Key Distribution
Module2.3

Key Distribution

- **A** term that refers to the means of delivering a key to two parties who wish to exchange data, without allowing others to see the key
- It can be achieved in number of ways, as follows:

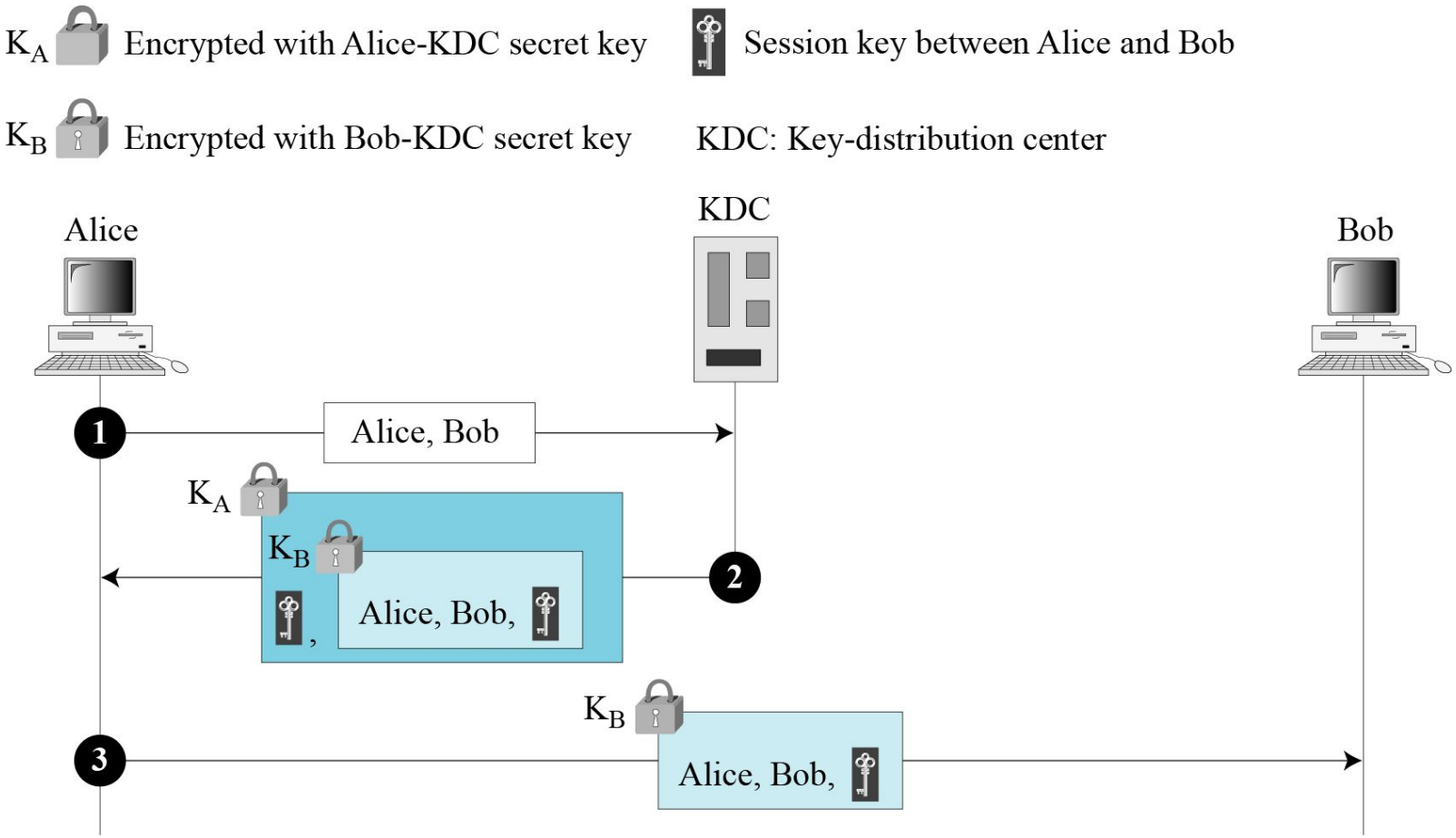
Key Distribution

- A can select a key and physically deliver it to B.
- A third party can select the key and physical deliver to A and B.
- If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using old Key.
- If A and B has encrypted connection to third party C, C can deliver the key on the encrypted links of A and B.

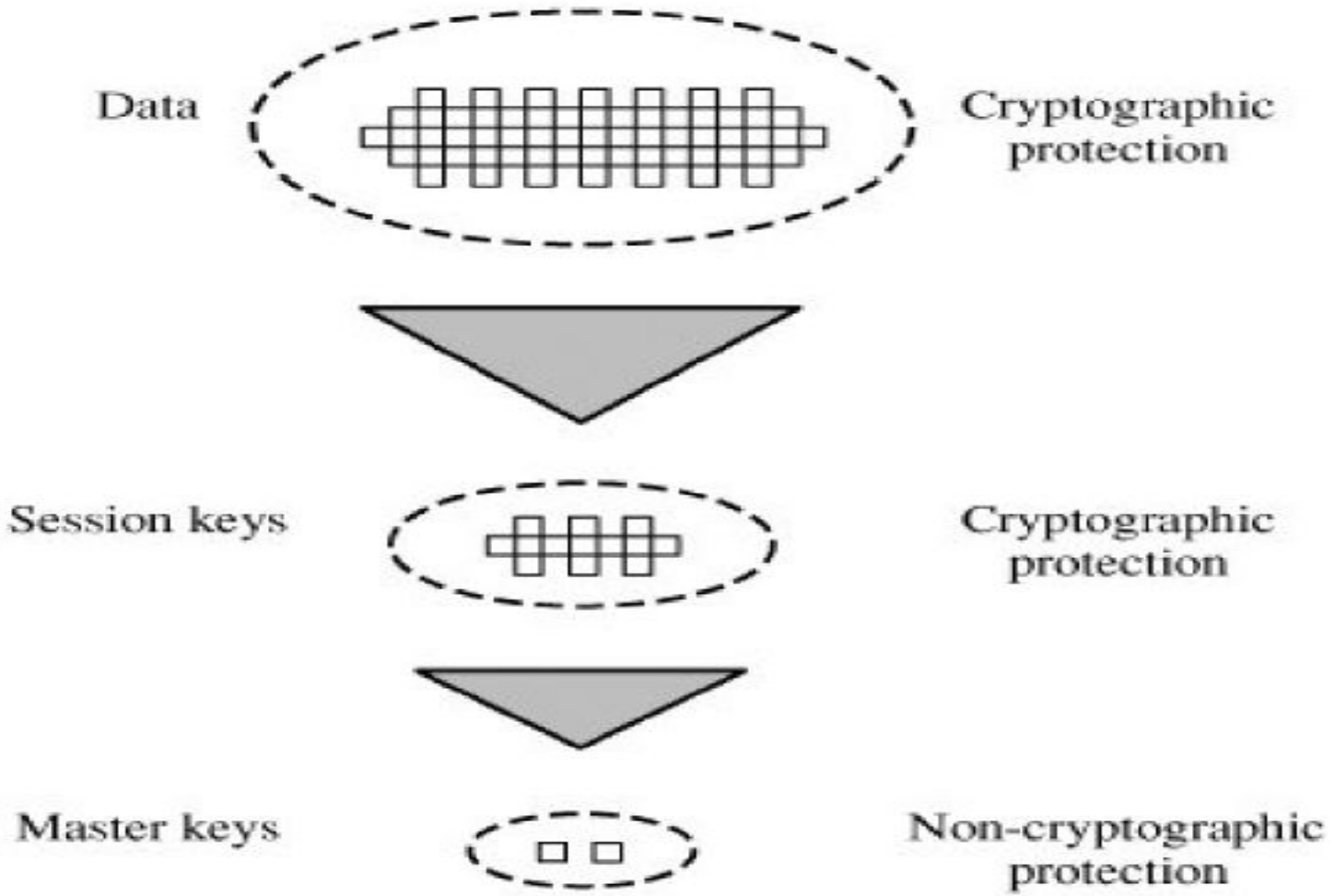
Key distribution

- A typical operation with a KDC involves a request from a user to use some service. The KDC will use cryptographic techniques to authenticate requesting users as themselves. It will also check whether an individual user has the right to access the service requested. If the authenticated user meets all prescribed conditions, the KDC can issue a ticket permitting access.
- KDCs mostly operate with symmetric Encryption.
- In most cases the KDC shares a key with each of all the other parties.
- The KDC produces a ticket based on a server key.
- The client receives the ticket and submits it to the appropriate server.
- The server can verify the submitted ticket and grant access to the user submitting it.
- Security systems using KDCs include **Kerberos**

A simple protocol using KDC



Use of Key Hierarchy



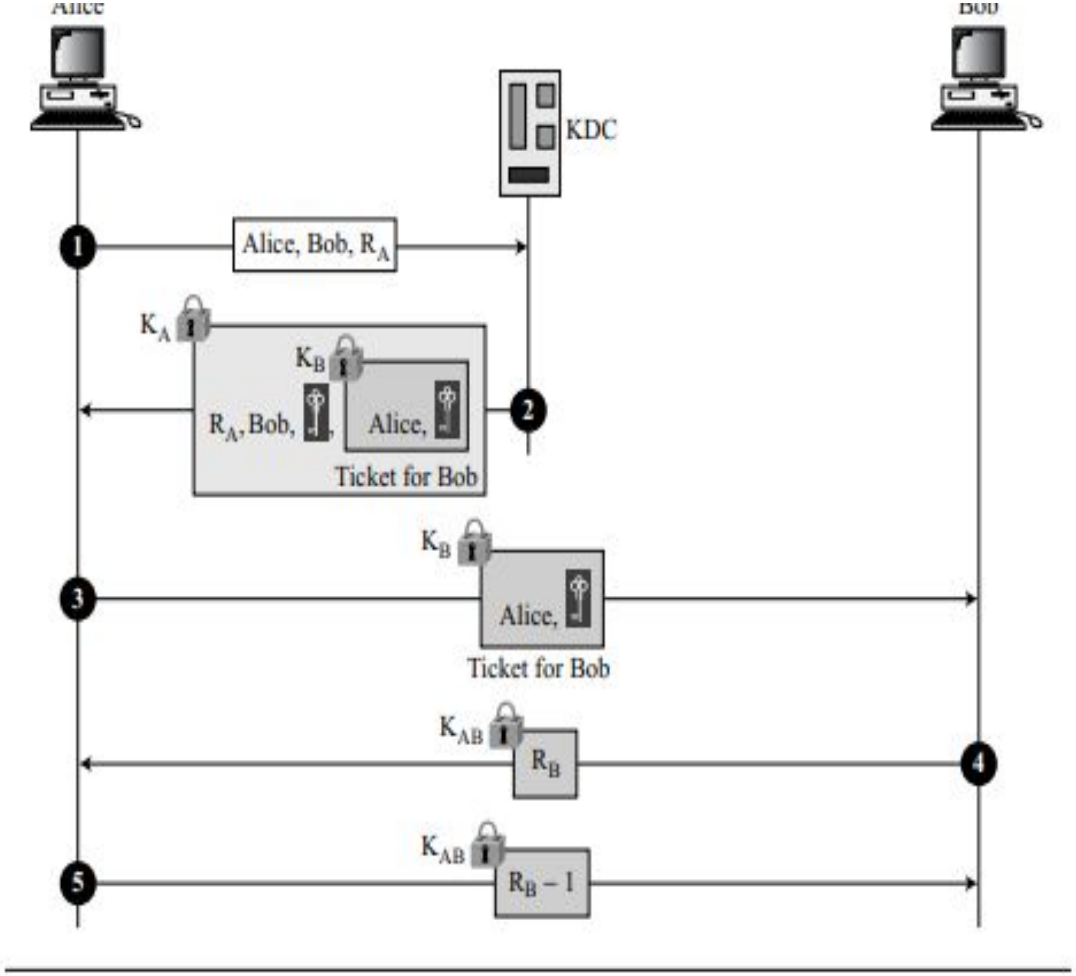
Use of Key Hierarchy

- **Session Key:** Temporary Key
- **Master Key:** There is a unique master key used for one complete session shared by key distribution center (KDC)
- For one Session: [N(N-1)]/2 session keys are required but N Master keys are used.

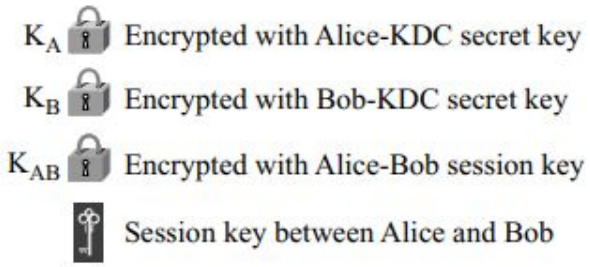
Needham Schroeder Protocol

- The *Needham–Schroeder Symmetric Key Protocol* is based on symmetric Encryption Algorithm
- It forms the basis for the Kerberos protocol. This protocol aims to establish a session key between two parties on a network, typically to protect further communication.
- The *Needham–Schroeder Public-Key Protocol*, based on public key cryptography.
- This protocol is intended to provide mutual authentication between two parties communicating on a network, but in its proposed form is insecure.

