Bachelor of Information System

IS2109 - Information System Security

Additional Lecture - 3

Kasun De Zoysa
kasun@ucsc.cmb.ac.lk

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

UCSC

Symmetric Key Cryptography

- Traditional **secret/single key** cryptography uses **one** key
- Shared by both sender and receiver
- If this key is disclosed, communications are compromised
- Also is **symmetric**, parties are equal
- Hence receiver can forge a message and claim it was sent by sender

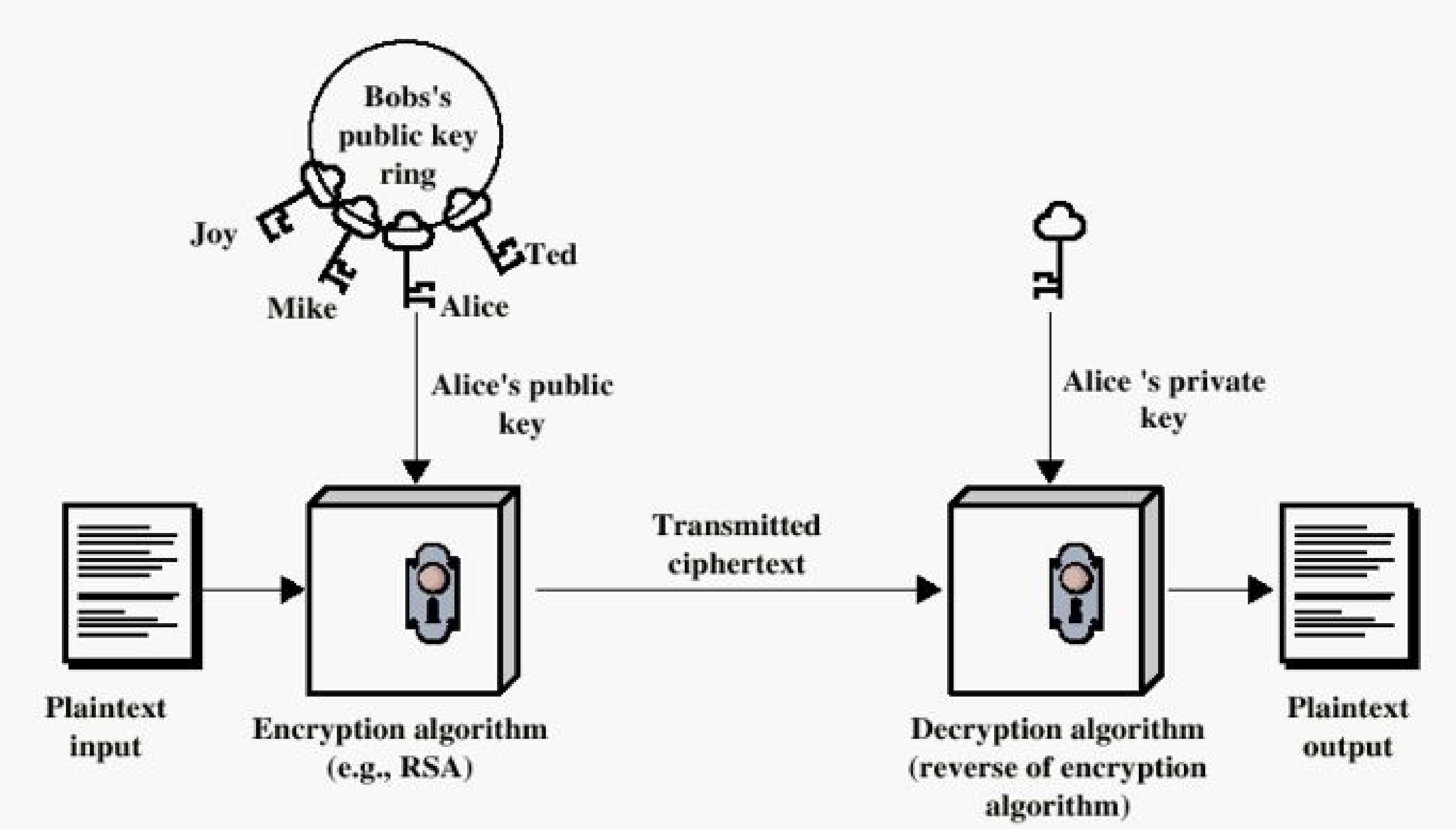
Why Public-Key Cryptography?

- Developed to address two issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
- **digital signatures** – how to verify a message comes intact from the claimed sender
- Whitfield Diffie and Martin Hellman in 1976 known earlier in classified community

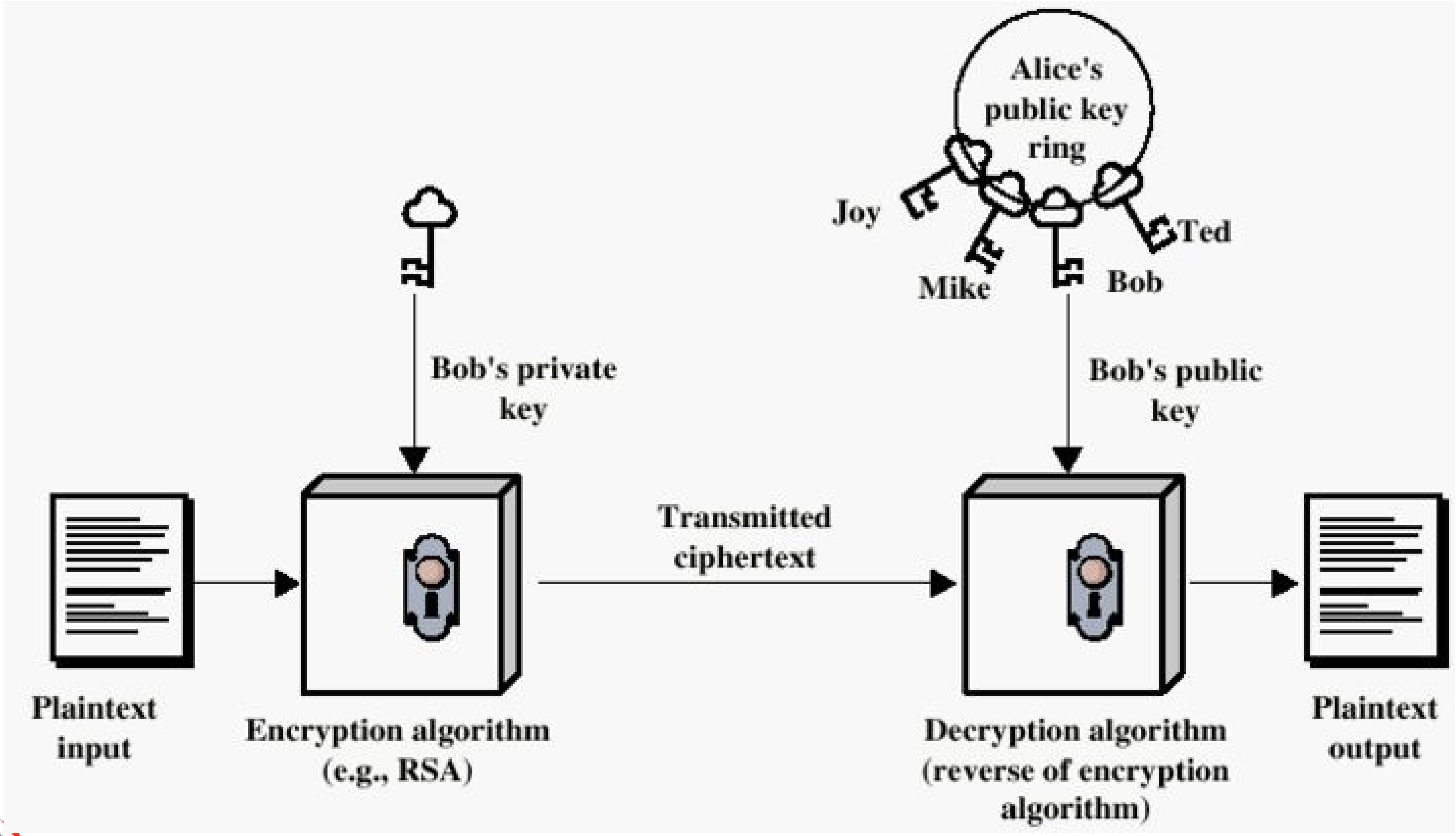
Public-Key Cryptography Principles

- The use of two keys has consequences in: key distribution, confidentiality and authentication.
- The scheme has six ingredients
  - Plaintext
  - Encryption algorithm
  - Public and private key
  - Ciphertext
  - Decryption algorithm

