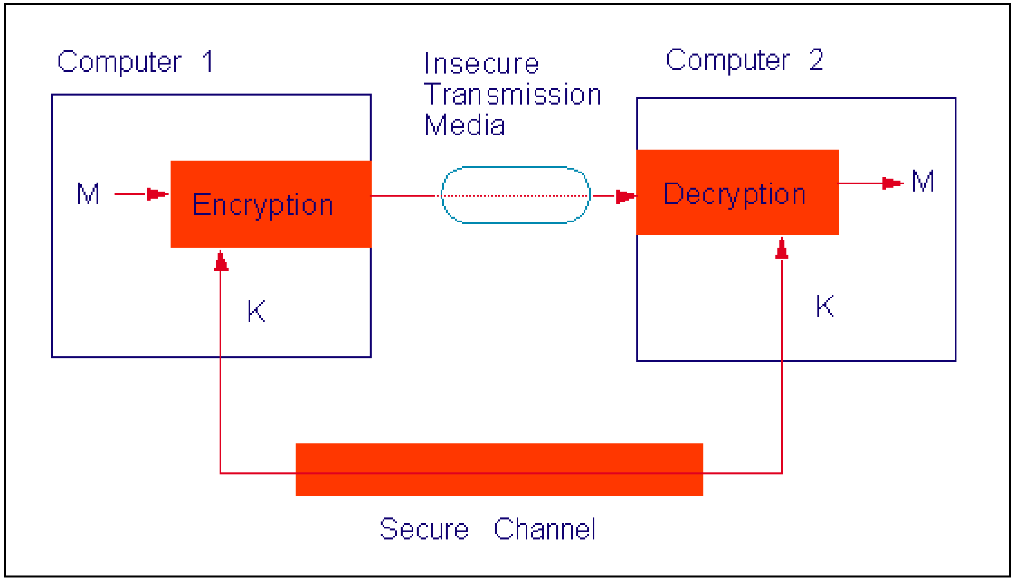
Then

$C = E_{Ke}(M)$ and $M = D_{Kd}(C)$

with

$D_{Kd}(C) = D_{Kd}(E_{Ke}(M)) = M$,
i.e. D is the inverse function to E

Symmetric Algorithms

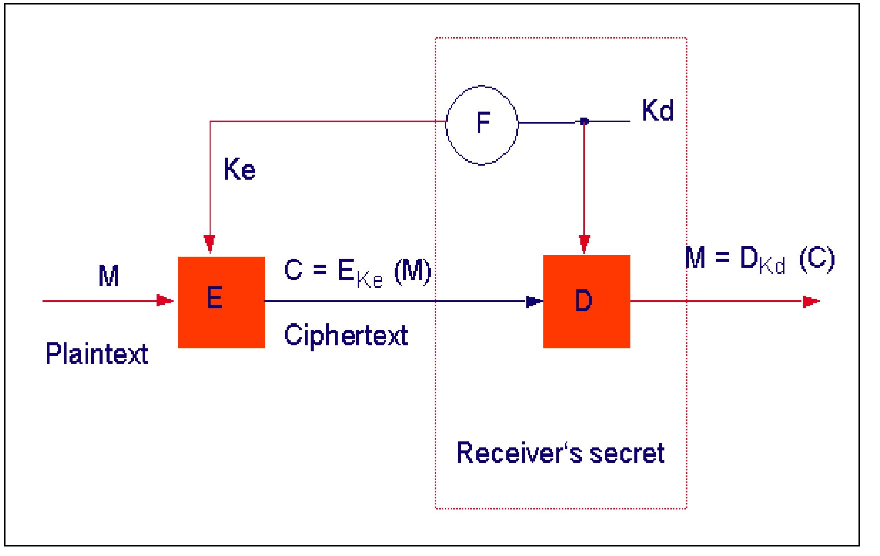


- Encryption and decryption with a **secret key** K
- $K = K_e = K_d$
- Key K must be exchanged over a secure channel

- **Examples: DES, AES**
- **Problem**: Key distribution

Asymmetric Algorithms (Public-Key Algorithms)

- Algorithms for E and D are public
- $K_e \neq K_d$ and K_e is public
  - ⇨ Key distribution is simplified
- K_e can be efficiently computed from K_d, but the computation of K_d from K_e is "practically impossible"
  - ⇨ F is a one-way function
- K_d is secret and $M = D_{Kd}(E_{Ke}(M))$ (i.e. D is the inverse function to E)



Only the receiver (who knows K_d) can read M

⇨ **Confidentiality**

RSA Algorithm

RSA was developed at MIT in 1978 by Rivest, Shamir and Adleman

It is based on principles of number theory

Trapdoor-function:

Multiplication of two large prime numbers

$\updownarrow$

Factoring the product

(Recent factoring record: RSA-768 challenge (232 decimal digits) took approx. 2000 CPU-years on Opteron 2.2GHz in Dec 2009)

Symmetric vs. Asymmetric Algorithms

Symmetric algorithms: e.g. DES

+ Low complexity, higher efficiency (hardware implementation)
– Difficult key distribution
– More complex to realise digital signatures

Asymmetric algorithms: e.g. RSA

+ Simple key distribution
+ Digital signatures are easily realised
– High complexity, lower efficiency (even if implemented in hardware).

A combination of both methods is advisable

- Start the interaction with an asymmetric algorithm
- Switch to a symmetric algorithm

Secure Hash Functions

Applications:

- Ensure integrity
- Used for digital signatures

hash code **hash function** **message**

$$h = H(M)$$

fixed length **variable length**

Algorithm must guarantee:

It is practically impossible to find (for a given h) a message $M' \neq M$, i.e. the computation of H^{-1} is practically impossible

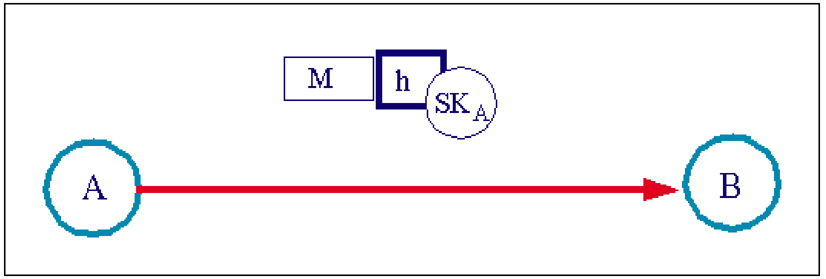
Digital Signatures

Required properties:

- Receiver can verify the authenticity of a message
- Sender cannot repudiate the message later

Ingredients:

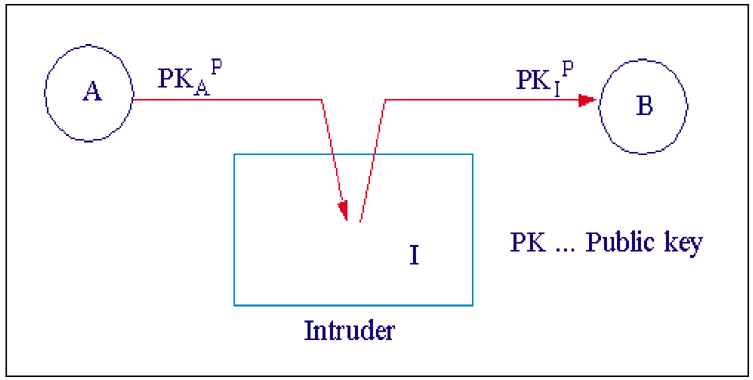
- (usually) asymmetric algorithm
- Secure hash function $h = \mathrm{H}(M)$



- h is encrypted with the private key (SK_A) of A
- Recipient can decrypt h using public key of A
- h can be used to check integrity
- Only the owner of the private key can have sent M → Authenticity

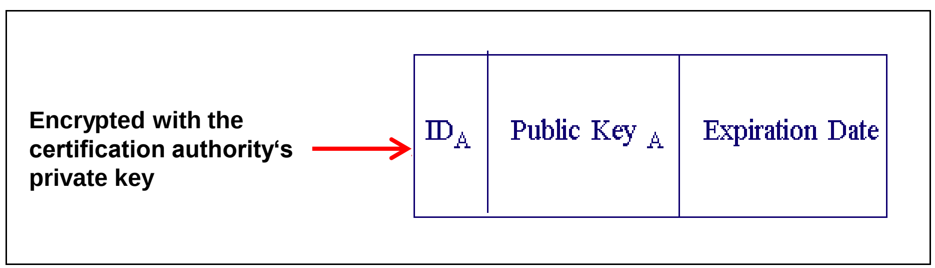
Certificates

Is key distribution with asymmetric algorithms really that simple?