Needham Schroeder Protocol



Needham Schroeder Protocol

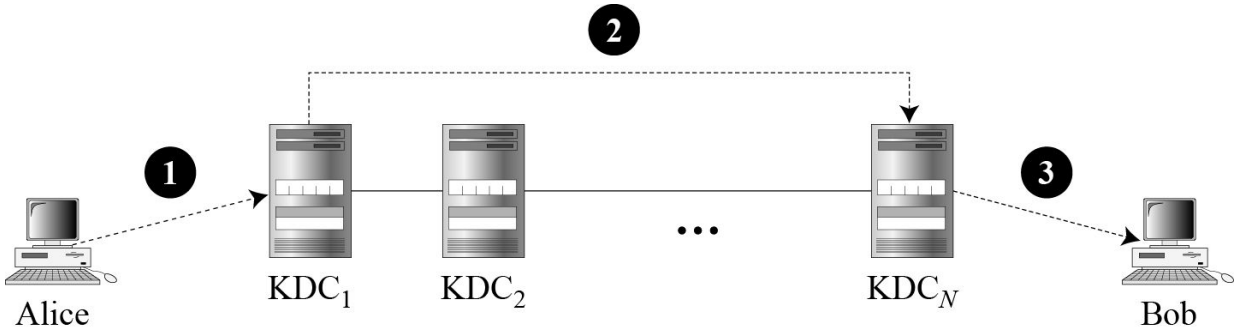


KDC: Key-distribution center
R_A: Alice's nonce
R_B: Bob's nonce

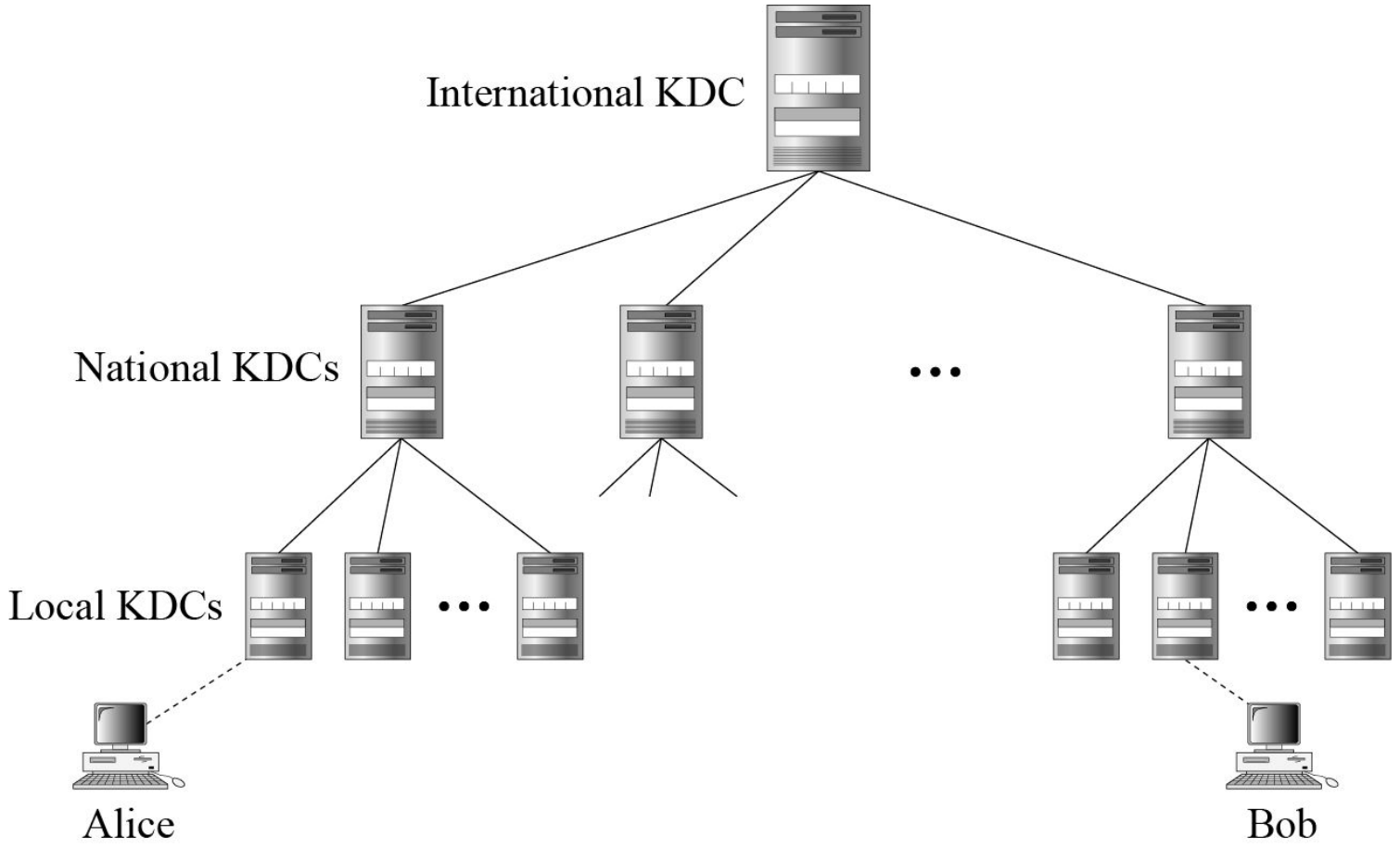
Needham Schroeder Protocol

1. Alice sends a message to Bob that includes a common nonce, R, the identities of Alice and Bob, and a ticket for KDC that includes Alice's nonce RA (a challenge for the KDC to use), a copy of the common nonce, R, and the identities of Alice and Bob.
2. Bob creates the same type of ticket, but with his own nonce RB. Both tickets are sent to the KDC.
3. The KDC creates a message that contains R, the common nonce, a ticket for Alice and a ticket for Bob; the message is sent to Bob. The tickets contain the corresponding nonce, RA or RB, and the session key, KAB.
4. Bob sends Alice her ticket.
5. Alice sends a short message encrypted with her session key KAB to show that she has the session key

Local KDC



Hierarchical KDC



Hierarchical Key Control

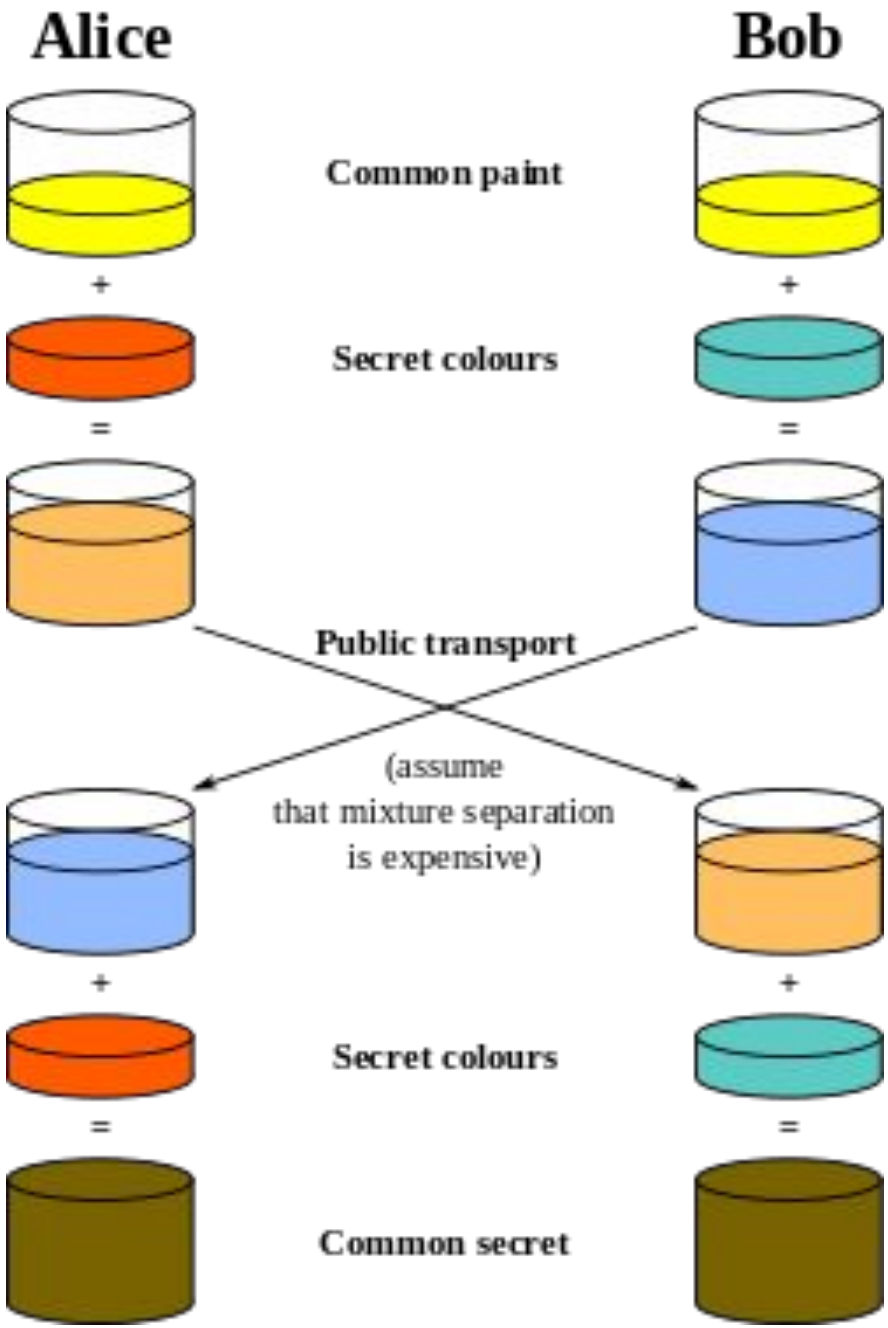
- Not to limit key distribution function to a single KDC
- A hierarchy of KDCs are used in Hierarchy
- For communication among entities within the same local domain, the local KDC is responsible for key distribution.
- If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC.
- In this case, any one of the three KDCs involved can actually select the key

Session key Lifetime

- More frequent exchange of session keys, more secure
- But distribution of session keys delays the start of any exchange and places a burden on network capacity
- For connection oriented protocol , use the same session key till the connection is open
- For connectionless protocol use a new session key for each exchange

Diffie-Hellman Key Exchange

- Suppose two people, Alice and Bob [traditional names], want to use insecure email to agree on a secret "shared key" that they can use to do further encryption for a long message. **How is that possible?**
- The so-called Diffie-Hellman method provides a way.
- The purpose of the algorithm to enable two users to exchange the key securely that then can be used for subsequent encryption of message.
- It is a practical method for public agreement of a secret key
- It is used in a number of commercial products
- EXCHANGE THE KEYS WITHOUT ACTUALLY EXCHANGING THEM


Alice
Bob
Common paint
+
Secret colours
=
Public transport
(assume
that mixture separation
is expensive)
+
Secret colours
=
Common secret

D H exchange procedure

- Diffie-Hellman key agreement protocol
  - Exponential key agreement
  - Allows two users to exchange a secret key
  - Requires no prior secrets
  - Real-time over an untrusted network
- Requires two large numbers, one prime (P), and (G), a primitive root of P
- P and G are both publicly available numbers
  - P is at least 512 bits
- Users pick private values a and b
- Compute public values
  - $x = g^a \bmod p$
  - $y = g^b \bmod p$
- Public values x and y are exchanged
- Compute shared, private key
  - $k_a = y^a \bmod p$
  - $k_b = x^b \bmod p$
- Algebraically it can be shown that $k_a = k_b$
  - Users now have a symmetric secret key to encrypt

D-H Key Exchange Algorithm

Global Public Elements

q	prime number
α	$\alpha < q$ *and* α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Calculation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Calculation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

Diffie-Hellman Key Exchange

Alice

Bob and Alice know and have the following :
$p = 23$ (*a prime number*) $g = 11$ (*a generator*)

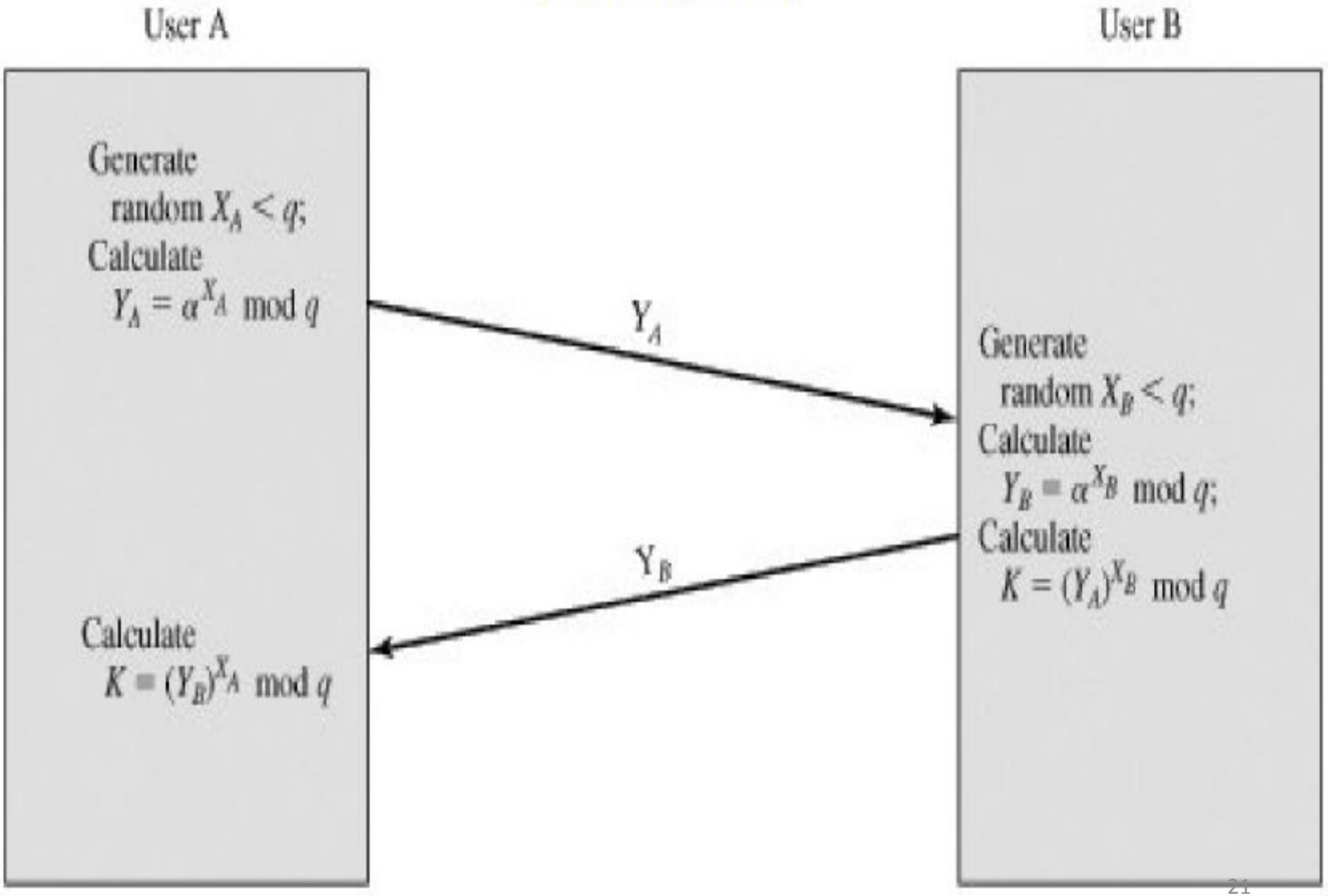
Bob

Alice	Bob
Alice chooses a secret random number a = 6	Bob chooses a secret random number b = 5
Alice computes : $A = g^a \bmod p$ $A = 11^6 \bmod 23 = 9$	Bob computes : $B = g^b \bmod p$ $B = 11^5 \bmod 23 = 5$
Alice receives B = 5 from Bob	Bob receives A = 9 from Alice
Secret Key = $K = B^a \bmod p$	Secret Key = $K = A^b \bmod p$
$K = 5^6 \bmod 23 = 8$	$K = 9^5 \bmod 23 = 8$

The common secret key is : 8

N.B. We could also have written : $K = g^{ab} \bmod$

Diffie-Hellman Key Exchange



Example

- Alice and Bob get public numbers
  - P = 23, G = 9
- Alice and Bob compute public values
  - $X = 9^4 \bmod 23 = 6561 \bmod 23 = 6$
  - $Y = 9^3 \bmod 23 = 729 \bmod 23 = 16$
- Alice and Bob exchange public numbers

Example

- Alice and Bob compute symmetric keys
  - $k_a = y^a \bmod p = 16^4 \bmod 23 = 9$
  - $k_b = x^b \bmod p = 6^3 \bmod 23 = 9$
- Alice and Bob now can talk securely!