Encryption using Public-Key system



Authentication using Public-Key System



Applications for Public-Key Cryptosystems

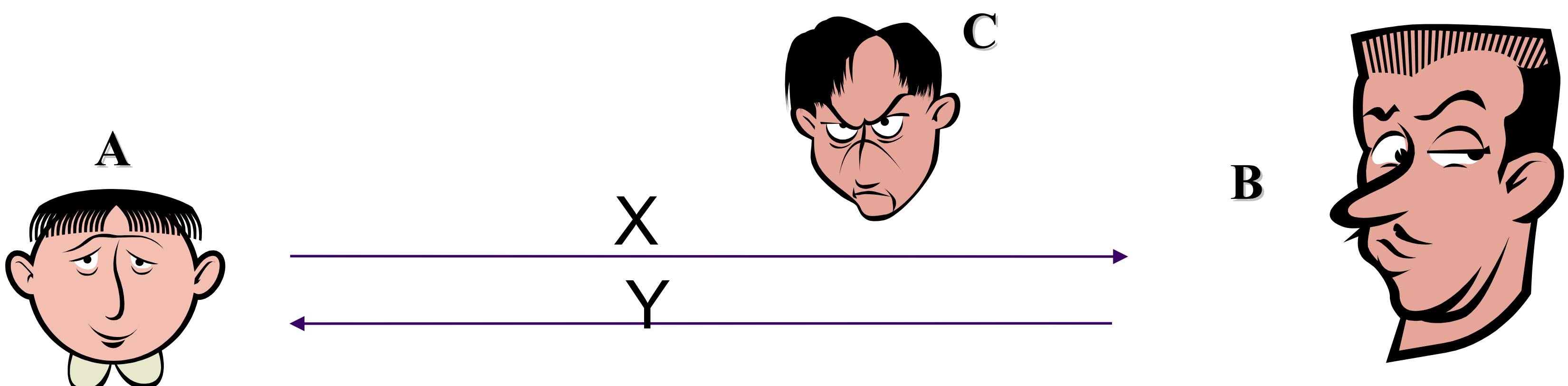
- Three categories:
  - **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
  - **Digital signature:** The sender ”signs” a message with its private key.
  - **Key exchange:** Two sides cooperate two exchange a session key.

Public-Key Cryptographic Algorithms

- **RSA** - Ron Rives, Adi Shamir and Len Adleman at MIT, in 1977.
  - RSA is a block cipher
  - The most widely implemented
- **Diffie-Hellman**
  - Exchange a secret key securely
  - Compute discrete logarithms
- **Elliptic Curve Cryptography (ECC)**

Diffie-Hellman Key Agreement

- **Published in 1976**
- **Based on difficulty of calculating discrete logarithm in a finite field**
- **Two parties agreed on two large numbers n and g, such that g is a prime with respect to n**



$X = g^x \bmod n$

$k = Y^x \bmod n = g^{xy} \bmod n$

$Y = g^y \bmod n$

$k = X^y \bmod n = g^{xy} \bmod n$

Possible to do man in the middle attack

Revest-Shamir-Adelman (RSA)

By Rivest, Shamir and Adelman in 1978

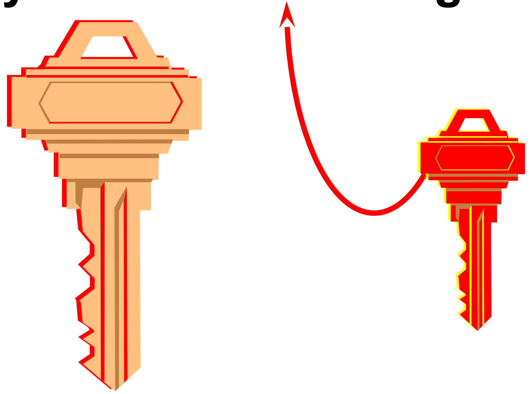
1. Find 2 large prime numbers p and q (100 digits=512bits)

2. Calculate the product n=p*q (n is around 200 digits)

3. Select large integer e relatively prime to (p-1)(q-1)
Relatively prime means e has no factors in common with (p-1)(q-1).
Easy way is select another prime that is larger than both(p-1) and (q-1).

4. Select d such that e*d mod (p-1)*(q-1)=1

Two keys are d and e along with n



Encryption
$C=P^e$ mod n

Decryption
$P=C^d$ mod n

RSA - Simple Example

1. Find 2 prime numbers p and q

Let p=11 and q=13

2. Calculate the product n=p*q

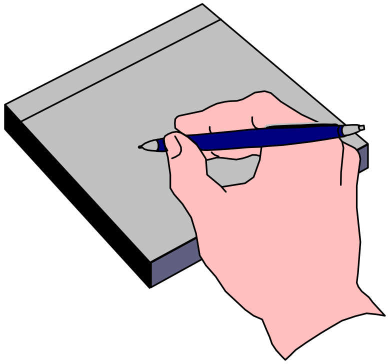
*n = 11*13=143*

3. Select large integer e relatively prime to (p-1)(q-1)

*E=11 ; 11 IS Relatively prime to (p-1)(q-1) =10*12=120*

4. Select d such that e*d mod (p-1)*(q-1)=1

*d=11 because, 11*11 mod 10*12=121 mod 120 =1*



Encryption

$C=P^e$ mod n

Let p=7 so that $C=7^{11}$ mod 143; C=106

Decryption

$P=C^d$ mod n

$p=106^{11}$ mod 143; P=7

RSA --- 1st small example (1)

- Kamal:
  - chooses 2 primes: $p=5, q=11$
    multiplies p and q: $n = p*q = 55$
  - finds out two numbers $e=3$ & $d=27$ which satisfy
    $(3 * 27) \bmod 40 = 1$
  - Kamal's public key
    - 2 numbers: ($3, 55$)
    - encryption alg: modular exponentiation
  - secret key: ($27,55$)

RSA --- 1st small example (2)

- Amal has a message $m=13$ to be sent to Kamal:
  - finds out Kamal's public encryption key $(3, 55)$
  - calculates c:

$$
\begin{aligned}
c &= m^e \pmod n) \\
&= 13^3 \pmod{55} \\
&= 2197 \pmod{55} \\
&= 52
\end{aligned}
$$

  - sends the ciphertext $c=52$ to Kamal

RSA --- 1st small example (2)

- Amal has a message $m=13$ to be sent to Kamal:
  - finds out Kamal's public encryption key (3, 55)
  - calculates c:

$$
\begin{aligned}
c &= m^e \ (mod\ n) \\
&= 13^3 \ (mod\ 55) \\
&= 2197 \ (mod\ 55) \\
&= 52
\end{aligned}
$$

  - sends the ciphertext $c=52$ to Kamal

RSA --- 1st small example (3)

- Kamal:
  - receives the ciphertext $c=52$ from Amal
  - uses his matching secret decryption key 27 to calculate m:

$$m = 52^{27} \pmod{55}$$
$$= 13 \text{ (Amal's message)}$$

RSA Signature --- an eg (1)

- Kamal:
  - chooses 2 primes: *p=5, q=11*
    multiplies p and q: $n = p*q = 55$
  - finds out two numbers *e=3* & *d=27* which satisfy
    $(3 * 27) \bmod 40 = 1$
  - Kamal's public key
    - 2 numbers: (*3, 55*)
    - encryption algo: modular exponentiation
  - secret key: (*27,55)*

RSA Signature --- an eg (2)

- Kamal has a document *m=19* to sign:
  - uses his secret key *d=27* to calculate the digital signature of *m=19*:

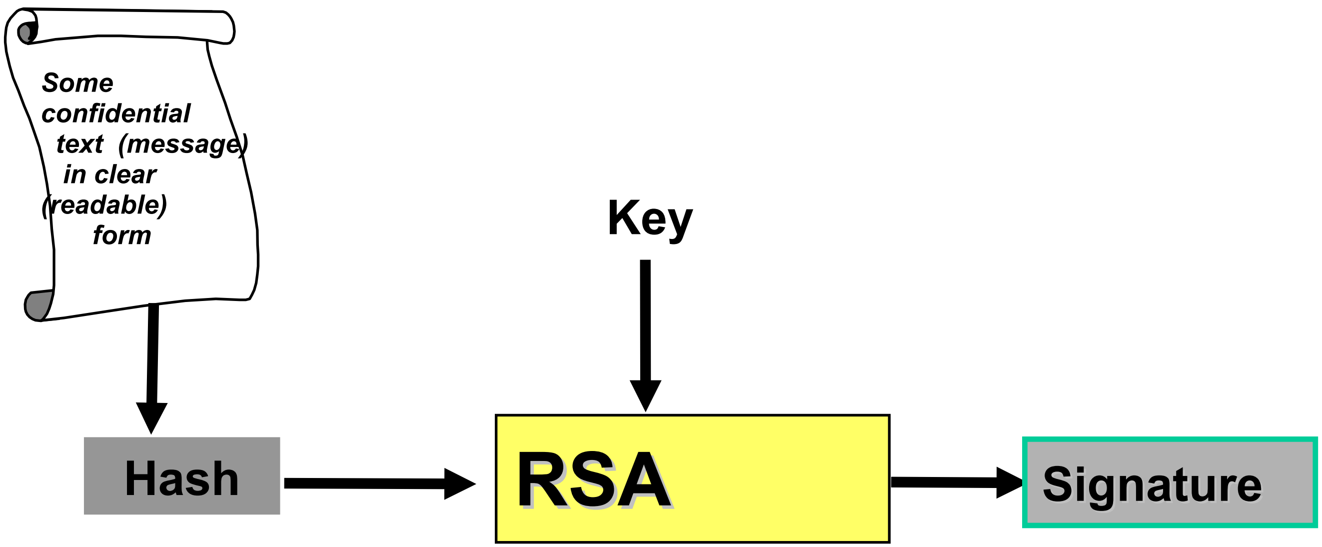
$$
\begin{aligned}
s &= m^d \pmod{n} \\
&= 19^{27} \pmod{55} \\
&= 24
\end{aligned}
$$

  - appends 24 to 19. Now *(m, s) = (19, 24)* indicates that the doc is 19, and Kamal's signature on the doc is 24.