Solution: Certificates:

- Mapping of a participant to its public key signed by a trusted entity

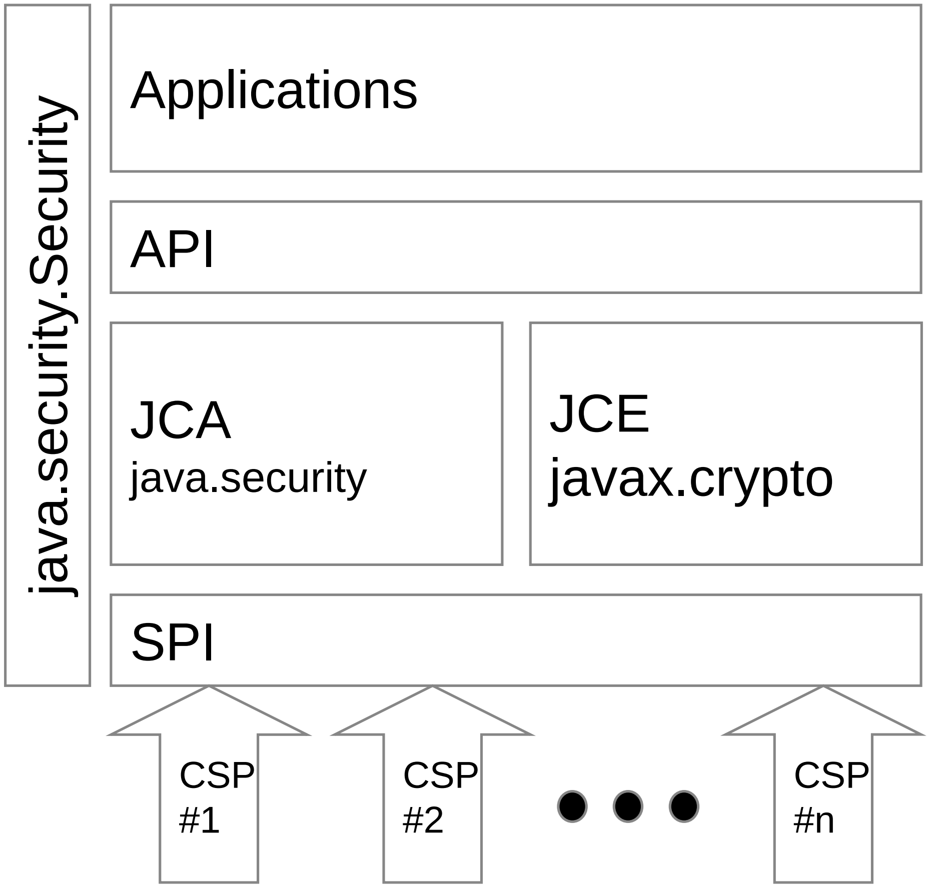


- The certification authority’s public key is well-known

Example: Java Cryptography Arcitecture

- Consists of two parts
  - Java Cryptography Architecture (JCA)
  - Java Cryptographic Extension (JCE)
  - Both parts included since Java 1.4
- JCA and JCE compose a framework
  - API for accessing cryptographic algorithms in applications
  - Service Provider Interface (SPI) for integrating a broad range of cryptographic implementations
- Cryptographic Service Providers (CSP)
  - Provide implementations of algorithms
  - Can be embedded into JCA/JCE
  - Sun/SunJCE provider included in Java SDK (implements DES, AES, MD5, etc.)

JCA/JCE Framework



- Multiple CSPs may be used at a time
- Class java.security.Security provides uniform access to JCA/JCE

Example: Encryption in Java (1)

```
KeyPair kp = null;
Cipher rsa = null;
String plainText = "This is the plain text.";
byte cipherBytes[], plainBytes[];

kpg = KeyPairGenerator.getInstance("RSA");
kpg.initialize(512);
kp = kpg.generateKeyPair();

rsa = Cipher.getInstance("RSA");
rsa.init(Cipher.ENCRYPT_MODE, kp.getPublic());
cipherBytes = rsa.doFinal(plainText.getBytes());

rsa.init(Cipher.DECRYPT_MODE, kp.getPrivate());
plainBytes = rsa.doFinal(cipherBytes);
```

- Declare variables
- Create a key-pair
- Encrypt plain text
- Decrypt cipher text

`KeyGenerator` for Asymmetric keys

Piecewise encryption:
`rsa.doUpdate(…);`
…
`rsa.doFinal(…);`

Example: Encryption in Java (2)

- Using cryptographic algorithms in Java is easy, but there is a lot more to do in a real application:
  - Management of certificates
  - Exchange of cryptographic keys
  - Secure storage of private keys
  - etc.

Conclusion

- Basic Techniques
  - Symmetric Cryptography
  - Asymmetric Cryptography
  - Secure Hash Functions
- Can Provide
  - Confidentiality
  - Integrity
  - Authenticity
  - Non-repudiation
  - Availability (to some extent)