Applications

- Diffie-Hellman is currently used in many protocols, namely:
  - Secure Sockets Layer (SSL)/Transport Layer Security (TLS)
  - Secure Shell (SSH)
  - Internet Protocol Security (IPSec)
  - Public Key Infrastructure (PKI)

Diffie-Hellman Example

- Alice & Bob who wish to swap keys: key exchange is based on the use of prime numbers, **prime q=353** and a **primitive root of 353, in this case α=3**
- They select the random secret keys:
  - A chooses x_A=97, B chooses x_B=233 respectively
- Each computes its public key:
  - $y_A = \alpha^{xA} \bmod q = 3^{97} \bmod 353 = 40$ (Alice)
  - $y_B = \alpha^{xB} \bmod q = 3^{233} \bmod 353 = 248$ (Bob)
- After they exchange public keys, each can compute common secrete keys as:

  $K_{AB} = y_B^{xA} \bmod 353 = 248^{97} \bmod 353 = 160$ (Alice)

  $K_{AB} = y_A^{xB} \bmod 353 = 40^{233} \bmod 353 = 160$ (Bob)
  - We assume that the attacker would have available the following information.
  - Q, α, Y_A and Y_B

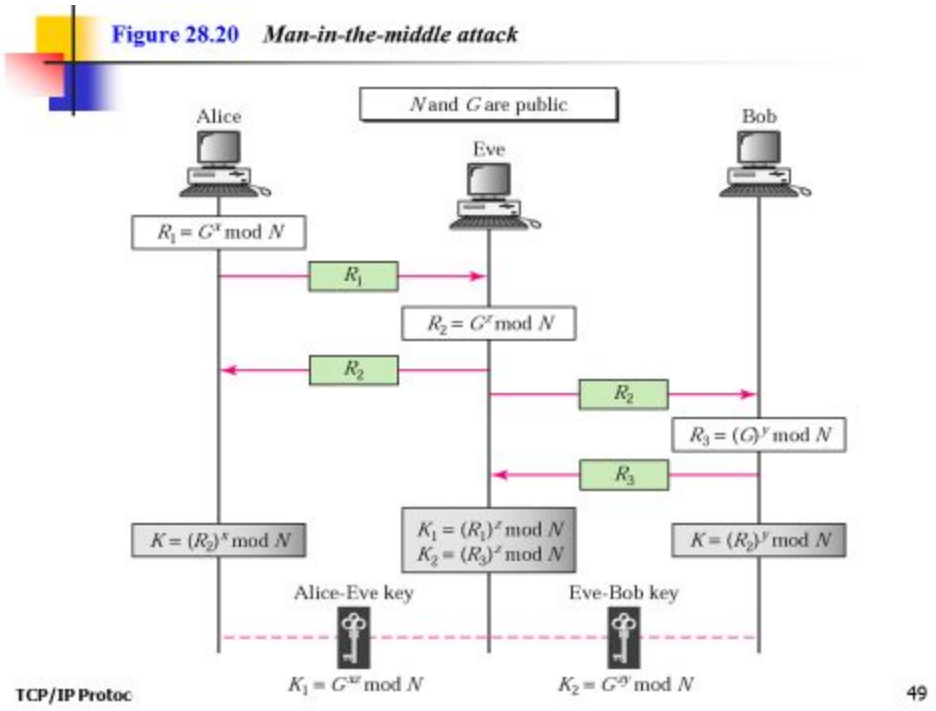
For calculations for modulus exponentiation refer to

https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/fast-modular-exponentiation

How to Calculate Primitive Root?

- **Let p be a prime. Then b is a *primitive root* for p** if the powers of b i e 1,b, b^2, b^3, …include all of the residue classes mod p
- **Examples:** If p=7, then 3 is a primitive root for p because the powers of 3 are 1, 3, 2, 6, 4, 5---that is, **every number mod 7 occurs except 0.** But 2 isn't a primitive root because the powers of 2 are 1, 2, 4, 1, 2, 4, 1, 2, 4...missing several values.
- **Example:** If p=13, then 2 is a primitive root because the powers of 2 are 2, 4, 8, 3, 6, 12, 11, 9, 5, 10, 7---which is all of the classes mod 13 except 0. There are other primitive roots for 13.

Man in the middle attack



Man in the middle Attack

- Darth prepares for the attack by generating two random private keys X^D1 and X^D2 and then computing the corresponding **public keys Y^D1 and Y^D2.**
- Alice transmits Y^A to Bob.
- Darth intercepts Y^A and transmits Y^D1 to Bob. Darth also calculates

  $K2 = (Y^A)\ X^D2 \bmod q$
- Bob receives Y_D1 and calculates $K1 = (Y^D1)\ X^B \bmod q$.

  Bob transmits X^A to Alice.
- Darth intercepts X^A and transmits Y^D2 to Alice. Darth calculates
- $K1 = (Y^B)\ X^D1 \bmod q$

  Alice receives Y^D2 and calculates $K2 = (Y^D2)\ X^A \bmod q$.
- At this point, Bob and Alice think that they share a secret key, but instead **Bob and Darth share secret key *K1*** *and* ***Alice and Darth share*** **secret key *K2.*** *All future communication between Bob and Alice is compromised in the following way:*

Man in the middle Attack

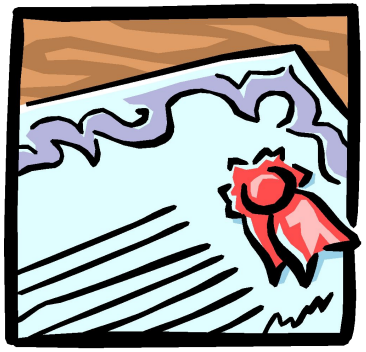
- Alice sends an encrypted message *M : E(K2, M)*.
- **Darth intercepts the encrypted message and decrypts it, to recover *M*** Darth sends Bob E(K1, M) or E(K1, M'), where M' is any message.
- **In the first case,** Darth simply wants to eavesdrop on the communication without altering it.
- **In the second case,** Darth wants to modify the message going to Bob.

NOTE: The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants.

Public Key Deception

- Impostor Claims to be a True Party
  - True party has a public and private key
  - Impostor also has a public and private key
- Impostor sends impostor's own public key to the verifier
  - Says, "This is the true party's public key"
  - This is the critical step in the deception

Digital Certificates



- Digital Certificate is a data with digital signature from one trusted Certification Authority (CA).
- This data contains:
  - Who owns this certificate
  - Who signed this certificate
  - The expired date
  - User name & email address
  - Subjects public key value

Certificate Authority

- A CA is an authority in a network that issues and manages security credentials and public key for message encryption
- As a part of PKI a CA checks with the RA to verify information about the requestor of the digital certificate
- If RA verifies the requestor information then CA can issue a certificate

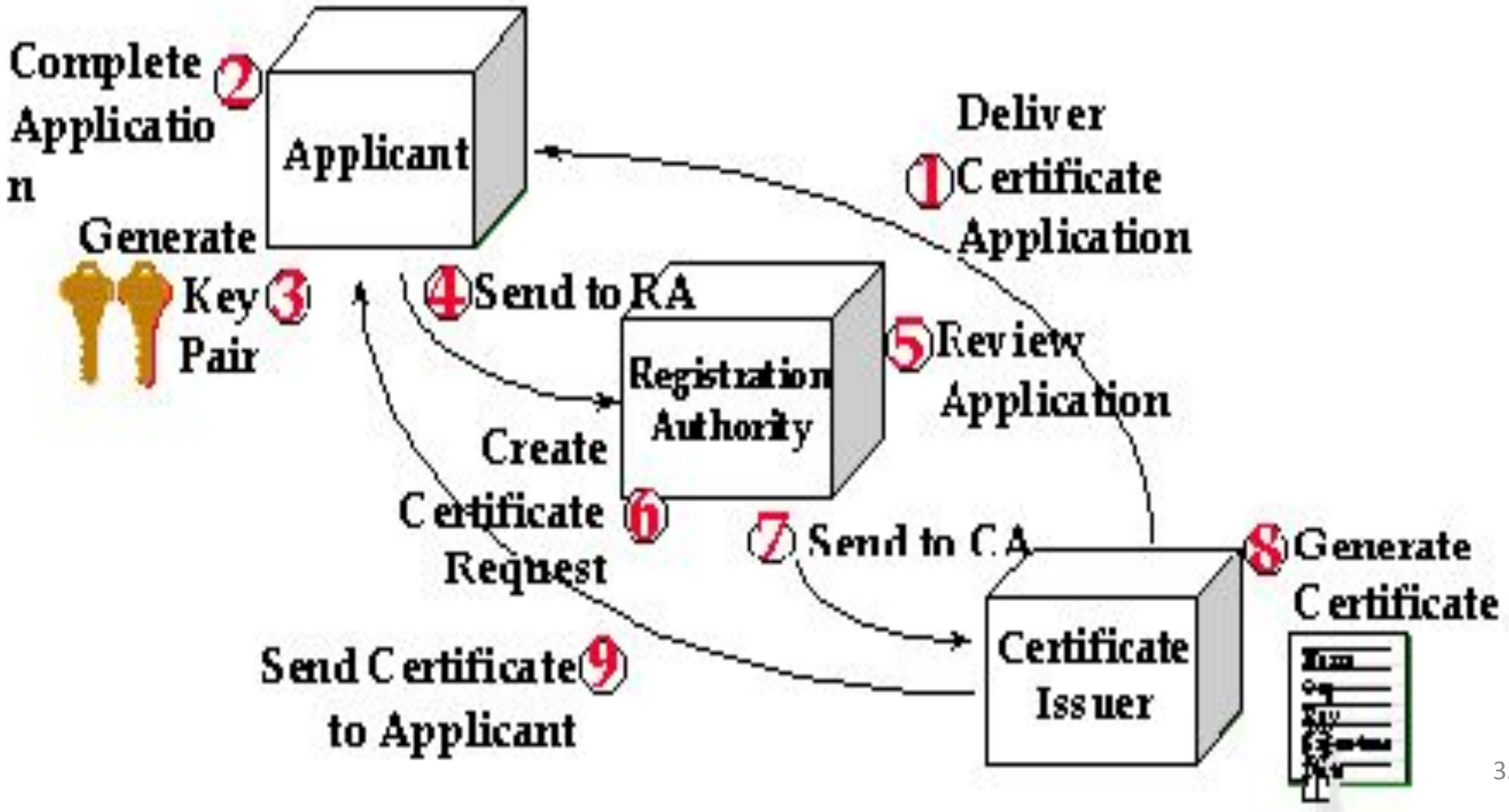
Certification Authority (CA)

- A trusted agent who certifies public keys for general use (Corporation or Bank).
  - User has to decide which CAs can be trusted.
- The model for key certification based on friends and friends of friends is called “Web of Trust”.
  - The public key is passing from friend to friend.
  - Works well in small or high connected worlds.
  - What if you receive a public key from someone you don’t know?

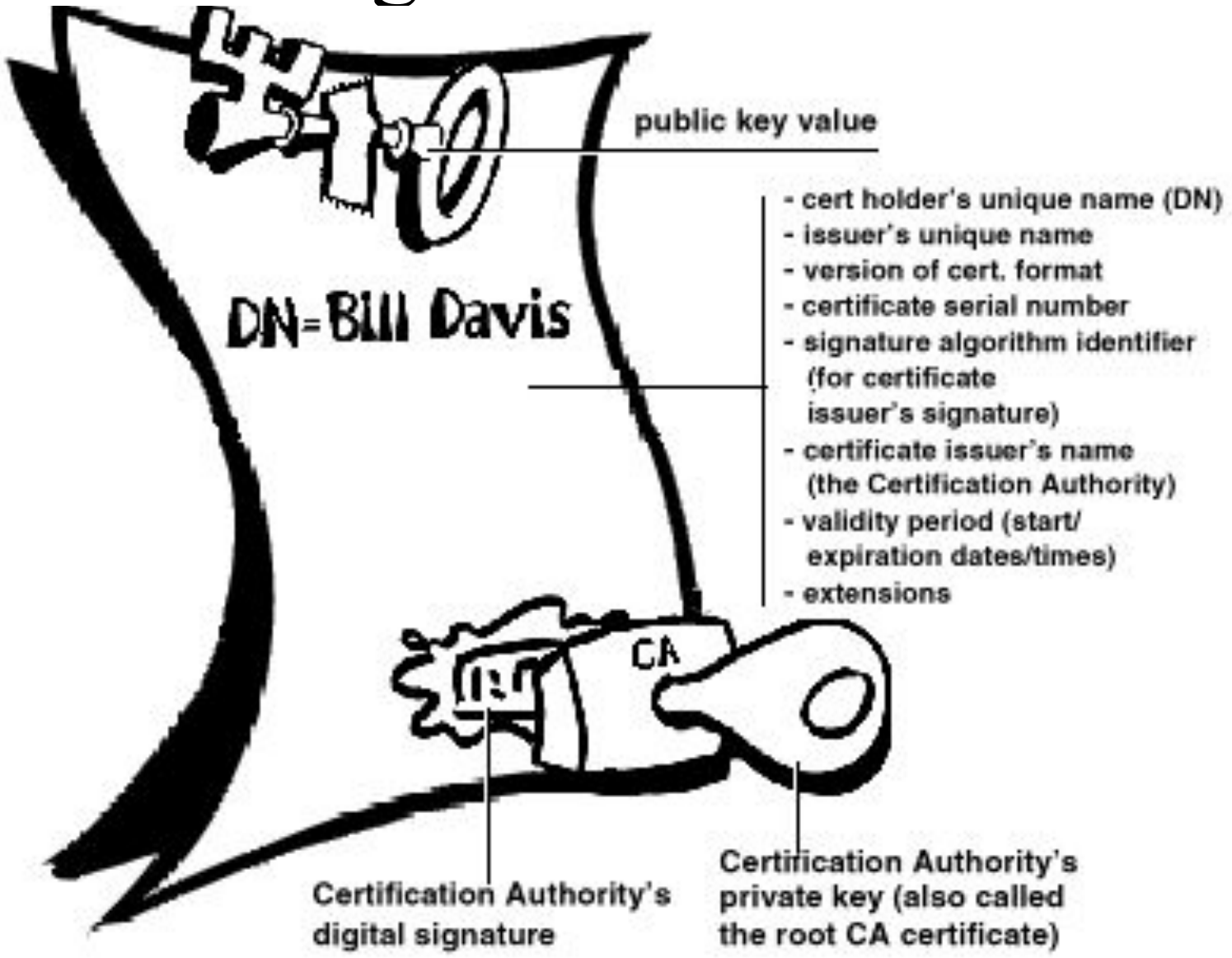
Certificate authority

- Certificate authorities may revoke digital certificates before the expiration date listed in the digital certificate
  - Revoked certificate ID numbers are placed in a *certificate revocation list (CRL)*
  - Verifier must check with the certificate authority to determine if a digital certificate is on the CRL
- Without the CRL check, digital certificates do not support authentication

