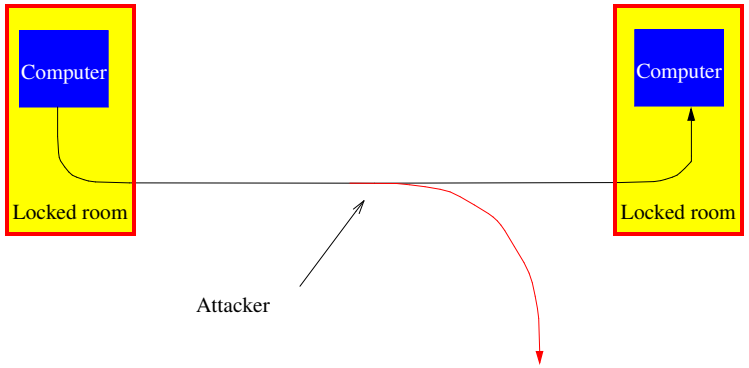
Security

- General Concepts
- Threats in Distributed Systems
- Access Control
- Authentication Protocols
- Kerberos
- Cryptography
  - with Private Keys
  - with Public Keys
- Signing Messages

General Concepts

- Security
  - confidentiality
    the capability to protect objects and information from non- authorized **use**.
  - integrity
    the capability to protect objects and information from non- authorized **changes**.

Threats in Distributed Systems



- threat
  - accidental threats
  - intentional threats
    - passive attack
    - active attack

Trojan Horses, Virus and Worms

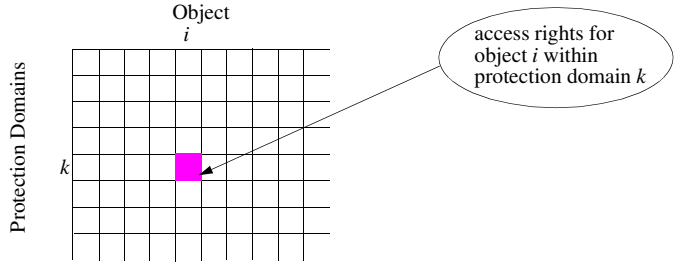
- exists in the system without permit from the system owner,
- can transfer from one computer to another,
- have potential capability to destroy or hurt files,
- have capability to hinder proper users from obtaining service.

Security Mechanisms

- prevention
- detection
- recovery

Access Control

- Protection Domains
  - Access matrix



  - Access lists
    - per object
  - Capability lists
    - per protection domain (“user”)

Capability

- A Capability will normally be composed of a three part bit pattern:
  - a unique identifier for the object or resource that is associated with the capability,
  - a specification of the access rights to the object, e.g. one boolean bit for each allowed operation type,
  - a random number making it hard to guess the content of the capability and also to make a fraud.

  This pattern will be encrypted so only the service giver can decrypt (read) it).

Identifying and Authentication Protocols

- Authentication - to verify an Identity.
- Main classes of Authentication in a distributed system:
  - Authentication of the contents of a message, i.e. verifying that the content is the same at receiving as at sending.
  - Authentication of the origins i of a message, i.e. verifying that the real sender of the message is the indicated sender.
  - Authentication of identities, i.e. verifying that an entity has the identity it claims.
- Authentication exists in the following combinations:
  - Host – Host.
  - User – Host.
  - Process – process.
- *Client-server* communication.
- *Peer-to-peer* communication.
- Authentication Protocol - to execute authentication in a distributed system

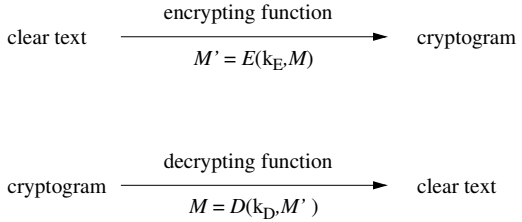
Authentication Protocol

Example:

Common login routine, e.g. like in UNIX:

U	→ H	U
H	→ U	"Password?"
U	→ H	*passwd*
H		compute λ = *f*(*passwd*).
		Coimpare λ with what is stored in a special file,
		the "password-file".
		if equal:**then** start login shell
H	→ U	"%"
		else: *abort login*

