Public-Key Cryptography and Message Authentication

Hemant Ghayvat
Department of Computer Science and Media Technology
hemant.ghayvat@lnu.se


Game of Threats:
protect the throne

Hash Function are applied in

- Message Integrity Check (MIC)
  - send hash of message (digest)
  - MIC always encrypted, message optionally

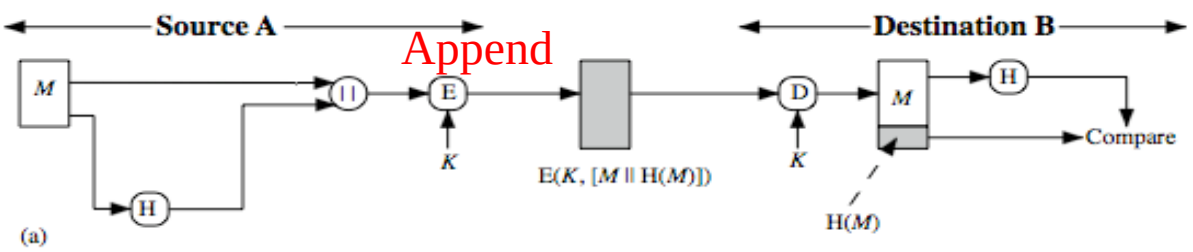
- Message Authentication Code (MAC)
  - send keyed hash of message
  - MAC, message optionally encrypted

- Digital Signature (non-repudiation)
  - Encrypt hash with private (signing) key
  - Verify with public (verification) key