Certification and Registration

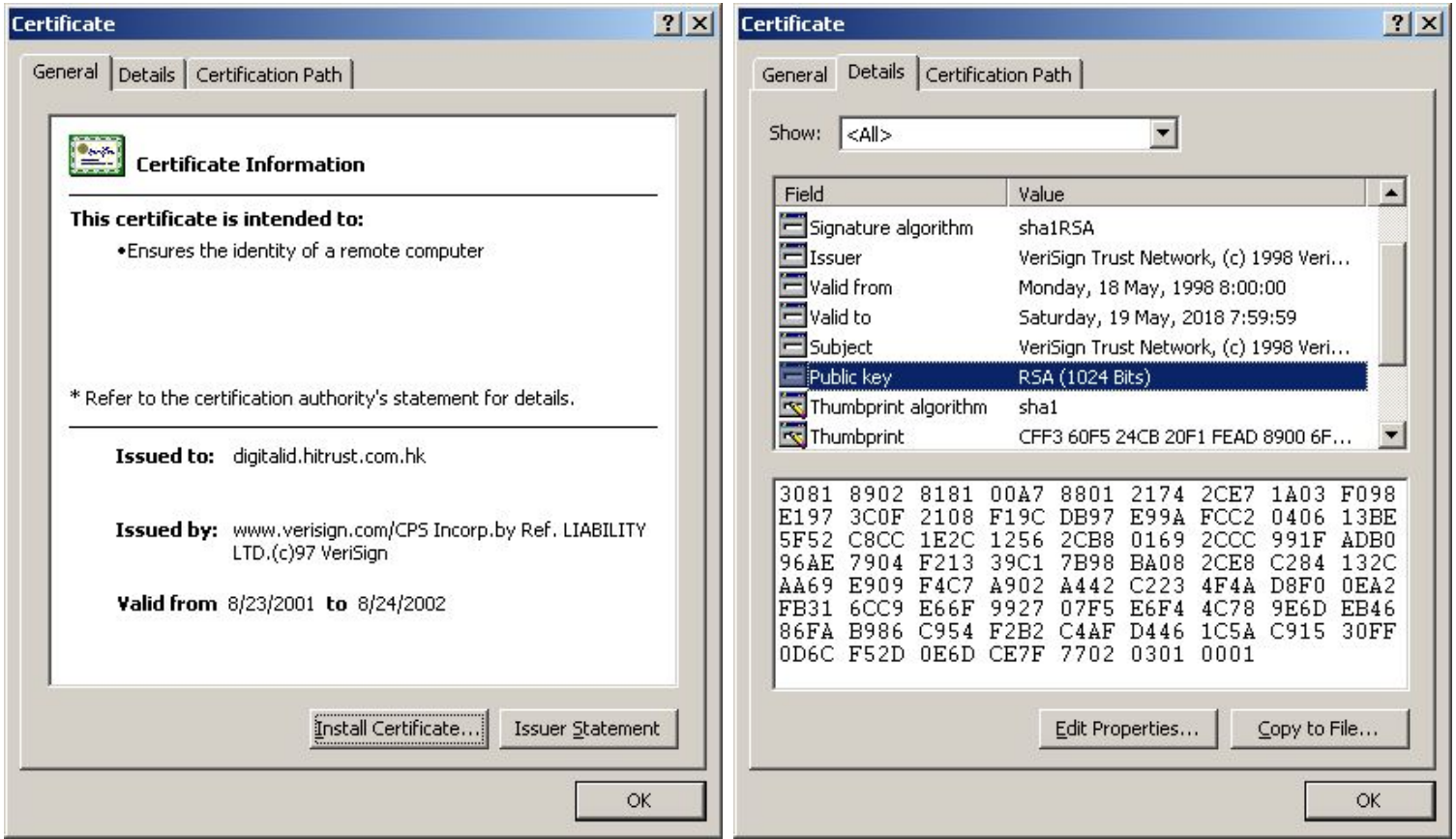


Digital Certificate



Elements of Digital Cert.

- A Digital ID typically contains the following information:
  - Your public key, Your name and email address
  - Expiration date of the public key, Name of the CA who issued your Digital ID



Types of Digital Certificates

- Personal Certificates
  - For verifying validity of user and clients
- Server Certificates
  - Normally for web servers
  - For identifying client and server applications
  - Requires for using SSL for secure communication
- Software Publisher Certificates
  - For authentic certificates
- Certificate Authority Certificates
  - Which validates the authority of a CA

Contents of Digital Certificate

Data Section

1. Serial number: used to uniquely identify certificate
2. Subject: the person or server or any entity identified
3. Signature Algorithm: algorithm used to create the signature
4. Issuer: the entity that verified the information and issued the certificate
5. Valid from: the date the certificate is first valid from
6. Valid to: the expiration date
7. Key usage: purpose of public key (signature/certificate signing/encryption)
8. Public key: the key value
9. SAN: subject alternative name, this field is optional. this is used to prove multiple identities which the certificate can authenticate. Can be used in case of cluster or server farm to simplify the support of the environment
10. Version: version of the certificate standard. e.g. "X.509 version 3

Signature Section

11. It contains the cipher or cryptographic algorithm used by issuing the CA to create its own digital signature

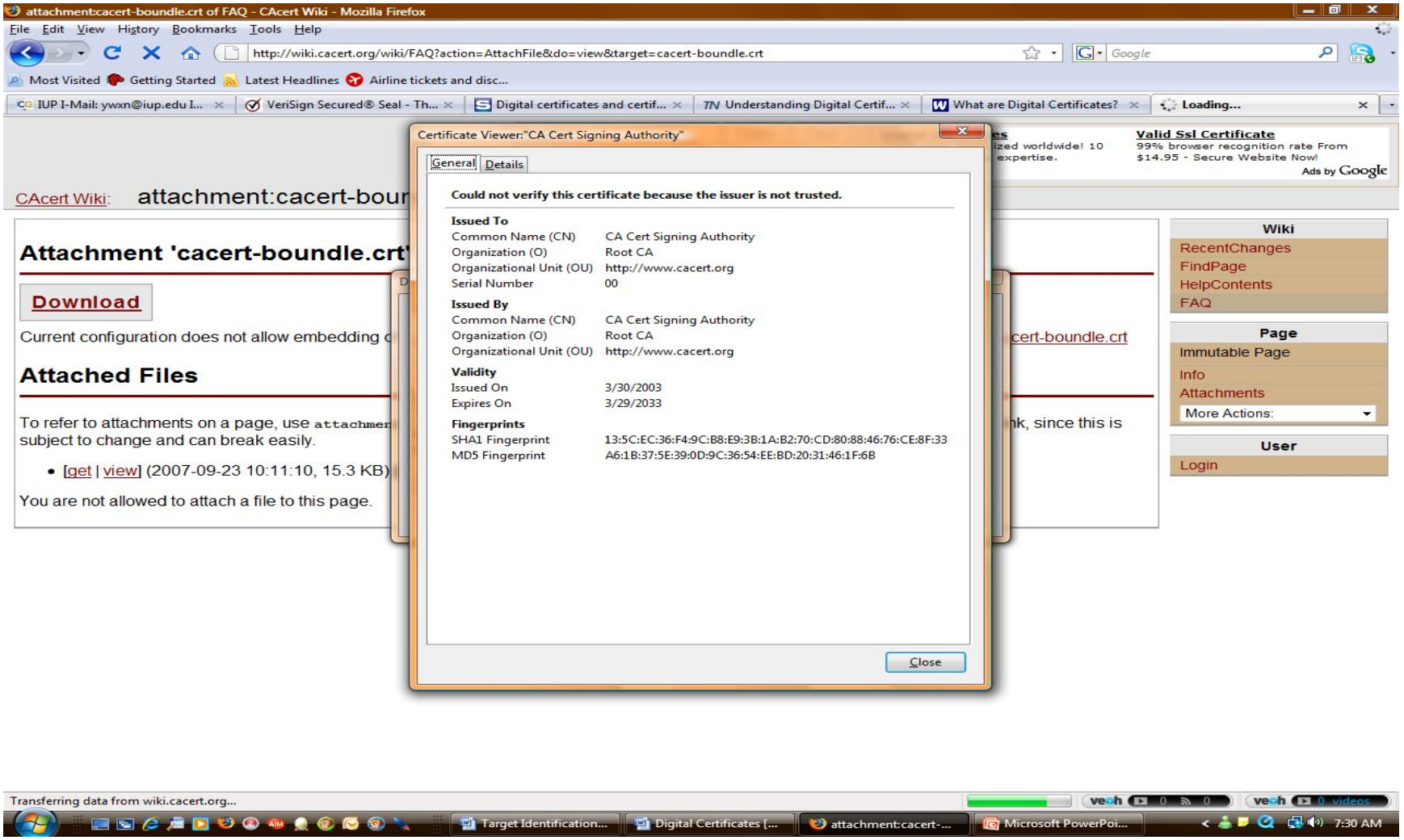
CAcert

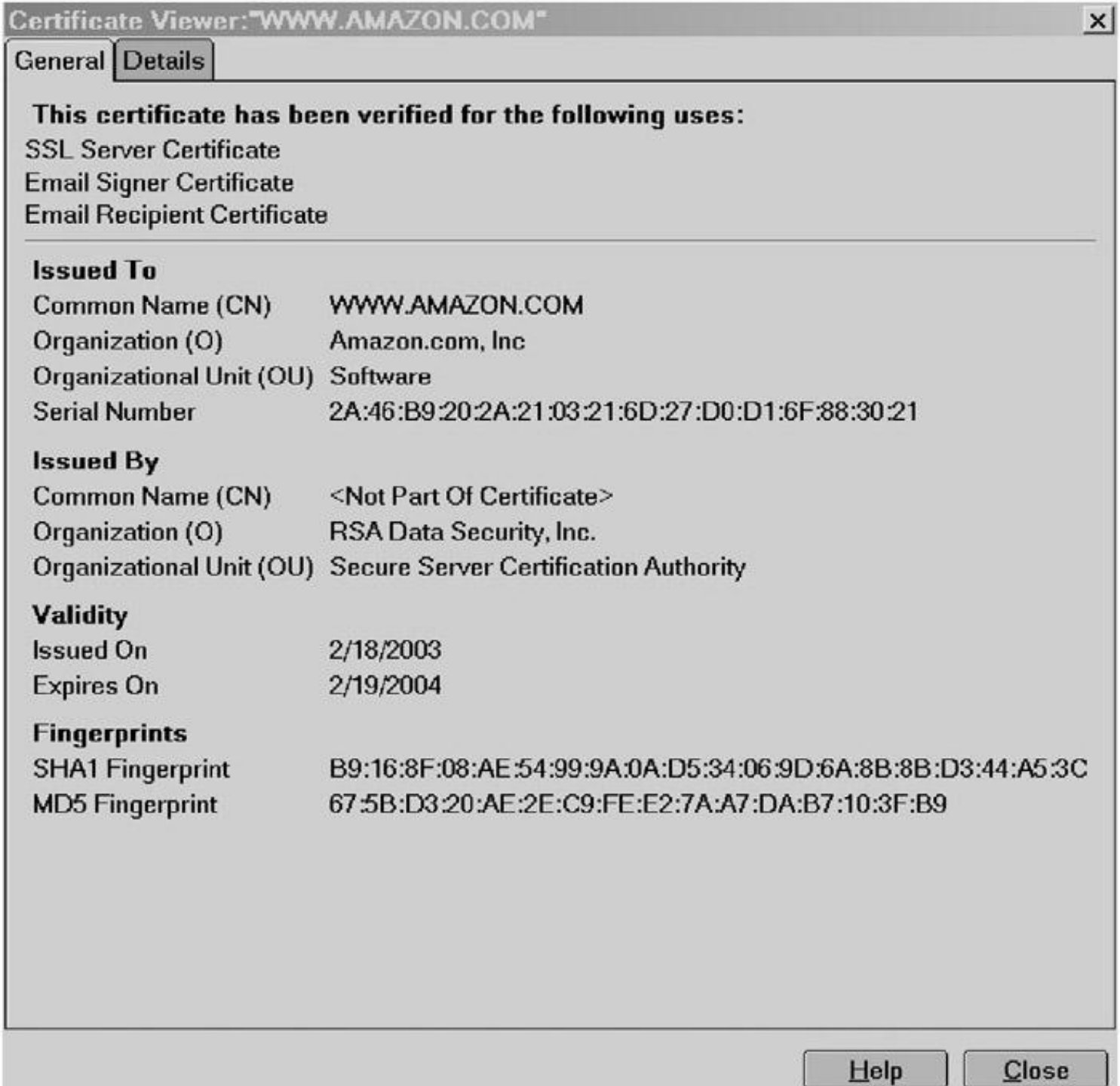
- Nonprofit Volunteer Organization
- Free Authentication
- Online registration; takes time

CAcert

- CAcert.org is a community-driven Certificate Authority that issues certificates to the public at large for free.
- CAcert's goal is to promote awareness and education on computer security through the use of encryption, specifically by providing cryptographic certificates. These certificates can be used to digitally sign and encrypt email, authenticate and authorize users connecting to websites and secure data transmission over the internet. Any application that supports the Secure Socket Layer Protocol (SSL or TLS) can make use of certificates signed by CAcert, as can any application that uses X.509 certificates, e.g. for encryption or code signing and document signatures.