Cryptography



- symmetrical encrypting: $k_E = k_D$
- asymmetrical encrypting: $k_E \neq k_D$

Authentication Protocol using symmetrical encrypting (1)

- In the following we will use the following notation:
  - m: Clear text message
  - *m'*: Encrypted message
  - *E* (*k*,*m*) : Encryption function with key *k*.
  - *D* (*k*, *m'*) : Decryption function with key *k*.
- First try:

P		create m = "I am P"
		compute *m'* = *E* (*k*,*m*)
P	→ Q	*m*,*m'*
Q		verify that *D* (*k*, *m'*) = *m*
		if equal **then** accept
		else reject

This protocol is vulnerable to replaying.

Authentication Protocol using symmetrical encrypting (2)

- Second try:

P	→ Q	"I am P"
Q	→ P	*ts* //*ts* is a unique value read from the clock each time
P		compute *ts'* = *E* (*k*, *ts*)
P	→ Q	*ts'*
Q		verify that *D* (*k*, *ts'*) = *ts*
		if equal **then** accept
		else reject

This protocol is secure but can not be used due to the problem to administer the many keys.
There has to be one key for each pair of processes P and Q.

Three way Authentication Protocol

- Third try:

  A notifies the Authentication Server:

P	→ Q	"I am P"
Q	→ P	ts
P		compute $ts' = E(k_P, ts)$
P	→ Q	ts'
Q		compute $ts'' = E(k_Q, P, ts')$
Q	→ A	ts''
A		retreive (P,ts') by decrypting: $D(k_Q, ts'')$
		compute $ts''' = D(k_P, ts')$ $(= ts)$
		compute $m = E(k_Q, ts''')$
A	→ Q	m
Q		verify $D(k_Q, m) = ts$
		if equal **then** accept
		else reject

Authentication Protocol using asymmetrical encrypting

- We will use the following notation:
  - ke_P: Encryption key (public)
  - kd_P: Decryption key (secret)

P	→ Q	"I am P"
Q	→ P	ts
P		compute $ts' = E(ke_P, ts)$
P	→ Q	ts'
Q		verify that $ts = D(kd_P, ts')$
		if equal **then** accept
		else reject

Authentication Protocol using asymmetrical encrypting Three Way

P	→ Q	"I am P"
Q	→ P	ts
P		compute $ts' = E(ke_P, ts)$
P	→ Q	ts'
Q	→ A	"I need P's public key"
A		retreive kd_P from the database
		compute $c = E(ke_A, P, kd_P)$
A	→ Q	P,c
Q		retreive (P, kd_P) by decrypting: $D(kd_A, c)$
		verify $ts = D(kd_P, ts')$
		if equal **then** accept
		else reject

Kerberos

- MIT - Athena project
- ticket-granting server
- authentication server
- Two main protocols:
  - The credential-initialization protocol.
    Authentication of users login in to the system.
    Is performed by the Kerberos authentication server.
    Certificated user will get a ticket-granting ticket.
  - The client-server authentication protocol.
    Authentication of user processes that requests services over the network.
    The authentication is executed as a three part authentication between client, server, and Kerberos ticket-granting server.

The credential-initialization protocol

- We will use the following notation:
  - L: session key length of life
  - T: a timestamp

U	→ H	U
H	→ Auth	U, TGS
Auth		retreive k_U and k_{TGS} from the database generate a session key k create $tick_{TGS} = E(k_{TGS}, (U,TGS,k,T,L))$
Auth	→ H	$E(k_U, (TGS, k, T, L, tick_{TGS}))$
H	→ U	"Password?"
U	→ H	*passwd*
H		compute $\lambda = f(passwd)$ decrypt $D(k_U, (TGS,k,T,L,tick_{TGS}))$ with λ to obtain k, $tick_{TGS}$ **if** not succeed: abort login else keep $tick_{TGS}$ and k. erase $passwd_U$ from memory