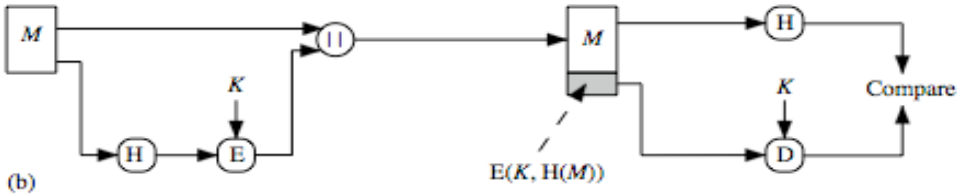
Hash Functions & Message Authentication

Symmetric Key
Unkeyed Hash

a) Message encrypted



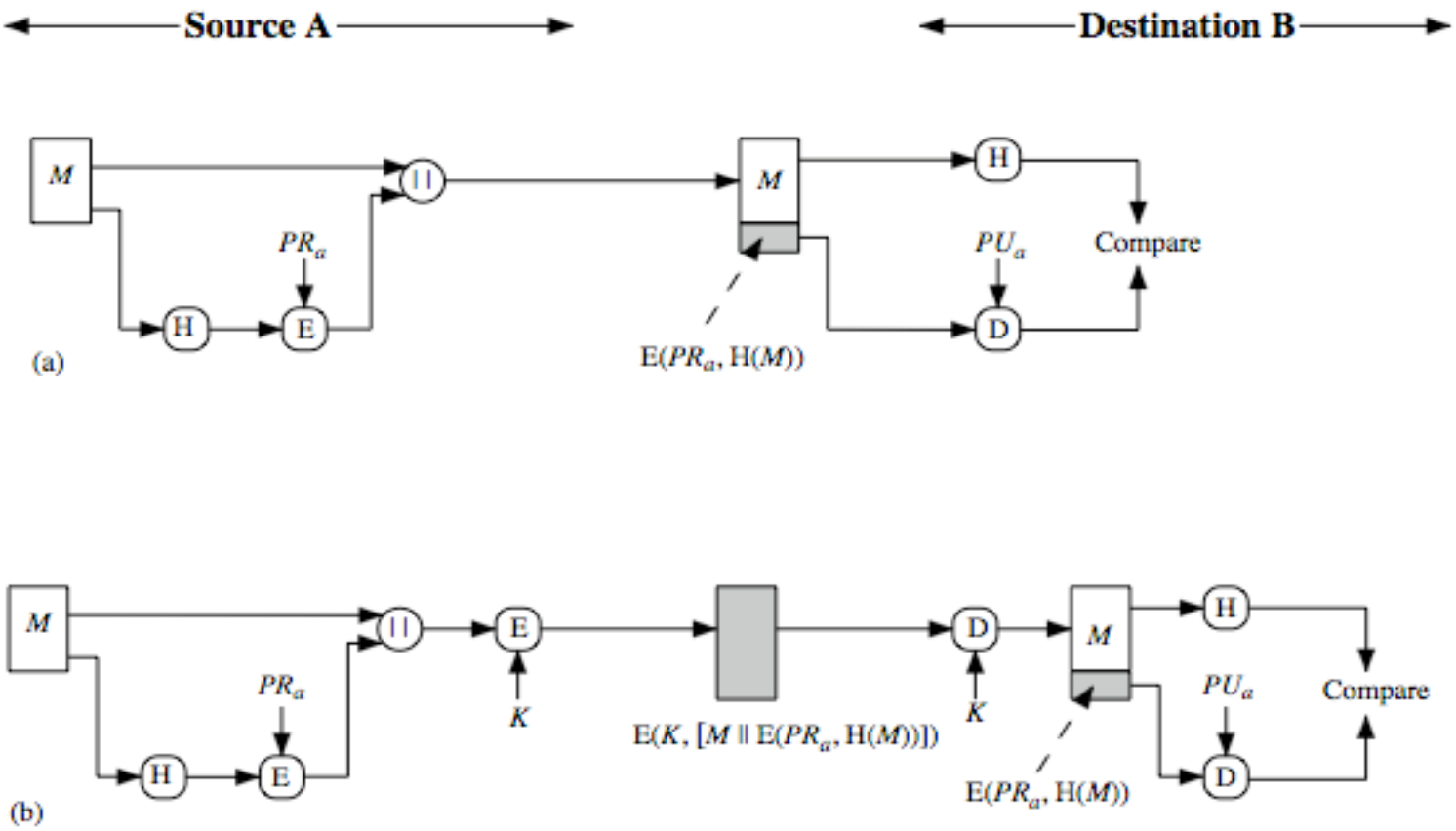
b) Message unencrypted



a) If both hash codes equal in the end, then authentication is performed, as only A and B share the secret key and msg must have come from A i.e has not been altered. Confidentiality is also there as msg was encrypted before sending

b) Only hash code is encrypted not the whole msg (it takes less time) and B can identify the sender. So only authentication, but no confidentiality as any one can read our Msg.

Hash Functions & Digital Signatures - PKPK



Message Authentication Code

Uses a shared secret key to generate (known as a cryptographic checksum or MAC) that is appended to the message