RSA Signature --- an eg. (3)

- Nimal, a verifier:
  - receives a pair $(m,s)=(19, 24)$
  - looks up the phone book and finds out Kamal's public key $(e, n)=(3, 55)$
  - calculates $t = s^e \pmod{n}$
    $= 24^3 \pmod{55}$
    $= 19$
  - checks whether $t=m$
  - confirms that ($19,24$) is a genuinely signed document of Kamal if $t=m$.

Typical Digital Signature



Signature Creation

- **Generate Public/Private key pair**

```
openssl genrsa -out mykey.pem
openssl rsa -in mykey.pem -pubout >mypub.pem
```

- **Create the signature**

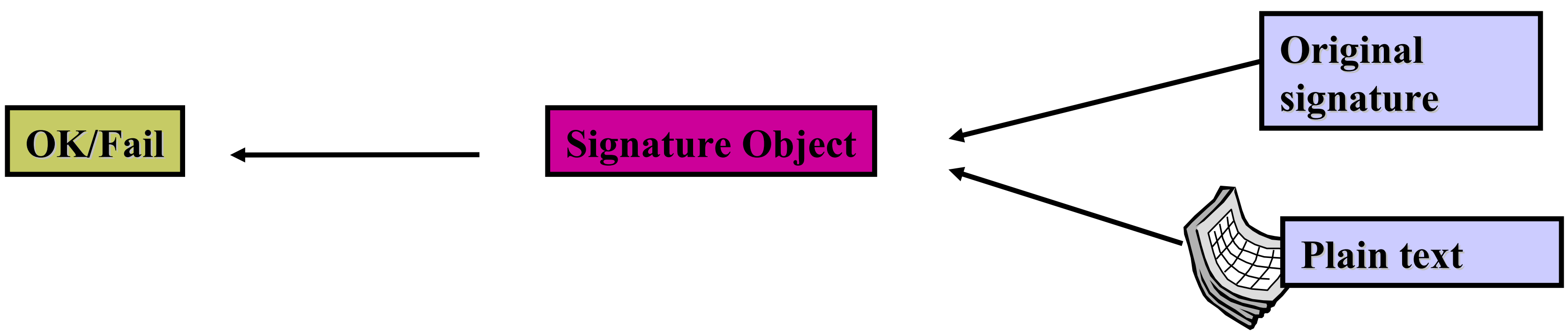
```
openssl dgst -sha1 -sign mykey.pem
-out mysign.sha1 jethavanaya.jpg
```

Signature Verification

- Retrieves the Public key

- Verify the signature

```
openssl dgst -sha1 -verify mypub.pem
-signature mysign.sha1 jethavanaya.jpg
```



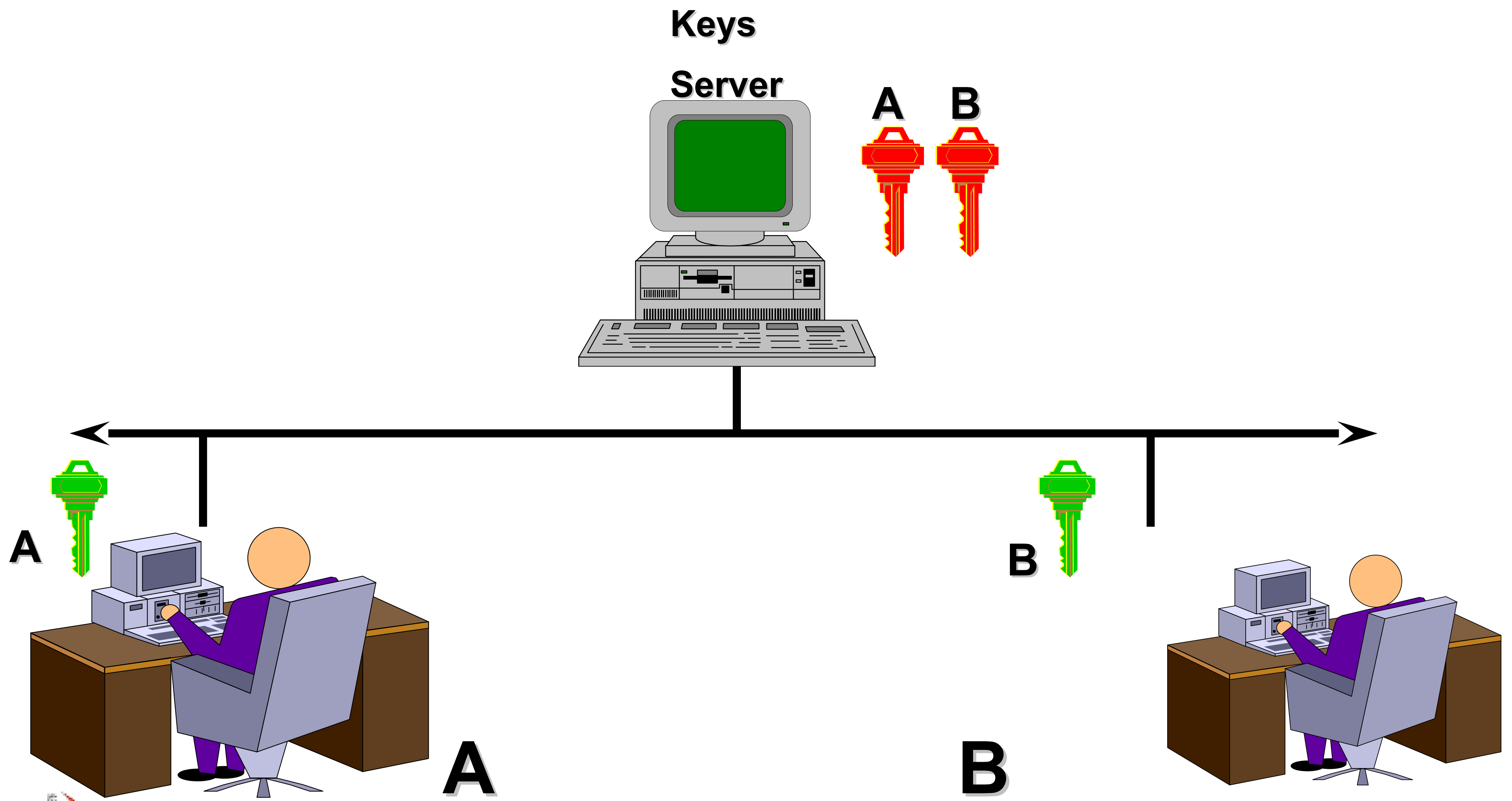
Hybrid Encryption

- Why is symmetric key encryption still used?
  - Performance
  - Also cryptographic reasons

In practice one uses hybrid encryption...

  - A one-time random key is generated ("session key")
  - This is used to symmetrically encrypt the message
  - The symmetric session key is encrypted through public key encryption and sent to the other party together with the (encrypted) message