CAcert cont:


attachment:cacert-boundle.crt of FAQ - CAcert Wiki - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://wiki.cacert.org/wiki/FAQ?action=AttachFile&do=view&target=cacert-boundle.crt
Most Visited Getting Started Latest Headlines Airline tickets and disc...
IUP I-Mail: ywxn@iup.edu I...
VeriSign Secured® Seal - Th...
Digital certificates and certif...
Understanding Digital Certif...
What are Digital Certificates?
Loading...
Certificate Viewer:"CA Cert Signing Authority"
General Details
Could not verify this certificate because the issuer is not trusted.
Issued To
Common Name (CN) CA Cert Signing Authority
Organization (O) Root CA
Organizational Unit (OU) http://www.cacert.org
Serial Number 00
Issued By
Common Name (CN) CA Cert Signing Authority
Organization (O) Root CA
Organizational Unit (OU) http://www.cacert.org
Validity
Issued On 3/30/2003
Expires On 3/29/2033
Fingerprints
SHA1 Fingerprint 13:5C:EC:36:F4:9C:B8:E9:3B:1A:B2:70:CD:80:88:46:76:CE:8F:33
MD5 Fingerprint A6:1B:37:5E:39:0D:9C:36:54:EE:BD:20:31:46:1F:6B
Close
Valid Ssl Certificate
99% browser recognition rate From $14.95 - Secure Website Now!
Ads by Google
CAcert Wiki: attachment:cacert-boun
Attachment 'cacert-boundle.crt'
Download
Current configuration does not allow embedding
Attached Files
To refer to attachments on a page, use attachmen
subject to change and can break easily.
[get | view] (2007-09-23 10:11:10, 15.3 KB)
You are not allowed to attach a file to this page.
Wiki
RecentChanges
FindPage
HelpContents
FAQ
Page
Immutable Page
Info
Attachments
More Actions:
User
Login
Transferring data from wiki.cacert.org...
Target Identification...
Digital Certificates [...
attachment:cacert-...
Microsoft PowerPoi...
7:30 AM


Certificate Viewer:"WWW.AMAZON.COM"
General
Details
This certificate has been verified for the following uses:
SSL Server Certificate
Email Signer Certificate
Email Recipient Certificate
Issued To
Common Name (CN) WWW.AMAZON.COM
Organization (O) Amazon.com, Inc
Organizational Unit (OU) Software
Serial Number 2A:46:B9:20:2A:21:03:21:6D:27:D0:D1:6F:88:30:21
Issued By
Common Name (CN) <Not Part Of Certificate>
Organization (O) RSA Data Security, Inc.
Organizational Unit (OU) Secure Server Certification Authority
Validity
Issued On 2/18/2003
Expires On 2/19/2004
Fingerprints
SHA1 Fingerprint B9:16:8F:08:AE:54:99:9A:0A:D5:34:06:9D:6A:8B:8B:D3:44:A5:3C
MD5 Fingerprint 67:5B:D3:20:AE:2E:C9:FE:E2:7A:A7:DA:B7:10:3F:B9
Help
Close

VeriSign

- Leading provider for online encryption with approximately 99% of all internet users
- Offers 4 different certificate options
- Secure Site, Secure Site with EV, Secure Site Pro, Secure Site Pro with EV
- Secure Site cheapest at $399 for year validity

Kerberos

- Kerberos is an **Authentication Protocol** based on conventional encryption(also a KDC)
- Kerberos is an authentication service designed for use in a **distributed environment.**
- Kerberos **makes use of a trusted third-party authentication** service that enables clients and servers to establish authenticated communication.
- **X.509 defines** this authentication Algorithm.
- systems, including Windows 2000, use Kerberos.

Why Kerberos?

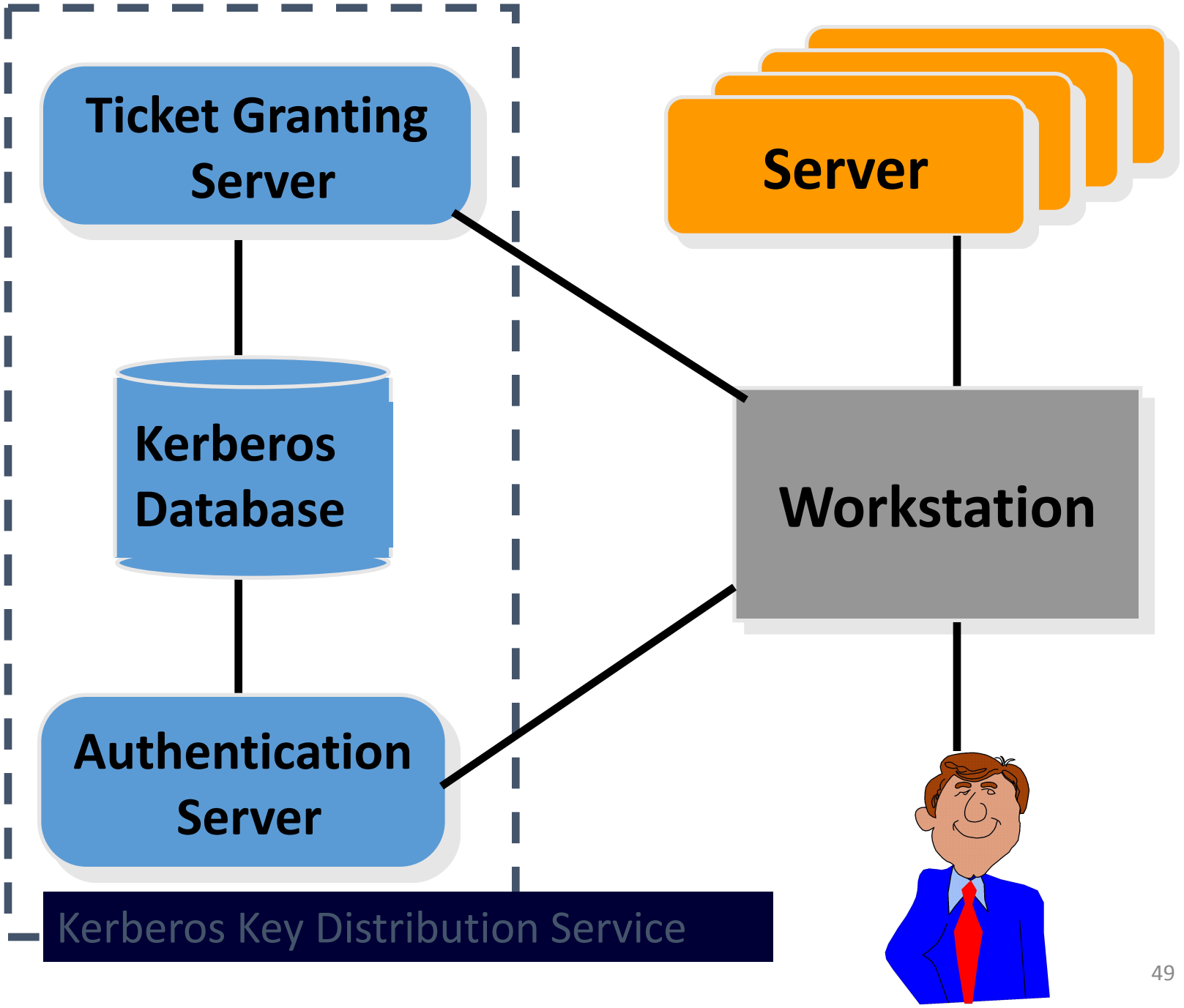
- A **user may gain access** to a particular workstation and **pretend to be another user** operating from that workstation.
- A **user may alter the network address** of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
- A **user may eavesdrop on exchanges and use a replay attack** to gain entrance to a server or to disrupt operations.
- To avoid this problem, **Kerberos provides a centralized authentication server** whose function is to authenticate users to servers and servers to users.

Requirements of Kerberos

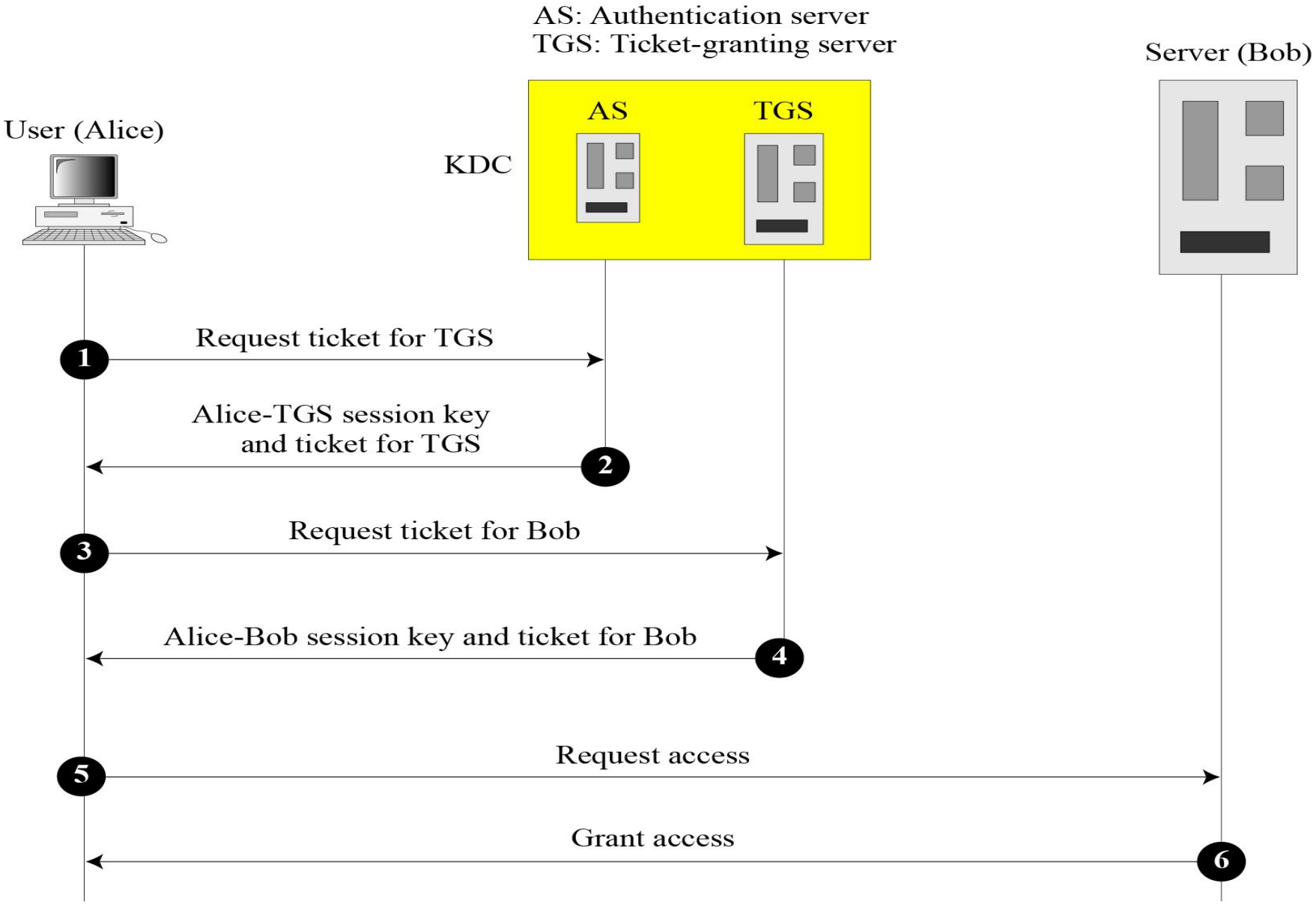
- The first published report on Kerberos listed the requirements for Kerberos as:
  - **Security** (i.e. eavesdropper should not be able to obtain information to impersonate a user)
  - **Reliability** (uses distributed server architecture with one system able to back up another and thus has high reliability)
  - **Transparency** (user does not know authentication is taking place beyond requirement of password)
  - **Scalability** (should support large number of clients and servers)
- implemented using an authentication protocol based on Needham-Schroeder

Versions of Kerberos

- Trusted key server system developed in MIT
- Provides centralized private-key third-party authentication in a distributed network
  - allows users access servers to services distributed through network
  - without needing to trust all workstations (because an attacker may alter the network ID of the workstation or pretend to be someone else)
  - rather all, trust a central authentication server
- Two versions in use: 4 & 5