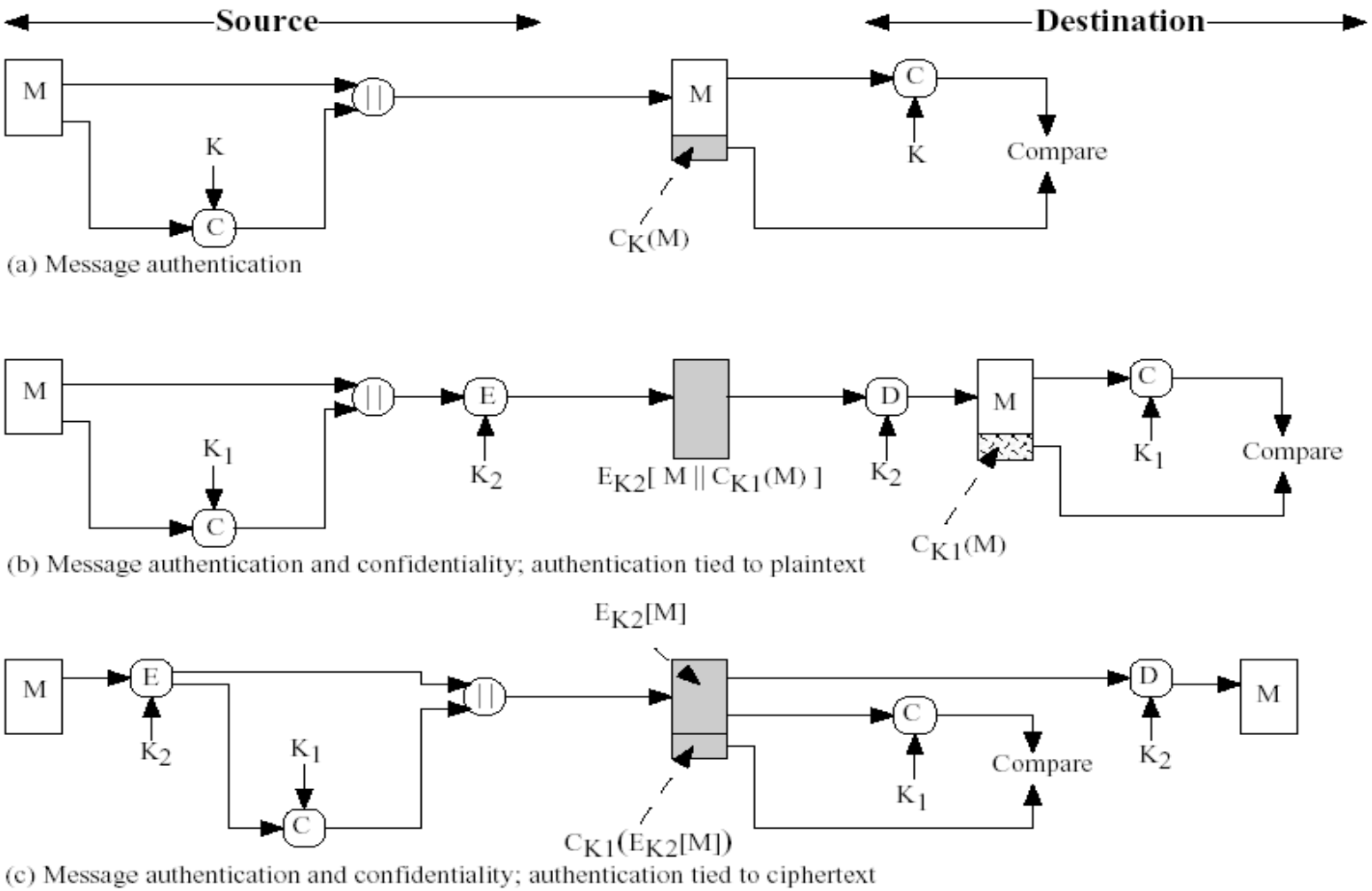
MAC = $C_K(M)$

It generate the small fixed size block of data from arbitrary length of input msg

Assurances:

- Message has not been altered
- Message is from alleged sender
- Message sequence is unaltered (requires internal sequencing)

Basic Uses of MAC



Basic Uses of MAC

(a) $A \rightarrow B: M \| C_K(M)$

- Provides authentication
  - —Only A and B share K

(b) $A \rightarrow B: E_{K_2}\left[M \| C_{K_1}(M)\right]$

- Provides authentication
  - —Only A and B share K_1
- Provides confidentiality
  - —Only A and B share K_2

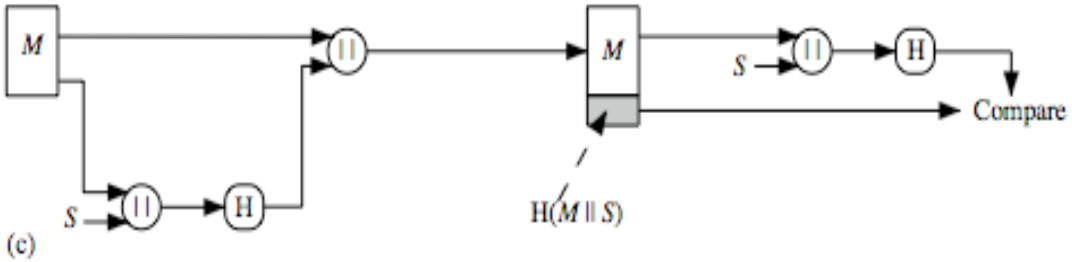
(c) $A \rightarrow B: E_{K_2}[M] \| C_{K_1}\left(E_{K_2}[M]\right)$

- Provides authentication
  - —Using K_1
- Provides confidentiality
  - —Using K_2

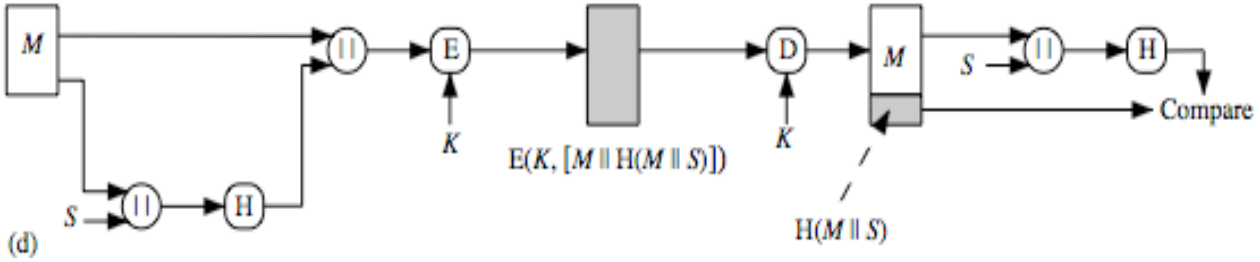
Keyed Hash Functions based on MAC

Symmetric Key
Keyed Hash

c) Message unencrypted



d) Message encrypted



Why/where Use MACs?