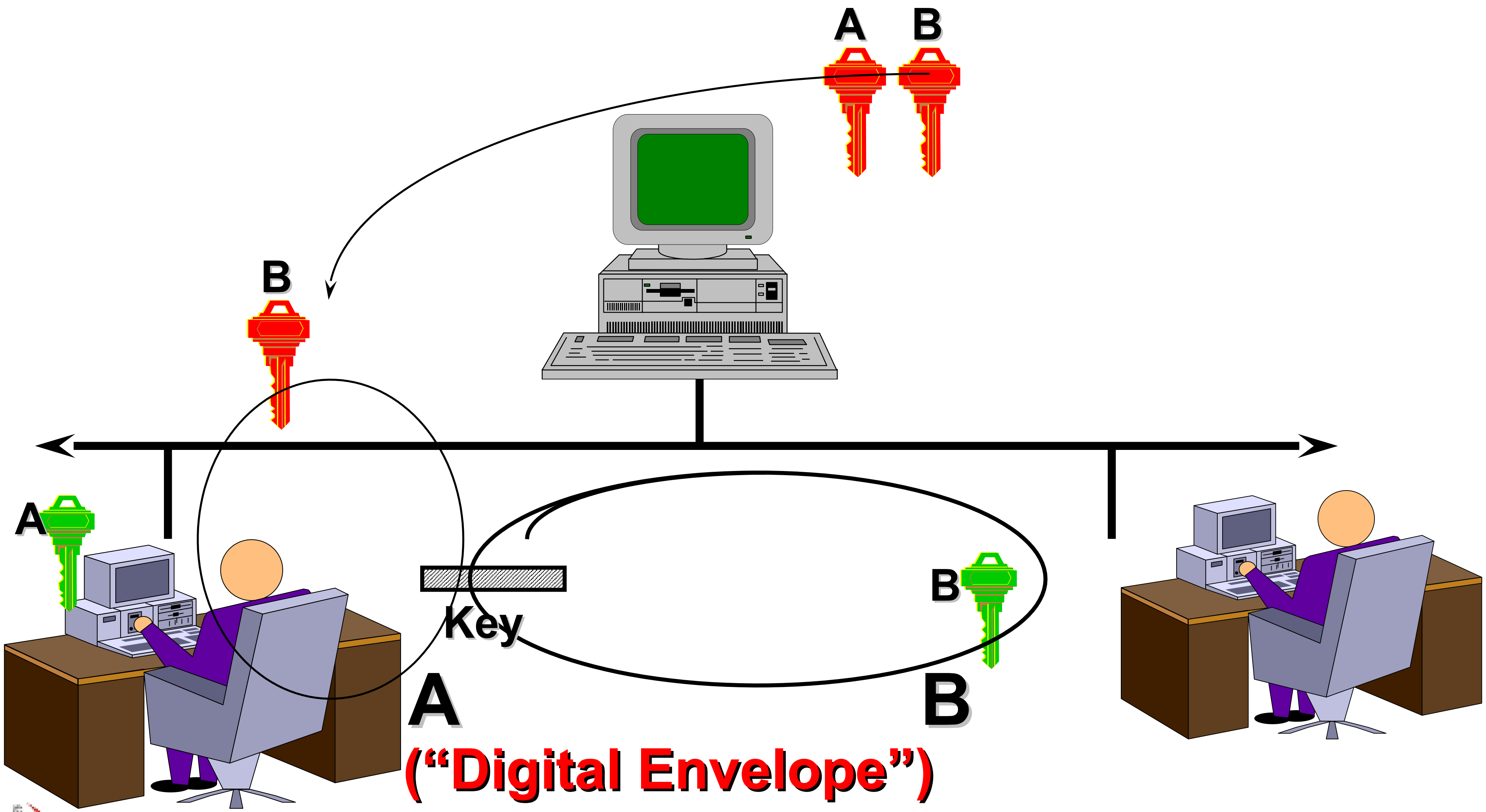
Storage and Handling Public Keys



Key Management

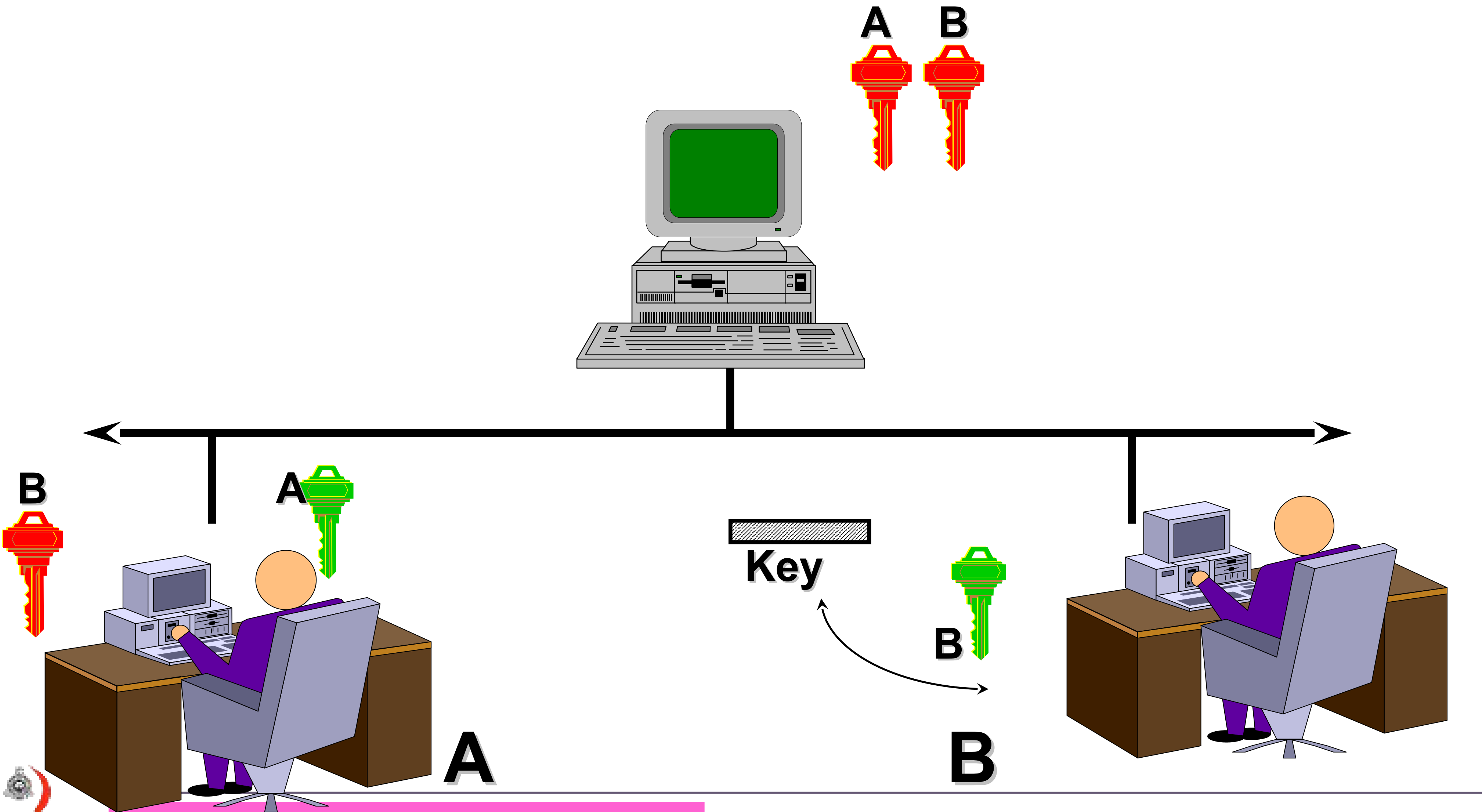
- Using a public key system, A wants to talk to B
- C is the Key Distribution Center(Key Server), has A and B's public key
- A calls B, and the calling protocol contacts C
- C encrypts a session key, "k", with the public keys and sends the encrypted "k" to A and B
- A and B can then communicate

Secure Sending of secret key

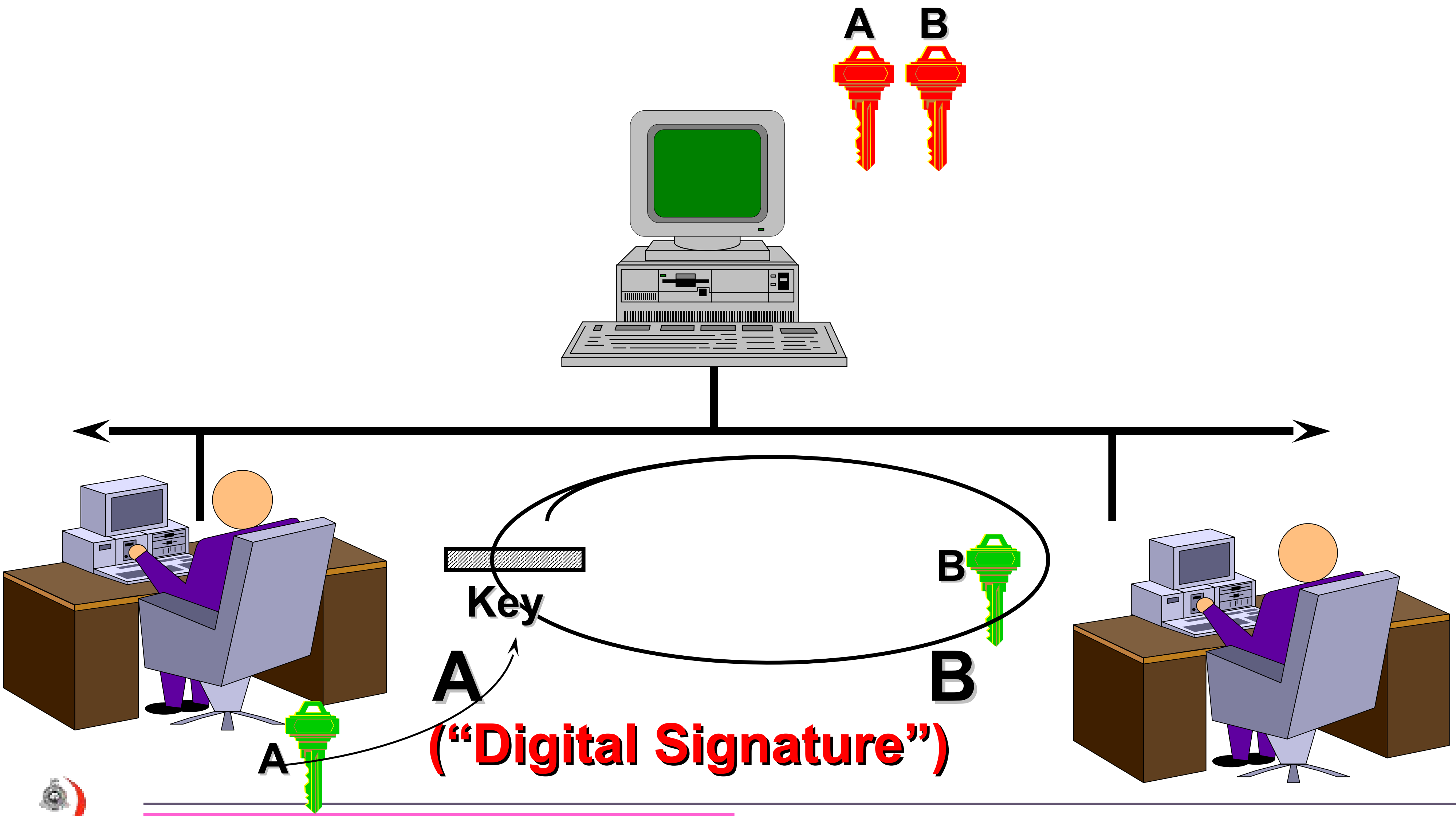


("Digital Envelope")