The client-server authentication protocol

C	→ TGS	S, $tick_{TGS}$, $E(k, (C, T_1))$
TGS		retreive k from $tick_{TGS}$ by decrypting with k_{TGS} retreive T_1 from $E(k, (C, T_1))$ by decrypting with k. test T_1 to the local clock create $tick_S = E(k_S, (C, S, k', T', L'))$
TGS	→ C	$E(k, (S, k', T', L', tick_S))$
C		retreive k' and $tick_S$ by decrypting with k.
C	→ S	$tick_S$, $E(k', (C, T_2))$
S		retreive k' from $tick_S$ by decrypting with k_S. retreive T_2 from $E(k', (C, T_2))$ by decrypting with k'. test T_2 to the local clock.
S	→ C	$E(k', (T_2 + 1))$

Cryptography

- private keys
  symmetrical cryptographic algorithms

  Pros:
  - Fast encryption/decrypting
  - Cheap encryption/decrypting
  - Implemented in hardware

  Cons:
  - Must be frequently exchanged
  - Must be transported in a safe manner

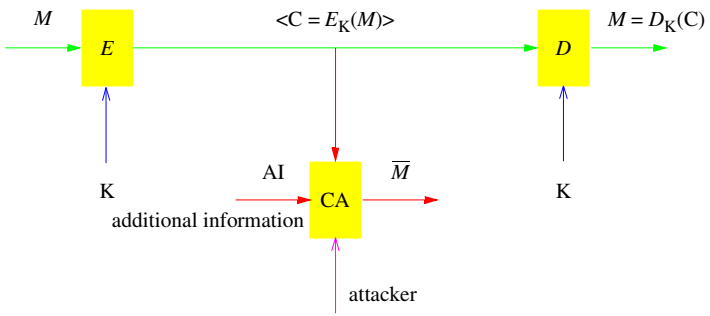
- public keys
  asymmetrical cryptographic algorithms

  Pros:
  - Can be openly transported
  - All can send encrypted to all others
  - Doesn't need frequently exchange

  Cons:
  - Slow encryption/decrypting
  - Expensive encryption/decrypting

- Strategy:
  - private key algorithm
    - for normal data
  - public key algorithm
    - for distributing private keys

Private keys



- *unicity distance*:
  - the number of bytes with the same key needed to retrieve it using the redundancy
- reducing redundancy:
  - block cipher
  - product cipher

Different types of cipher

```
We will attack at dawn
```

a) Message.

```
Xk xjmm buubdl bu ebxo
```

b) Caesar cipher (Shift cipher, K=1).

```
Ai ampp exxego ex hear
```

c) Shift cipher, K=4.

K:
```
abcdefghijklmnopqrstuvwxyz
qazwsxedcrfvtgbyhnujmiklpo
```

```
Ks kcvv qjjqzf qj wqkg
```

d) Permutation cipher, key K.

- Permutation cipher can be cracked using the frequencies of the letters in the English language. The more text with the same key the easier it will be done.

Block cipher

```
We will attack at dawn
```

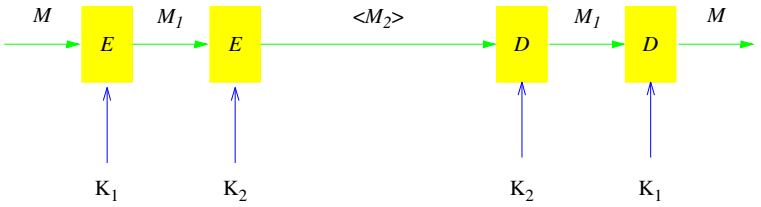
a) Message.

```
We w  ill  atta  ck a  t da  wn
```

e) Block cipher.

- To crack Block Cipher the frequencies of letter combinations must be used which will require more text

Product cipher



- To make it harder to crack we encrypt several times in a row.
- **Beware**: This is no guarantee that it will be harder to crack, e.g. two permutation ciphers after each other will just give one new permutation cipher.