Digital Signatures

have looked at message authentication

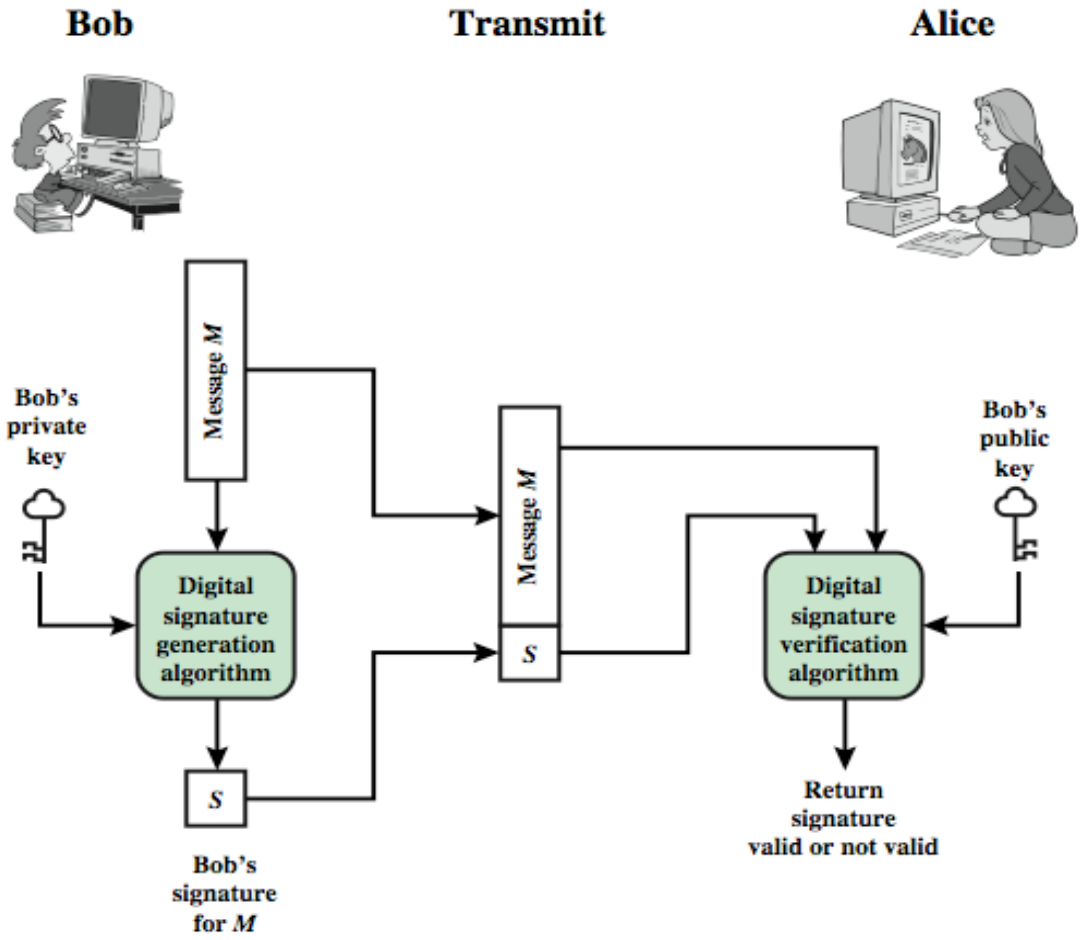
- but does not address issues of lack of trust

digital signatures provide the ability to:

- verify author, date & time of signature
- authenticate message contents
- be verified by third parties to resolve disputes

hence include authentication function with additional capabilities

Digital Signature Model



Definition

Birthday attacks are a class of brute-force techniques that target the cryptographic hash functions. The goal is to take a cryptographic hash function and find two different inputs that produce the same output.

Birthday Attacks

The probability that all 23 people have different birthdays is

$$1\times(1-\frac{1}{365})(1-\frac{2}{365})\ldots(1-\frac{22}{365})=0.493$$

Therefore, the probability of at least two having the same birthday is 1- 0.493=0.507

More generally, suppose we have N objects, where N is large. There are r people, and each chooses an object. Then

$$P(\text{there is a match}) \approx 1-e^{-r^2/2N}$$

Birthday Attacks

(Example) We have 40 license plates, each ending in a 3-digit number. What is the probability that two of the license plates end in the same 3 digits?