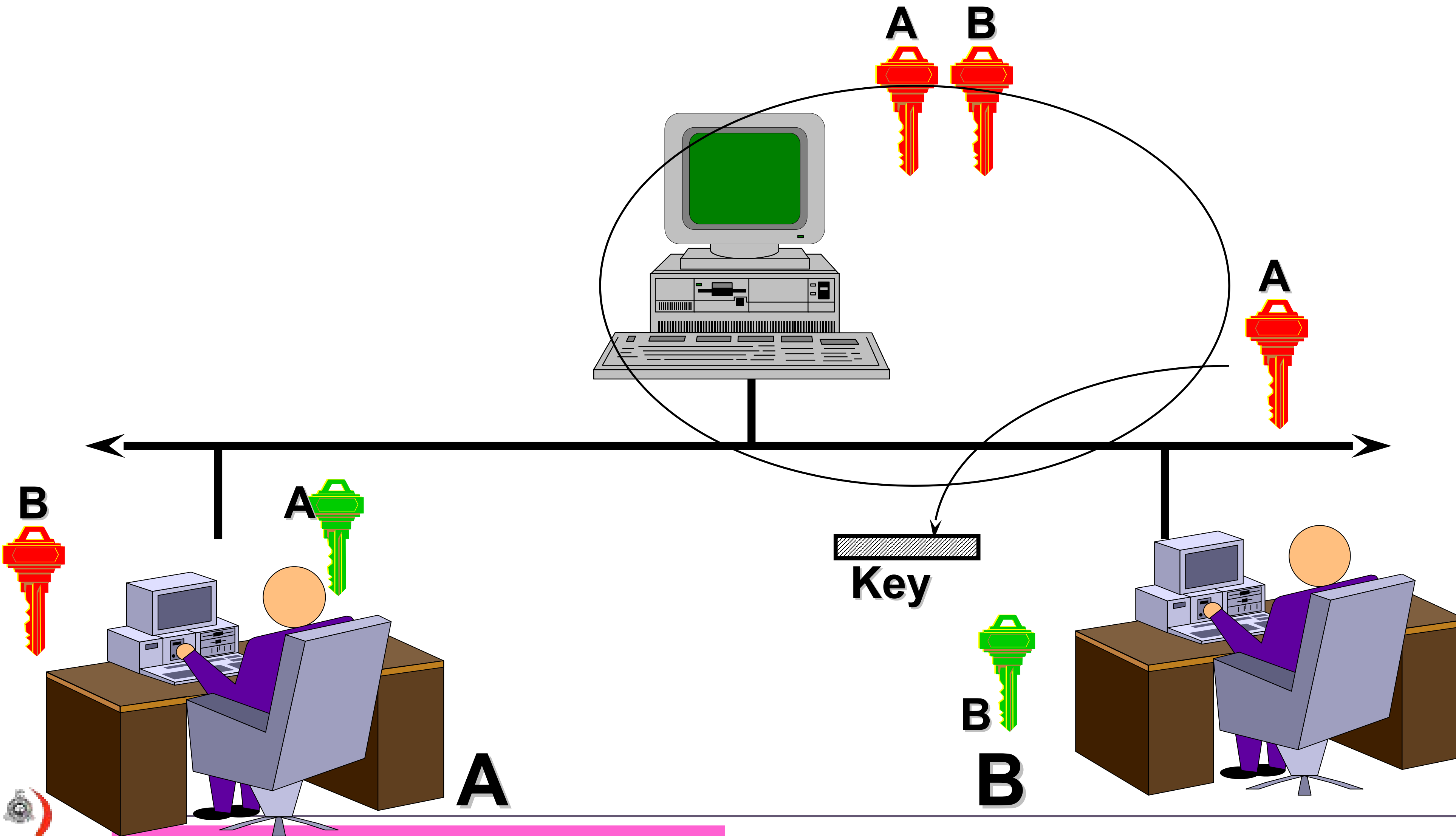
Recovery of Secret Key



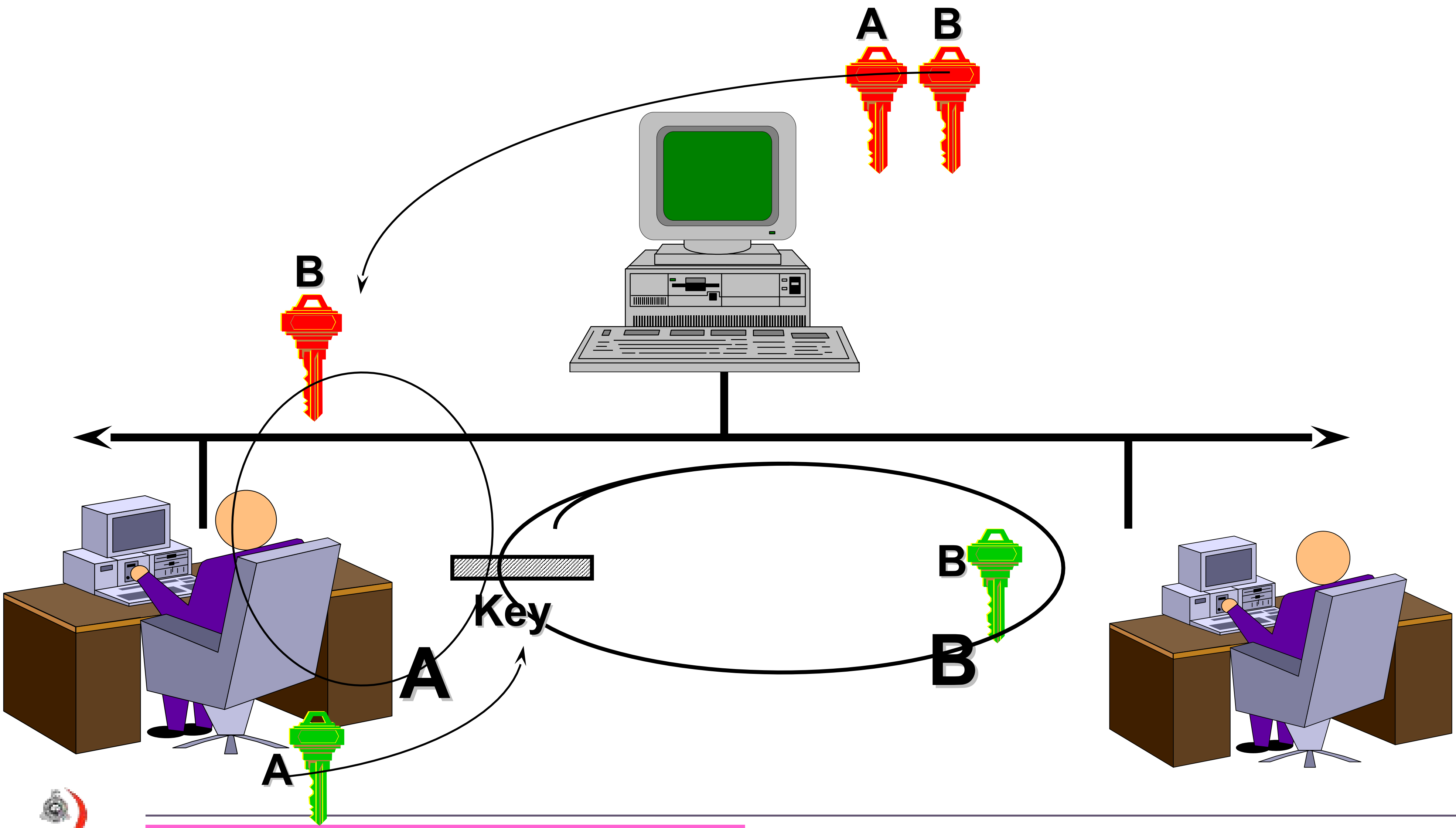
Authenticity of Sender



Verification of Signature

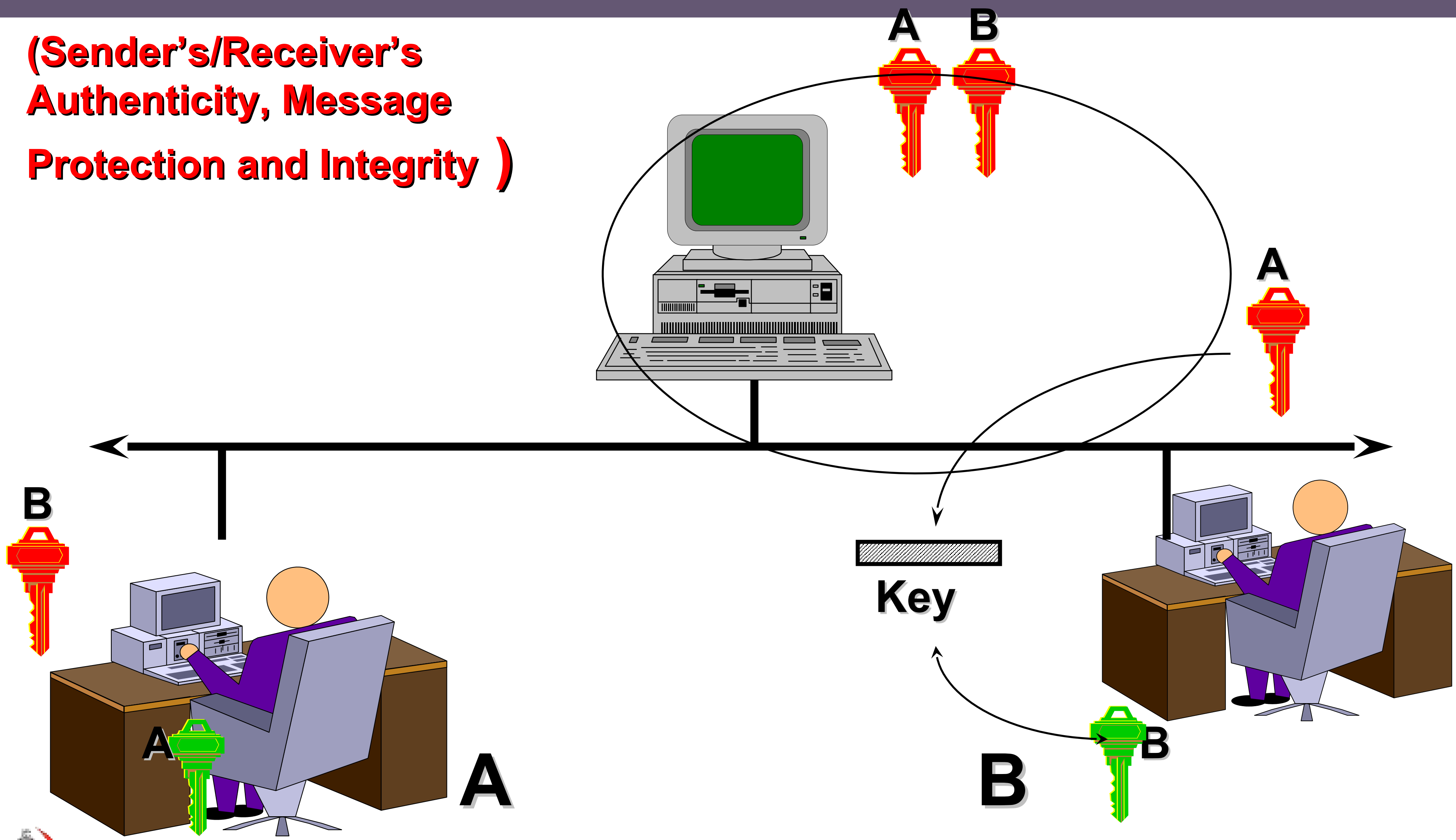


Authenticity of Sender and Receiver

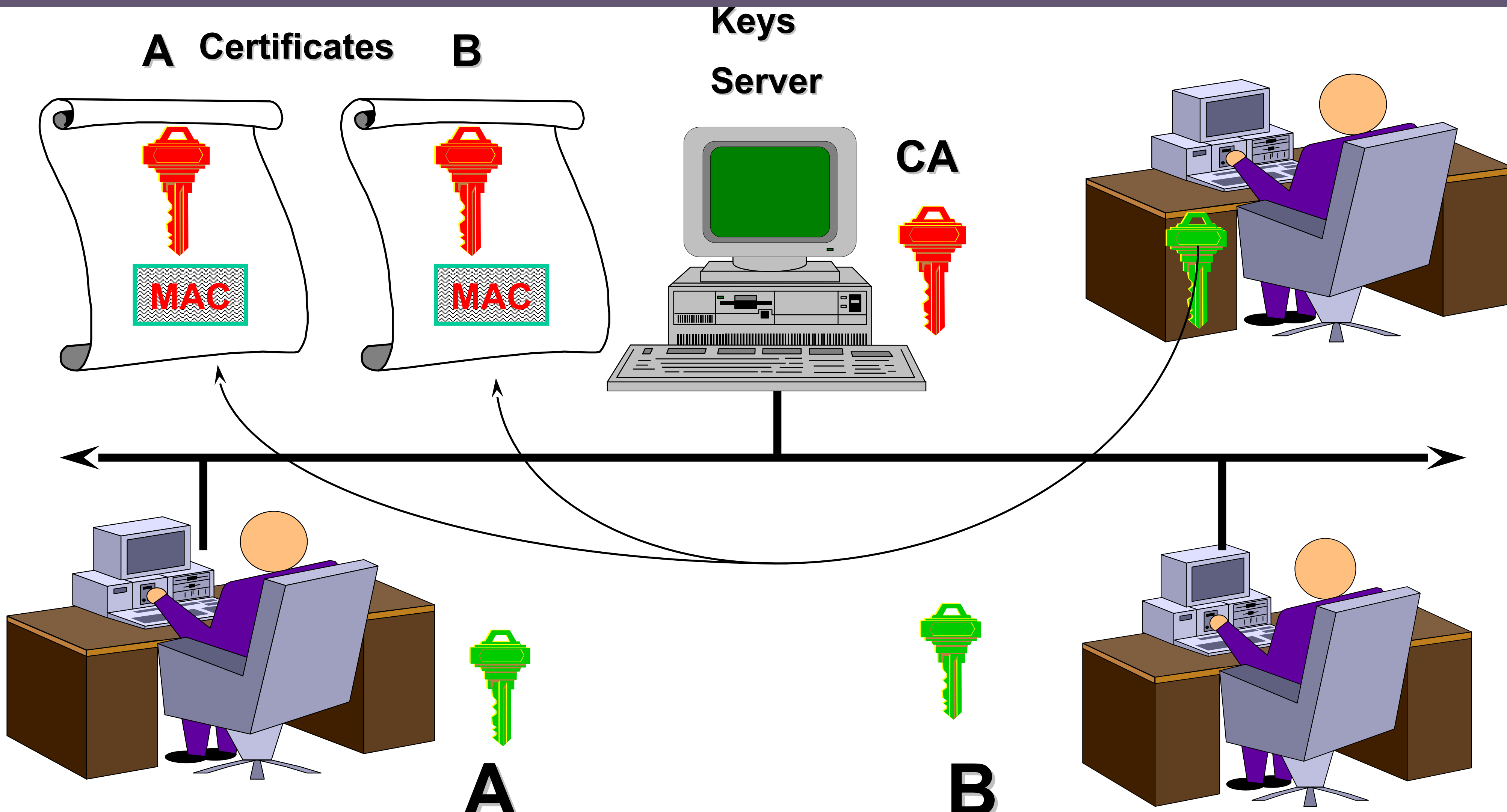


Full Verification

(Sender's/Receiver's Authenticity, Message Protection and Integrity)



Certificate Authority



Certificates Infrastructure

- Certificates need some infrastructure in place to allow users to verify a given certificate.
- This can be done centrally or via a distributed system.
- So how are certificates, and their certificate chains, verified and disseminated?

(1) Trusted Third Party (TTP)

(2) Certificate Authority (CA)

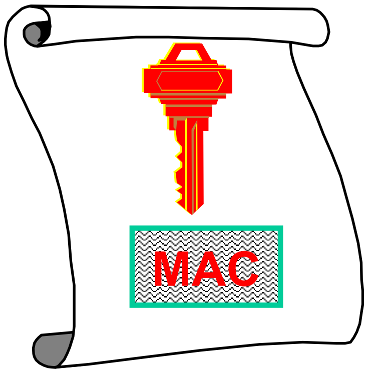
(3) Simple Public Key Infrastructure (SPKI)