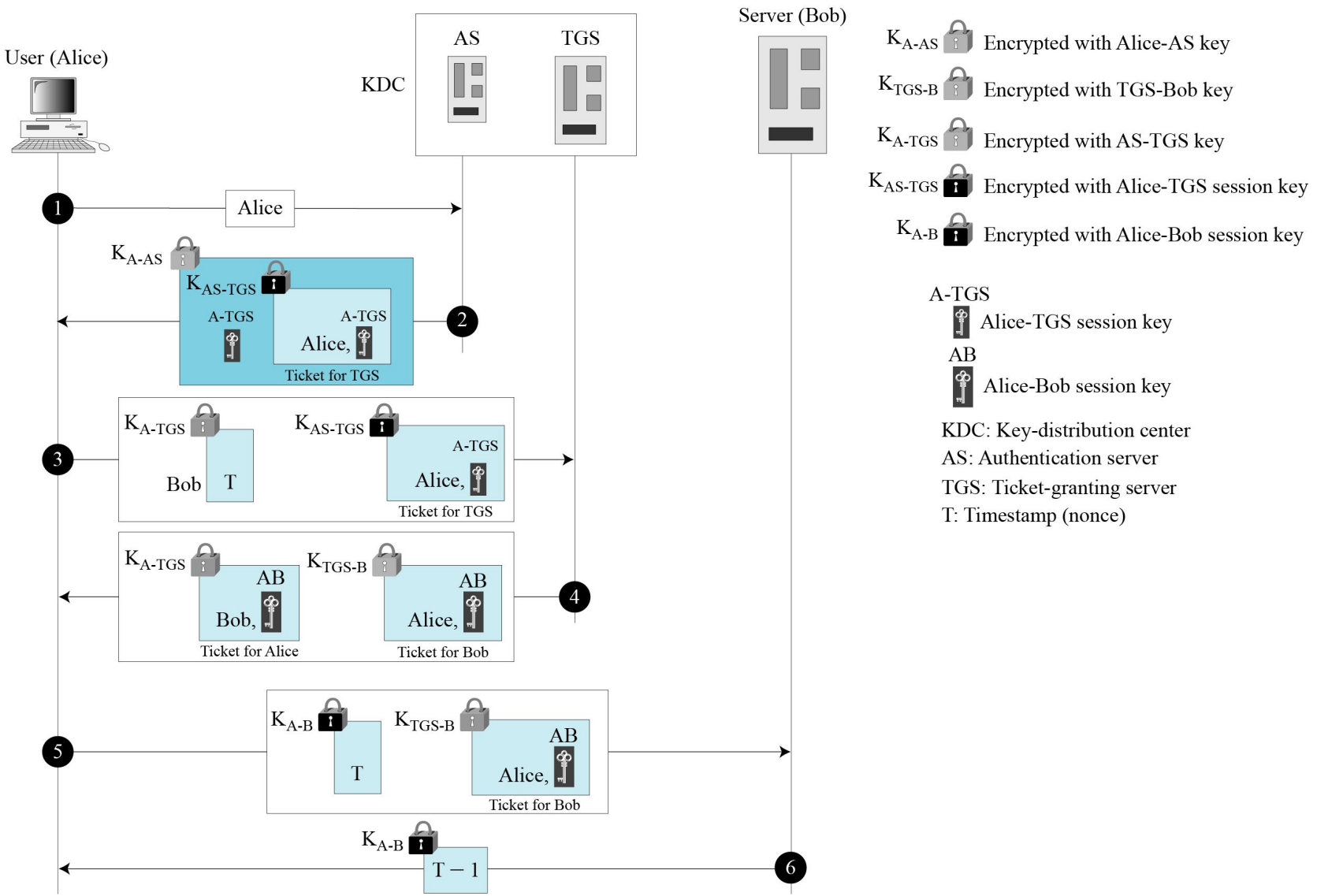

Ticket Granting Server
Server
Kerberos Database
Workstation
Authentication Server
Kerberos Key Distribution Service

Kerberos Servers



Kerberos

- Authentication Server (AS)
- The authentication server (AS) is the KDC in the Kerberos protocol.
- Ticket-Granting Server (TGS)
- The ticket-granting server (TGS) issues a ticket for the real server (Bob).
- Real Server
- The real server (Bob) provides services for the user (Alice).


User (Alice)
KDC
AS
TGS
Server (Bob)
1
Alice
2
K_{A-AS}
K_{AS-TGS}
A-TGS
A-TGS
Alice,
Ticket for TGS
3
K_{A-TGS}
Bob
T
K_{AS-TGS}
A-TGS
Alice,
Ticket for TGS
4
K_{A-TGS}
AB
Bob,
Ticket for Alice
K_{TGS-B}
AB
Alice,
Ticket for Bob
5
K_{A-B}
T
K_{TGS-B}
AB
Alice,
Ticket for Bob
6
K_{A-B}
T − 1
K_{A-AS} Encrypted with Alice-AS key
K_{TGS-B} Encrypted with TGS-Bob key
K_{A-TGS} Encrypted with AS-TGS key
K_{AS-TGS} Encrypted with Alice-TGS session key
K_{A-B} Encrypted with Alice-Bob session key
A-TGS
Alice-TGS session key
AB
Alice-Bob session key
KDC: Key-distribution center
AS: Authentication server
TGS: Ticket-granting server
T: Timestamp (nonce)

Kerberos Realms and Multiple Kerberi

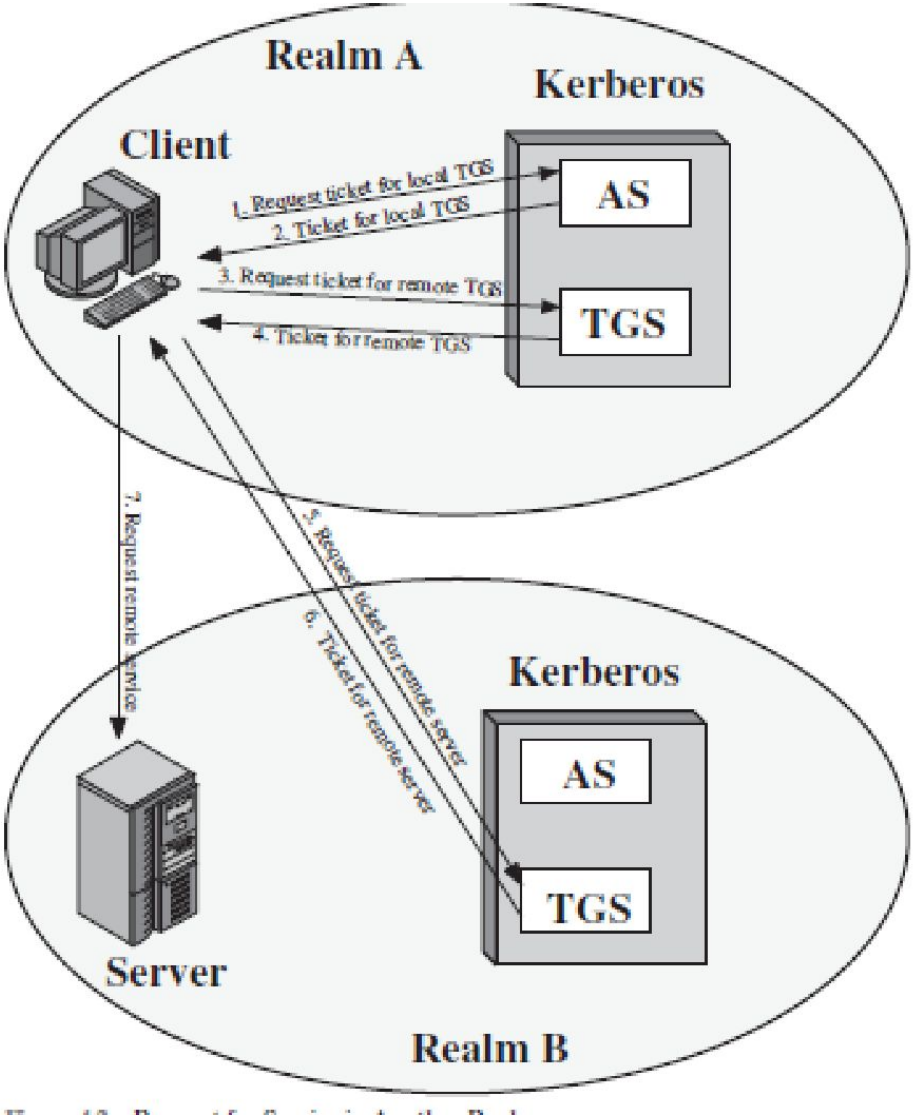
A full-service Kerberos environment consisting of **a Kerberos server, a number of clients,** and **a number of application servers** require the following:

The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.

The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server. Such an Environment is known as **Kerberos Realm.**

The Kerberos server in each interoperating realm shares a secret key with the server in the other realm.

Kerberos Realm



Kerberos Version 5

- Developed in mid 1990's
- Provides improvements over Version 4 (technical deficiencies)
- Specified as Internet standard RFC 1510

Difference Between Version 4 and 5

Encryption system dependence

Any encryption algorithms can be used in v5 but only DES is possible in v4

Internet protocol dependence

Only IP is possible through v4. But Version 5 network addresses are tagged with type and length, allowing any network address type to be used.

Message byte ordering

In v4 the sender of a message employs a byte ordering of its own choosing and tags the message but v5 uses Abstract Syntax Notation 1 (ASN.1)

ASN.1 encoding rules facilitates exchange of structured data

Difference Between Version 4 and 5…

- **Ticket Life Time:**
  - Lifetime values in v4 are encoded in an **8-bit quantity in units of five minutes.** Thus, the maximum lifetime that can be expressed is 2^8x 5 = 1280 minutes, or a little over 21 hours. In version 5, tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.
- **Authentication forwarding:**
  - Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. **This capability would enable a client to access a server and have that server access another server on behalf of the client.** For example, a client issues a request to a print server that then accesses the client's file from a file server, using the client's credentials for access. Version 5 provides this capability.
- **Interrealm Authentication:**
  - Interoperability between N realms requires N^2 relationships but for Version 5 very few relationships

Major differences

1)Version 5 has a longer ticket lifetime.

2)Version 5 allows tickets to be renewed.

3)Version 5 can accept any symmetric-key algorithm.

4)Version 5 uses a different protocol for describing data types.

5)Version 5 has more overhead than version 4.

Four New Elements in V5

- **Realm**: Indicate realm of user
- **Options**: Used to request that certain flags to be set in the returned ticket
- **Times**: Used by a client to request a time setting
  - from: the desired start time for the requested ticket
  - till: the requested expiration time for the requested ticket
  - rtime: requested renew-till time
- **Nonce**: A random value to be repeated in Message (2) to assure that the response is fresh and has not been replayed by an opponent

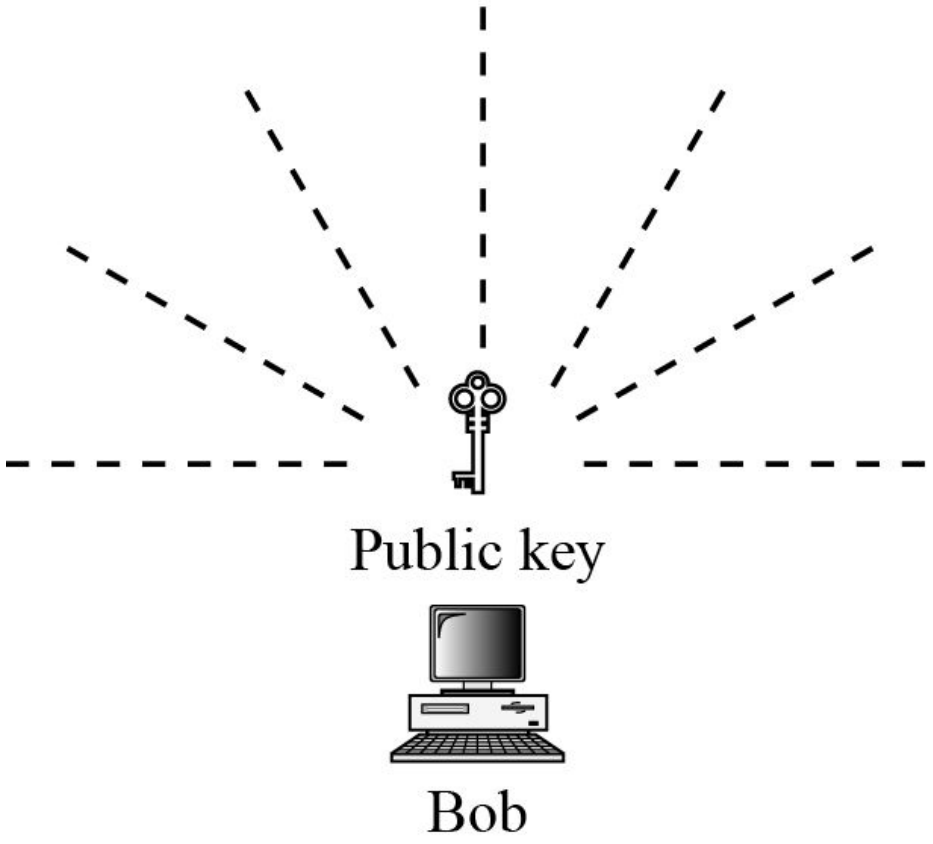
Three New Features in V5

- **Subkey**: This reflects the client's choice of encryption key to be used to protect this specific application session.
- **Sequence Number**: An optional field to specify the starting sequence number to be used by the server. This is used to detect replays.
- **Ticket Flags**: These flags expands functionalities compared to V4. e.g. who issued ticket, if authentication was done, ticket replacement request

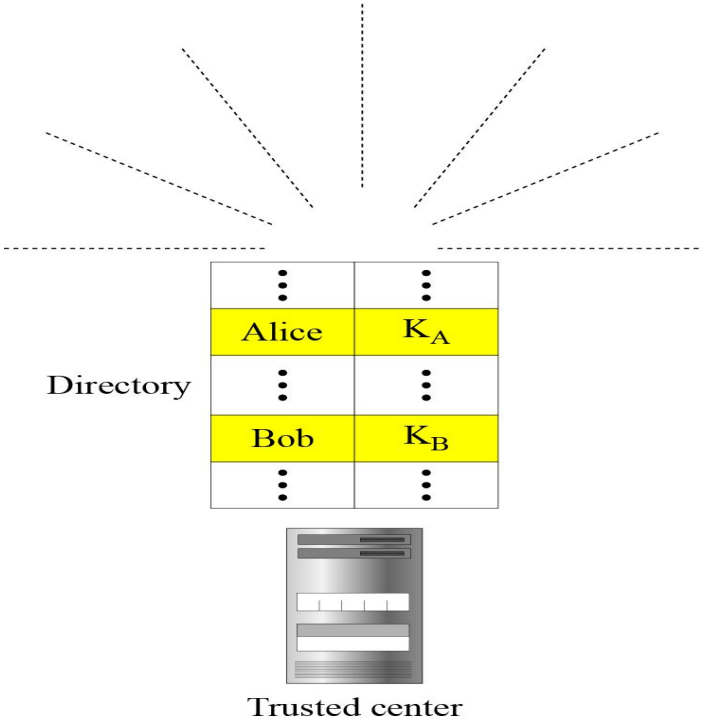
Public Key distribution

In asymmetric-key cryptography, people do not need to know a symmetric shared key; everyone shields a private key and advertises a public key.