Different Attacks

- Attack:
  - only encrypted text
  - known encrypted text
  - chosen encrypted text
- secure system
  - once use key
    - expensive
    - key as long as the encrypted text
      - hard to create
      - hard to transport

DES

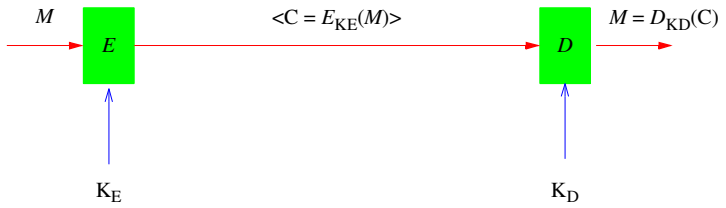
Data Encryption Standard

- IBM
- DARPA
- standard
- hardware
- cheap and fast
- block length: 64 bits
- key length 56 bits + 8 control bits

Triple DES (3DES)

- A more secure variant of DES
  - Encryption, decrypting and Encryption again of the blocks according to DES. Should be done using different keys.
  - estimated to be 2^{56} times safer then DES.
  - will probably be used for long time
- As Advanced Encryption Standard for safe encryption the US government has chosen the Rijndael algorithm (Joan Daemon, Vincent Rijmen).
  - A block cipher with variable block and key length.

Public Key



K_E known K_D secret

- hard to compute K_D when K_E is known
- trap door function

RSA

Rivest-Shamir-Adleman 1978

- choose two big primes p and q
- let $n = p * q$
- choose a random number e.
- let $K_E = (e,n)$ be the public key

- Encryption:
  - Dive the message into blocks not bigger than n-1 bytes
  - $M = P_1 P_2 ... P_m$
  - Compute $C_i = P_i^e$ modulo n.
  - $C = C_1 C_2 ... C_m$ will be the encrypted message.

RSA

Decrypting

- Uses *Euler's totient function*:

  $\phi(n) = (p - 1) * (q - 1)$

  for which it holds:

  $X^Y \bmod n = X^{Y \bmod \phi(n)}$

- d is computed by solving the equation:

  $e*d \bmod \phi(n) = 1$

- For each C_i we compute:

  $C_i^d \bmod n = P_i^{e*d} \bmod n = P_i^{e*d \bmod \phi(n)} = P_i^1 = P_i$

- The message is decrypted!

RSA - Example

- Assume that we have chosen:

  $p = 73, \quad q = 151 \quad$ and $\quad e = 11$

- This implies $n = p*q = 11\ 023$.
- The public key will be (e,n).
- The secret key (d,n) is computed by solving:

  $11*d \bmod ((73 - 1)*(151 - 1)) = 1$

- giving

  $d*11 \bmod 10\ 800 = 1 \quad \Rightarrow d = 5\ 891$

- We will now show how the message: "*How are you*?" is sent in a secure way.
  - The letters will be coded as numbers:

    $H \rightarrow 33$, $o \rightarrow 14$, $w \rightarrow 22$ ... $? \rightarrow 66$

- The message will be:

  33 14 22 62 00 17 04 62 24 14 20 66.

- We will use blocks with 4 digits. The message will be:

  $P_1 = 3\ 314,\ P_2 = 2\ 262,\ P_3 = 0\ 017,\ P_4 = 0\ 462,\ P_5 = 2\ 414,\ P_6 = 2\ 066.$

- The sender will encrypt each block as:

  $C_1 = 3\ 314^{11} \bmod 11\ 023 = 10\ 260.$

  $C_2 = 9\ 489,\ C_3 = 1\ 782,\ C_4 = 727,\ C_5 = 10\ 032,\ C_6 = 2\ 253.$

- The encrypted message $C = (C_1 .. C_6)$ that is sent will be:

  10 260, 9 489, 1 782, 727, 10 032, 2 253.

- The receiver will decrypt using the secret key (d):

  $10\ 260^d \bmod n = 10\ 260^{5\ 891} \bmod 11\ 023 = 3\ 314 = P_1$

Signing messages

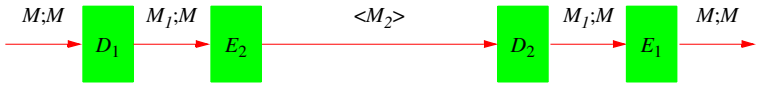
- RSA algorithm is symmetric

  I:

  - encrypted using K_E
  - decrypt using K_D

  II:

  - encrypted using K_D
  - decrypt using K_E



$M_1 = D_1(M)$ $M_2 = E_2(M_1;M)$