Certificates Infrastructure

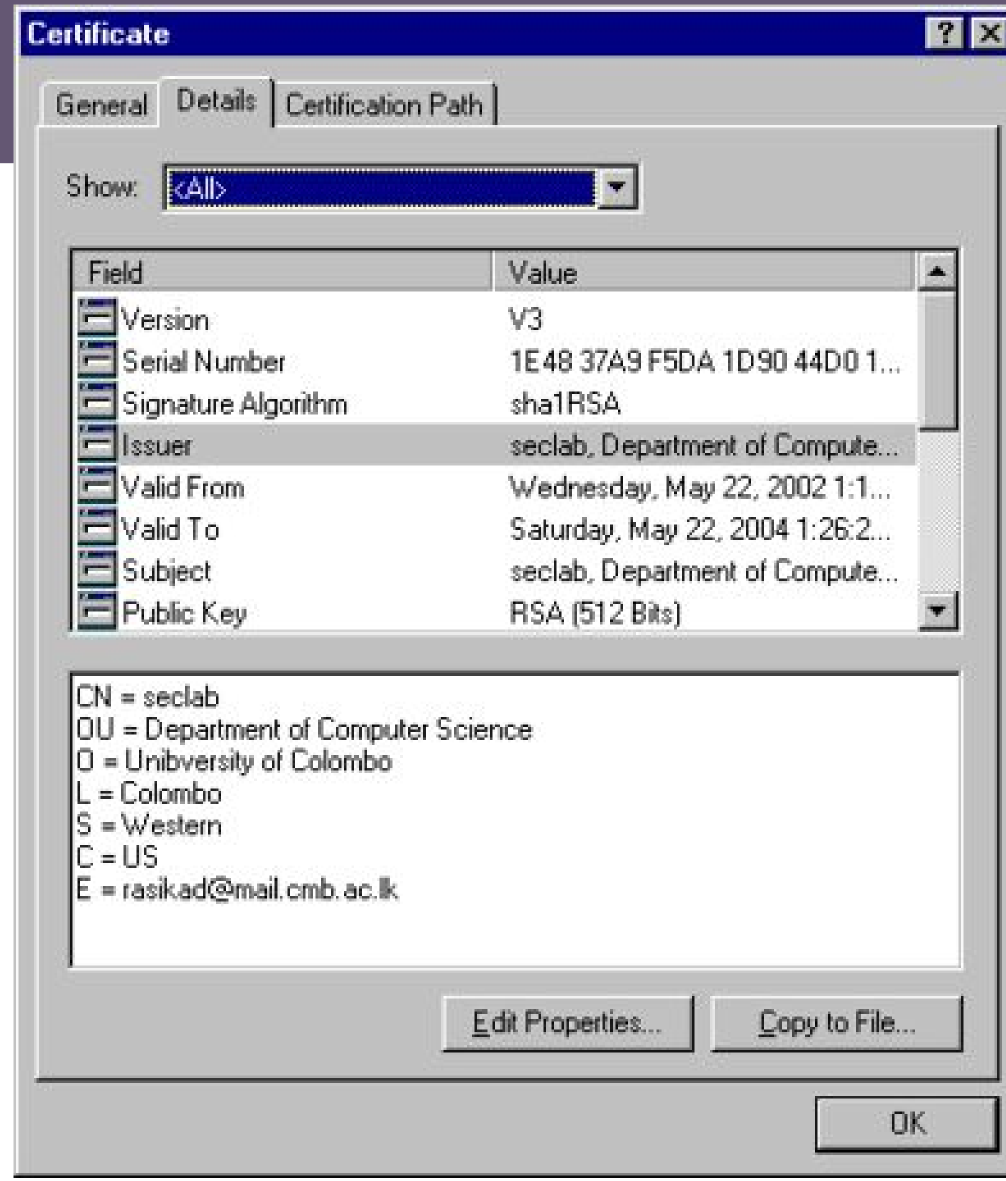
- Trusted, 3rd party organization
- CA (Certificate Authority) guarantees that the individual granted a certificate is who he/she claims to be
- CA usually has arrangement with financial institution to confirm identity
- Critical to data security and electronic commerce
- Well known organisation establish themselves to act as certificate authorities. Verisign, CREN, etc.
- One can then obtain X.509 public key certificates from them by submitting satisfactory evidence of their identity.

Internal Structure of Certificate

- Version
- Serial Number
- Signature Algorithm
- Issuer
- Subject
- Validity
- Subject Public Key Information
- Extensions
- Signature



Root Certificate



Public key infrastructure (PKI)

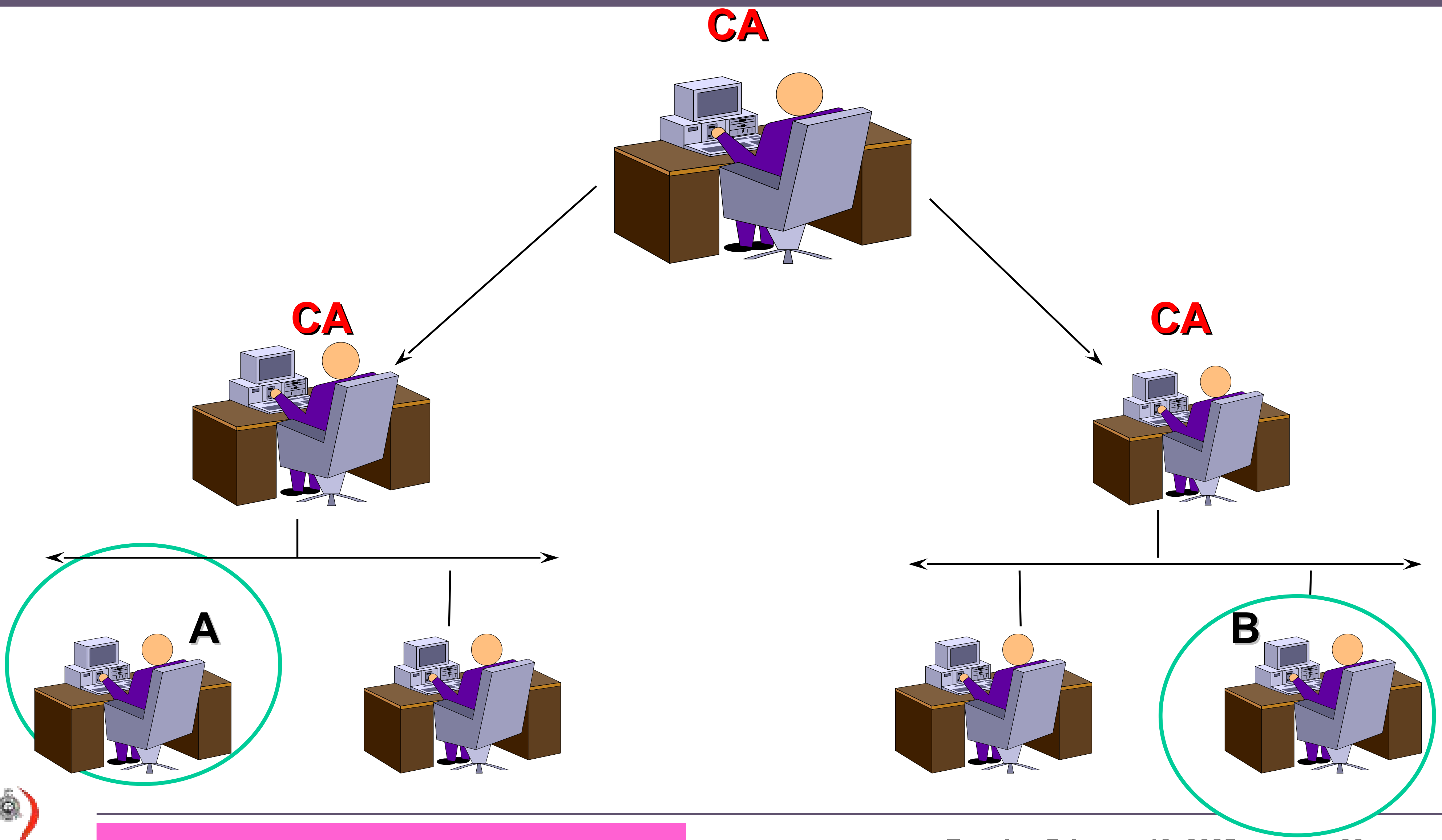
- Public key infrastructure (PKI) - provides the foundation necessary for secure e-business through the use of cryptographic keys and certificates
  - Enables secure electronic transactions
  - Enables the exchange of sensitive information

PKI

Public Key Infrastructure (PKI)

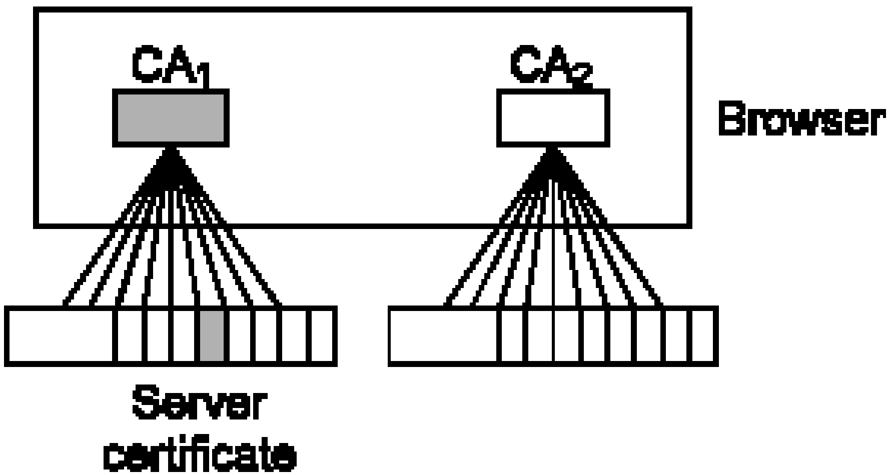
- PKI is an ISO authentication “framework” that uses public key cryptography and X.509 standard protocols.
- The framework establishes a generalized architecture for exchanging secure communication across networks. (Internet, internal / external).
- PKI is a Hybrid Key System with an infrastructure allowing the PKI certificate authority to create, maintain and manage digital certificates.
- Each user is issued a Digital Certificate (DC) which contains the end users public key along with other identifying information.
- The Digital Certificate is signed / validated by a trusted third party / Certificate Authority (CA). The CA enables users who are not trusted to each other (unknown) to trust each other.
- The most popular DC is an X.509 v3 This is same type of certificate as an SSL / HTTPS Certificate.