Announcing the public key

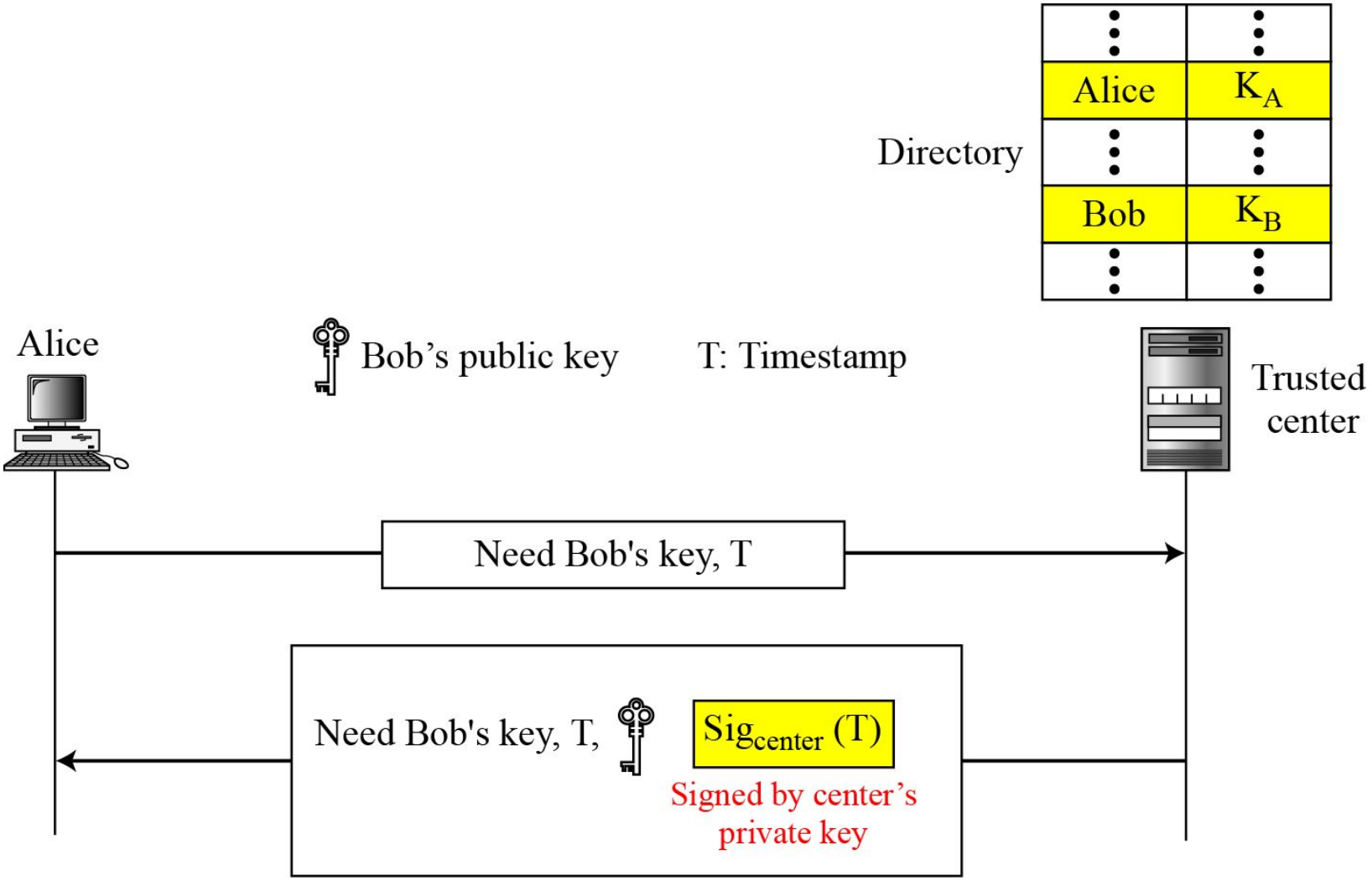


Trusted Centre

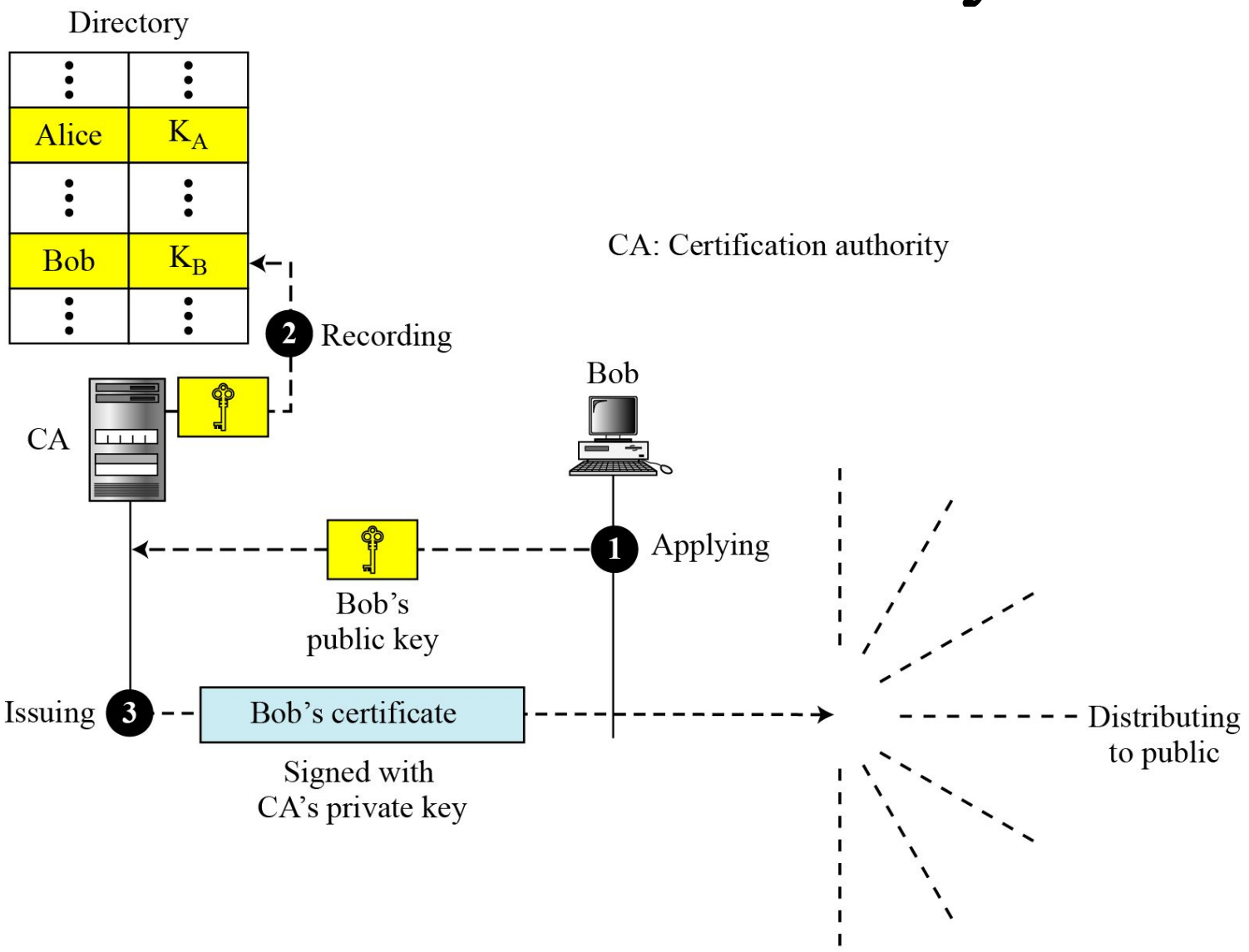


A directory of public key can be maintained
Each user should register in the centre and prove his or her entity
The directory can be publicly advertised

Controlled Trusted Centre



Certification Authority



Certificate Authority

- Bob wants two things
  - Public key should be made public
  - Also no one should accept a forged key
- Bob can go to certificate authority
- CA has a well known public key itself which cannot be forged
- CA checks Bob's identification
- Then writes Bob's private key on the certificate
- To prevent it from being forged it signs with its own private key
- This certificate is uploaded
- Anyone who wants Bob's public key should download the signed certificate and use the CA's public key t oextract Bob's public key

X.509 Authentication Service

X.509 Certificate

- Distributed set of servers that maintains a database about users.
- Each certificate contains the public key of a user and is signed with the private key of a CA.
- Is used in e.g. S/MIME, IP Security, SSL/TLS and SET.
- RSA is recommended to use.

X.509 Certificate

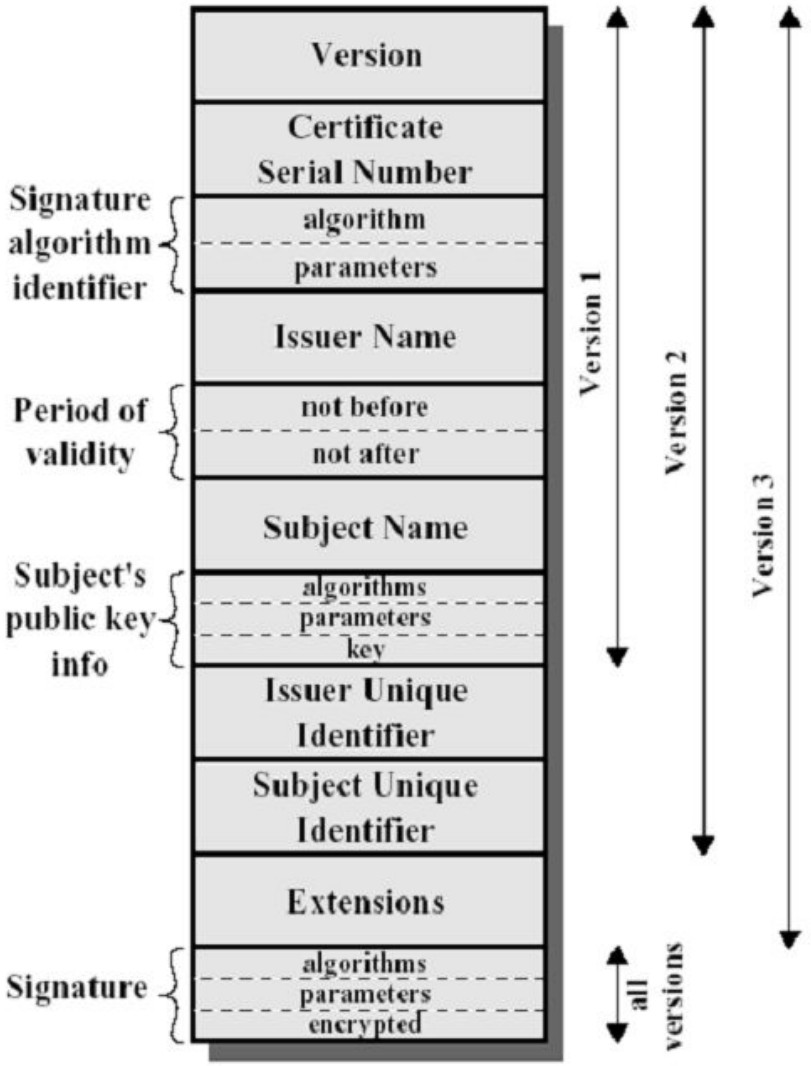
- An X.509 certificate is a digital certificate that uses the widely accepted international X.509 public key infrastructure (PKI) standard to verify that a public key belongs to the user, computer or service identity contained within the certificate.
- It is used by S/MIME secure email, SSL/TLS secure Internet links (eg for secure web). S/MIME: Secure/Multipurpose Internet Mail Extension

X.509 Authentication Service (Directory Service Standard)

- Part of ITU-T X.500 **directory service** standards
  - **The directory service is a server or distributed set of servers maintains some databases of information about the users (e.g.** the information includes mapping user name to network address)
  - **This directory may store public-key certificates**
    - public key of user
    - signed by certification authority
- **X. 509 uses public-key crypto & digital signatures**
  - Standard does not dictate the use of a specific algorithm , **but RSA recommended**
- X.509 certificate format is used in many services such as email services **S/MIME and IP security, SSL/TLS, etc.**

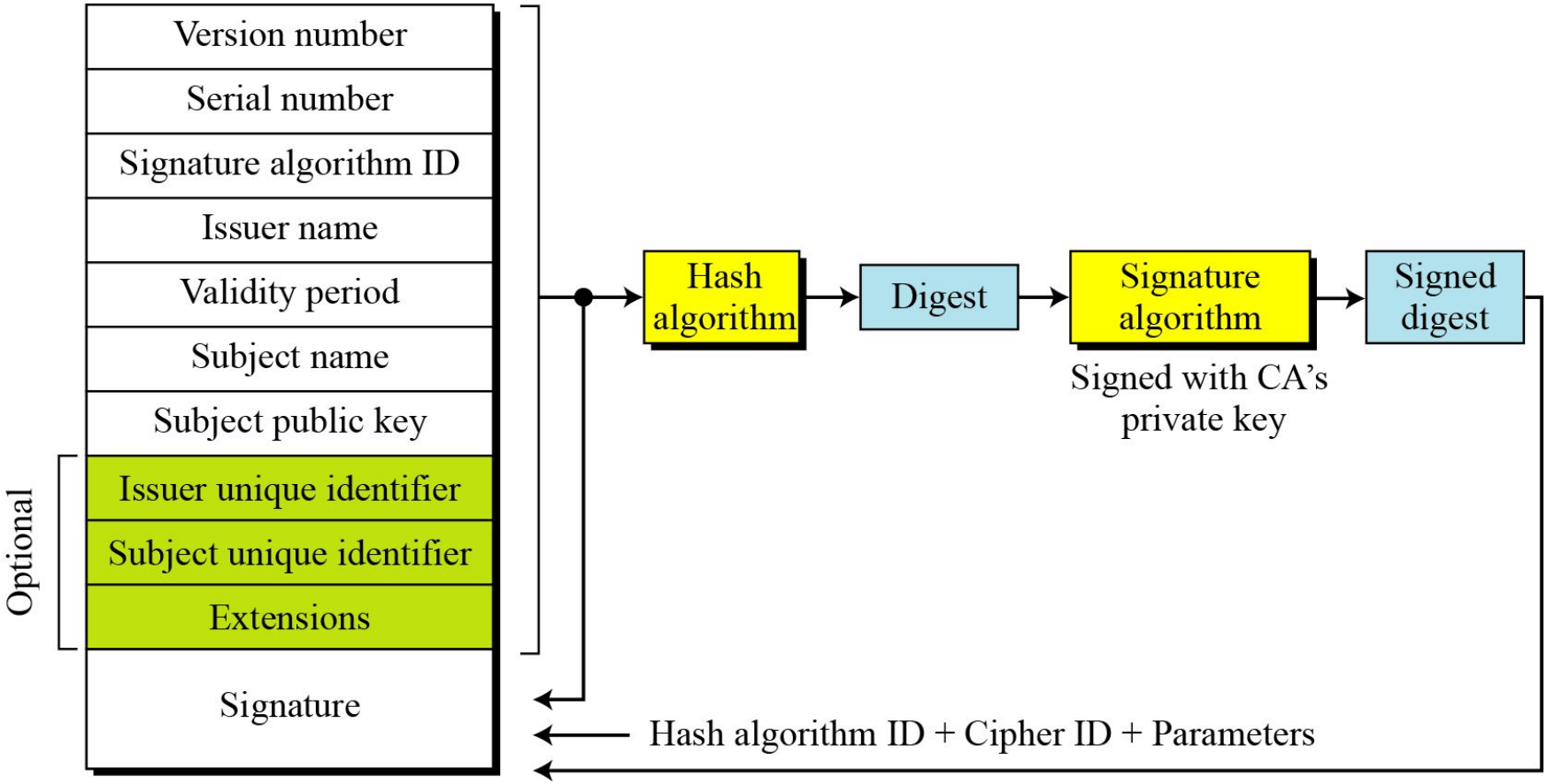
Certificate

- **Format of certificates**
  - Version (v1,v2,v3)
  - Serial number
  - Signature algorithm Identifier
  - Issuer name
  - Period of validity
  - Subject's public-key information
  - Issuer unique identifier
  - Subject unique identifier
  - Extension
  - Signature



(a) X.509 Certificate

Certificate-format



Certificate

Certificate Renewal

Each certificate has a period of validity. If there is no problem with the certificate, the CA issues a new certificate before the old one expires.