Certificate Hierarchy



CA Hierarchy in Practice

Flat or Clayton's hierarchy

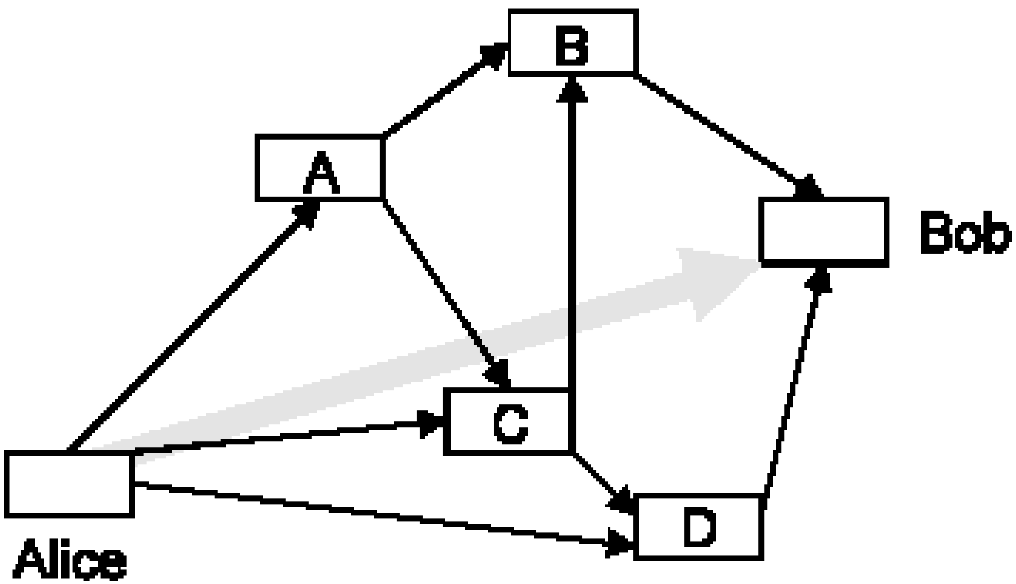


CA certificates are hard-coded into web browsers or email software

- Later software added the ability to add new CAs to the hardcoded initial set

Alternative Trust Hierarchies

PGP web of trust



Bob knows B and D who know A and C who know Alice

⇒ Bob knows the key came from Alice

Web of trust more closely reflects real-life trust models

Certificate Revocation

•Revocation is managed with a Certificate Revocation List (CRL), a form of anti-certificate which cancels a certificate

• Equivalent to 1970s-era credit card blacklist booklets

• Relying parties are expected to check CRLs before using a certificate

– *“This certificate is valid unless you hear somewhere that it isn’t”*

CRL Distribution Problems

• CRLs have a fixed validity period

– Valid from *issue date* to *expiry date*

• At *expiry date*, all relying parties connect to the CA to fetch the new CRL

– Massive peak loads when a CRL expires (DDOS attack)

• Issuing CRLs to provide timely revocation exacerbates the problem

– 10M clients download a 1MB CRL issued once a minute = ~150GB/s traffic

– Even per-minute CRLs aren't timely enough for high-value transactions with interest calculated by the minute

Online Status Checking

•Online Certificate Status Protocol, **OCSP**

• Inquires of the issuing CA whether a given certificate is still valid

– Acts as a simple responder for querying CRL's

– Still requires the use of a CRL to check validity

• OCSP acts as a selective CRL protocol

– Standard CRL process: "Send me a CRL for everything you've got"

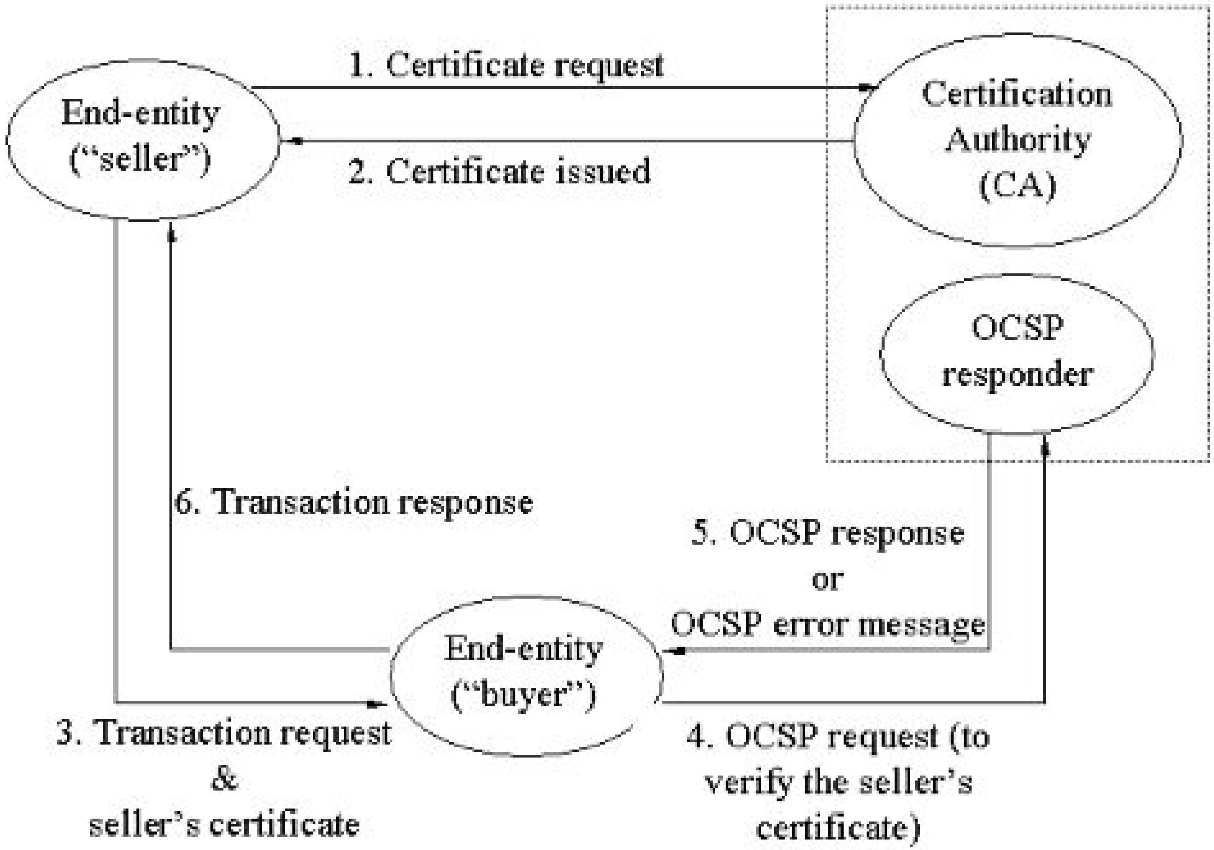
– OCSP process: "Send me a pseudo-CRL/OCSP response for only these certs"

– Lightweight pseudo-CRL avoids CRL size problems

– Reply is created on the spot in response to the request

– Ephemeral pseudo-CRL avoids CRL validity period problems

Online Status Checking



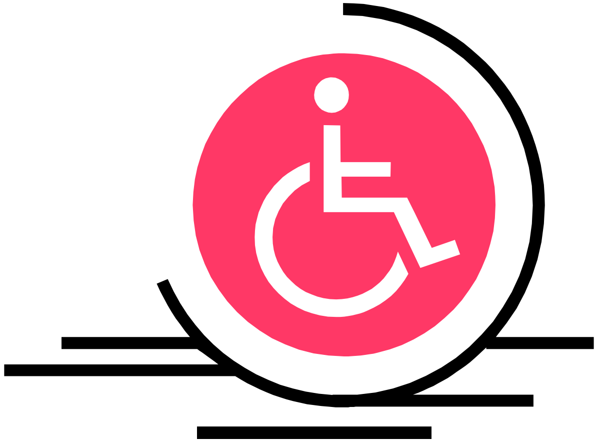
Online Certificate Status Protocol (OCSP)

- Returned status values are non-orthogonal

– Status = “good”, “revoked”, or “unknown”

– “Not revoked” doesn’t necessarily mean “good”

– “Unknown” could be anything from “Certificate was never issued” to “It was issued but I can’t find a CRL for it”



OCSP Problems

•Problems are due in some extent to the CRL-based origins of OCSP
– CRL can only report a negative result
– “Not revoked” doesn’t mean a cert was ever issued
– Some OCSP implementations will report “I can’t find a CRL” as “Good”
– Some relying party implementations will assume “revoked” “not good”, so any other status = “good”
– Much debate among implementors about OCSP semantics

Discussion