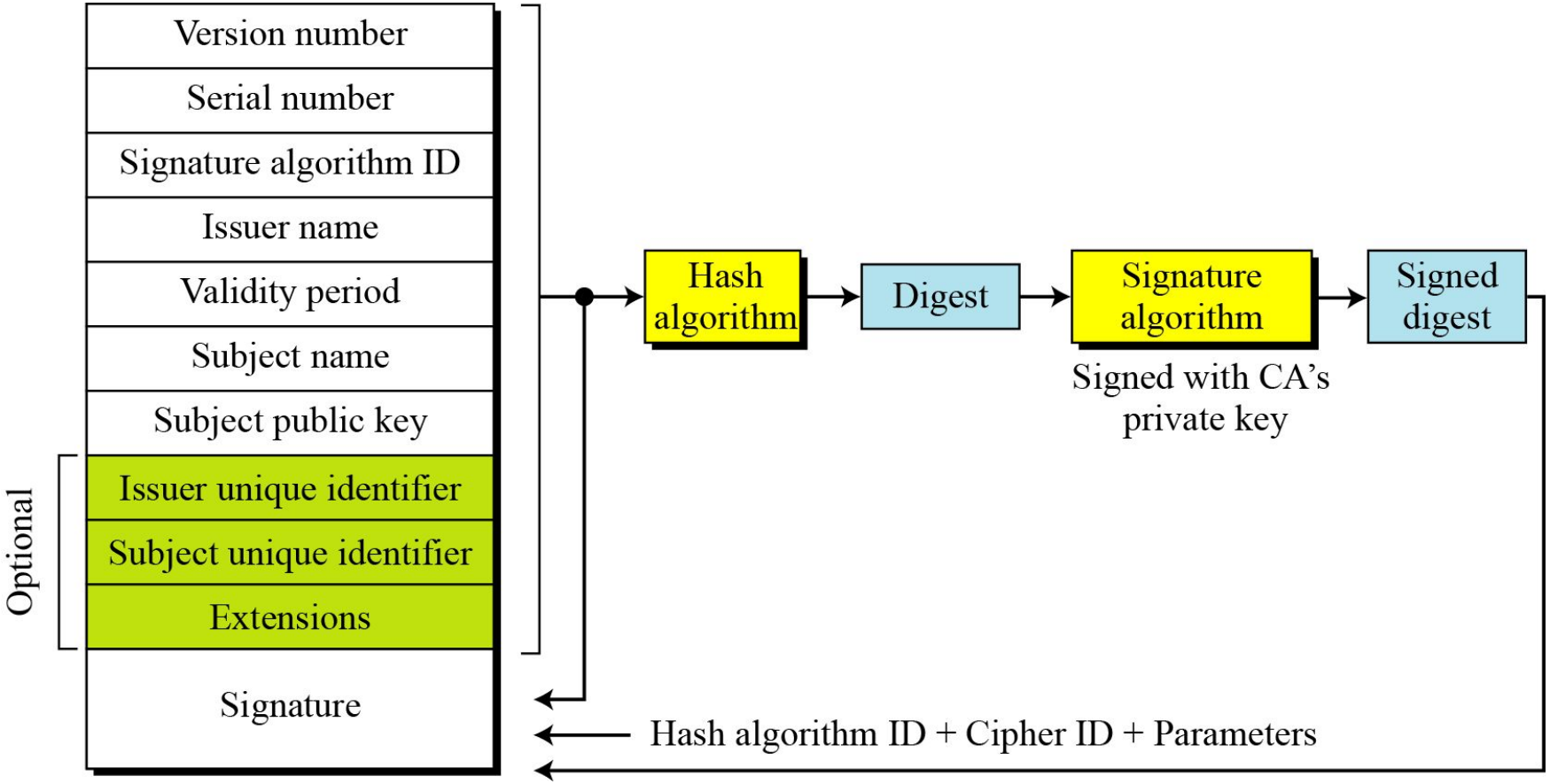
Certificate Renewal

In some cases a certificate must be revoked before its expiration.

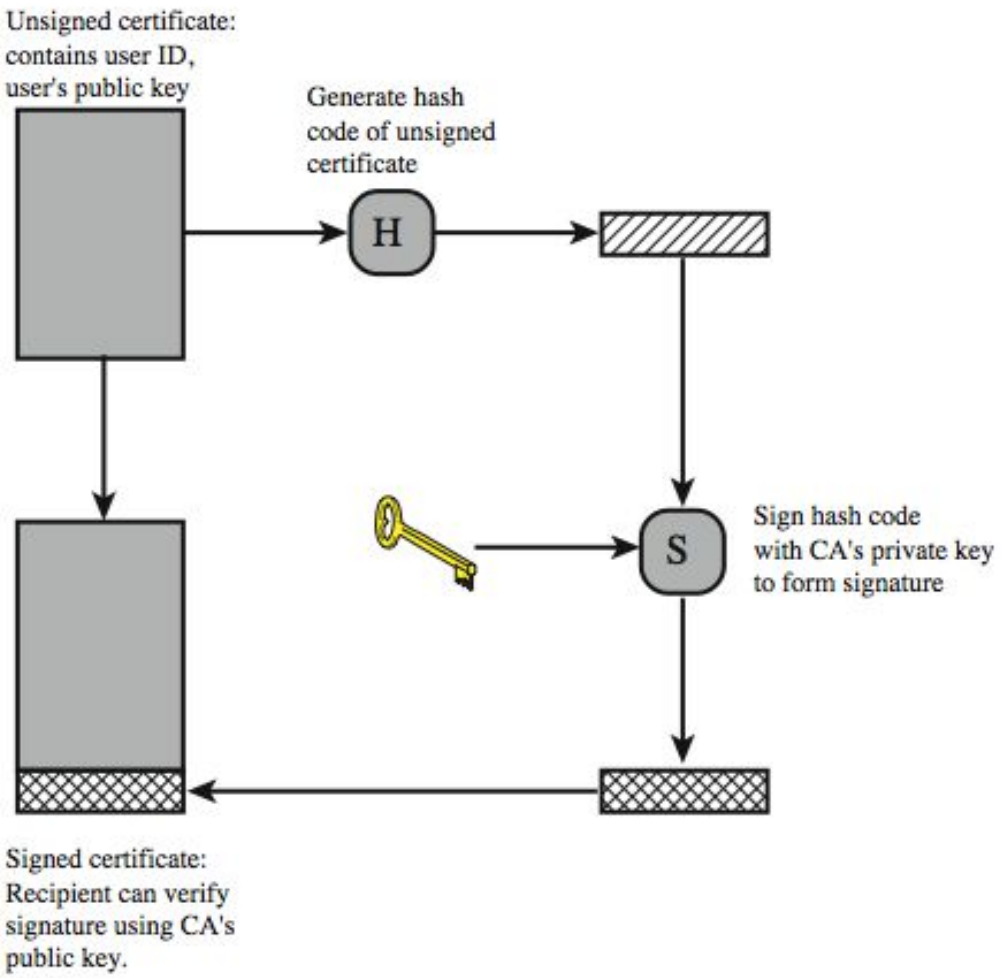
Delta Revocation

To make revocation more efficient, the delta certificate revocation list (delta CRL) has been introduced.

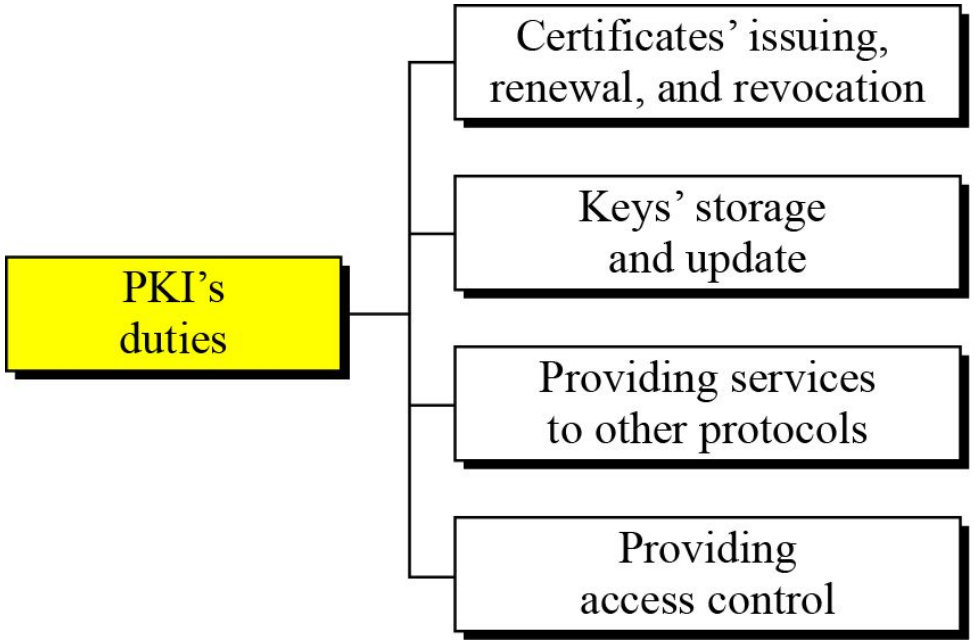
Certificate Revocation format



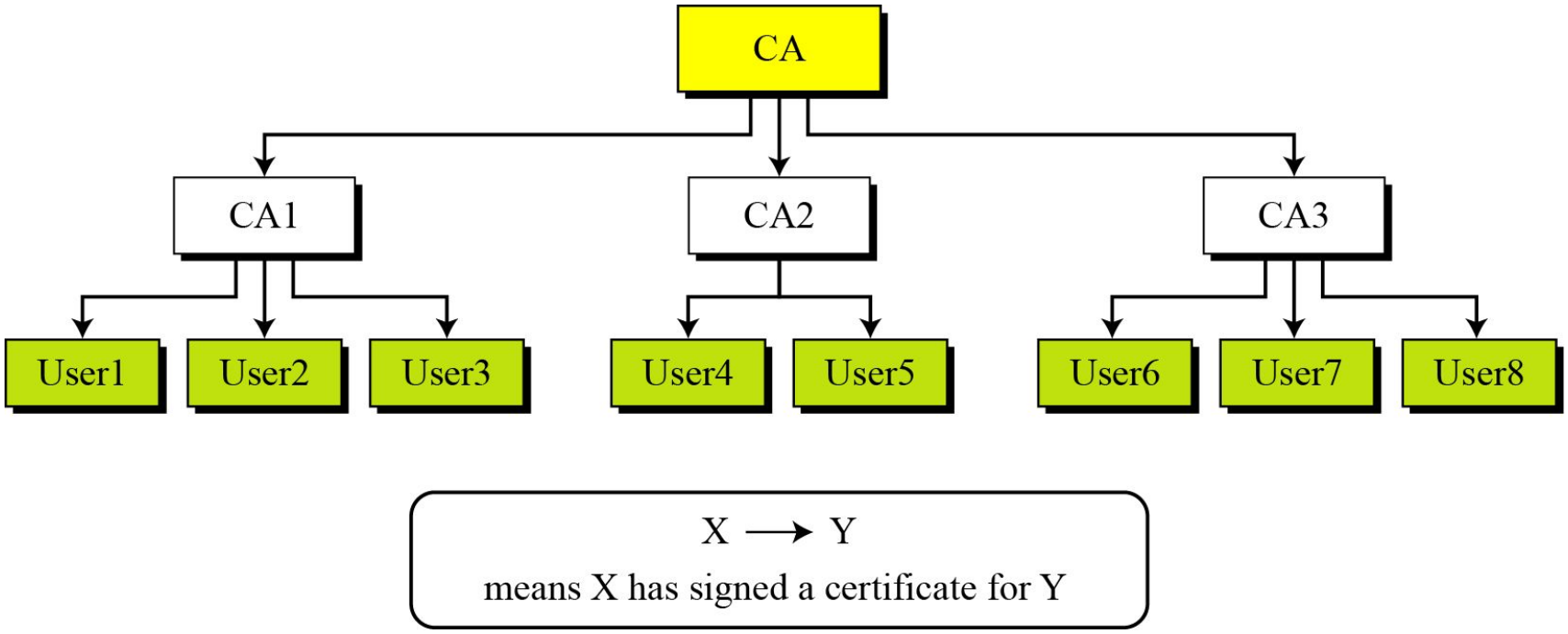
Public key Certificate



Some duties of PKI



PKI Hierarchical Model



Example

Show how User1, knowing only the public key of the CA (the root), can obtain a verified copy of User3's public key.

Solution

User3 sends a chain of certificates, CA<<CA1>> and CA1<<User3>>, to User1.

a. User1 validates CA<<CA1>> using the public key of CA.
b. User1 extracts the public key of CA1 from CA<<CA1>>.
c. User1 validates CA1<<User3>> using the public key of CA1.
d. User1 extracts the public key of User 3 from CA1<<User3>>.

Obtaining a User's Certificate

- Any user with access to the public key of the CA can recover all of the user certificate certified by the CA
- No one except CA can modify any user's certificate without detected
- If all user subscribe the same CA, directly get any certificate of user's using directory of CA.
- But single CA for large user is impractical
  - CA must sign all user certificates
  - CA's certificates must be distributed to all users
  - CA must provided his public-key to be distributed securely
- use certificates linking members of hierarchy to validate other CA's
  - each CA has certificates for clients (forward) and parent (backward)
- each client trusts parents certificates
- enable verification of any certificate from one CA by users of all other CAs in hierarchy

Discuss man in the middle attack

Generate a key using DH

Consider p=353, g= 3 (3 is primitive root of 353)

Consider p = 23 . Is 5 a generator of 23?