(Solution) N=1000, r=40

1. Approximation: $1-(1-\frac{1}{1000})(1-\frac{2}{1000})...(1-\frac{39}{1000})=0.546$

2. The exact answer:

$$1-e^{-40^2/2\cdot 1000}=0.551$$

Attack Prevention

The important property is the length in bits of the message digest produced by the hash function.

If the number of *m* bit hash , the cardinality *n (number of elements in the hash set)* of the hash function is

$$n = 2^m$$

The 0.5 probability of collision for m bit hash, expected number of operation *k* before finding a collision is very close to

$$k \approx \sqrt{n} = 2^{m/2}$$

m should be large enough so that it's not feasible to compute hash values!!!

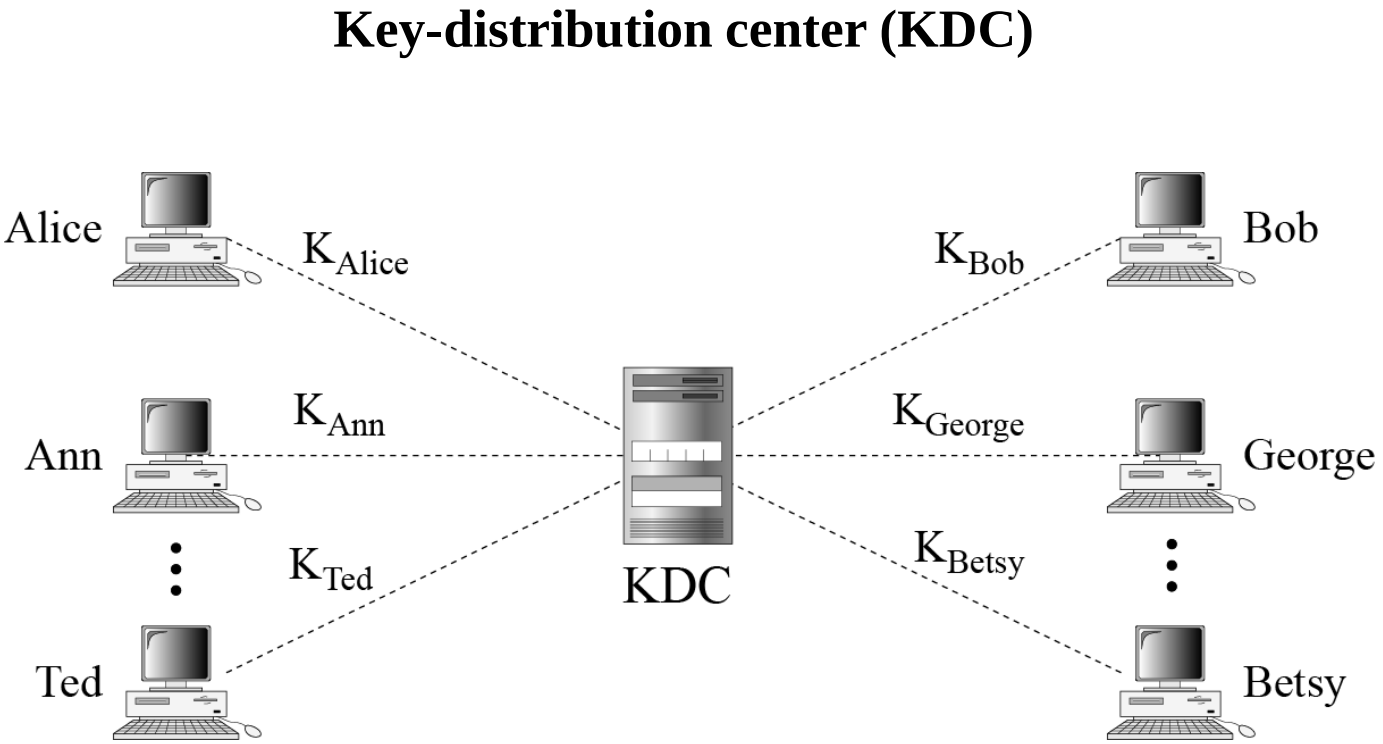
Key Distribution and Management

SYMMETRIC-KEY DISTRIBUTION

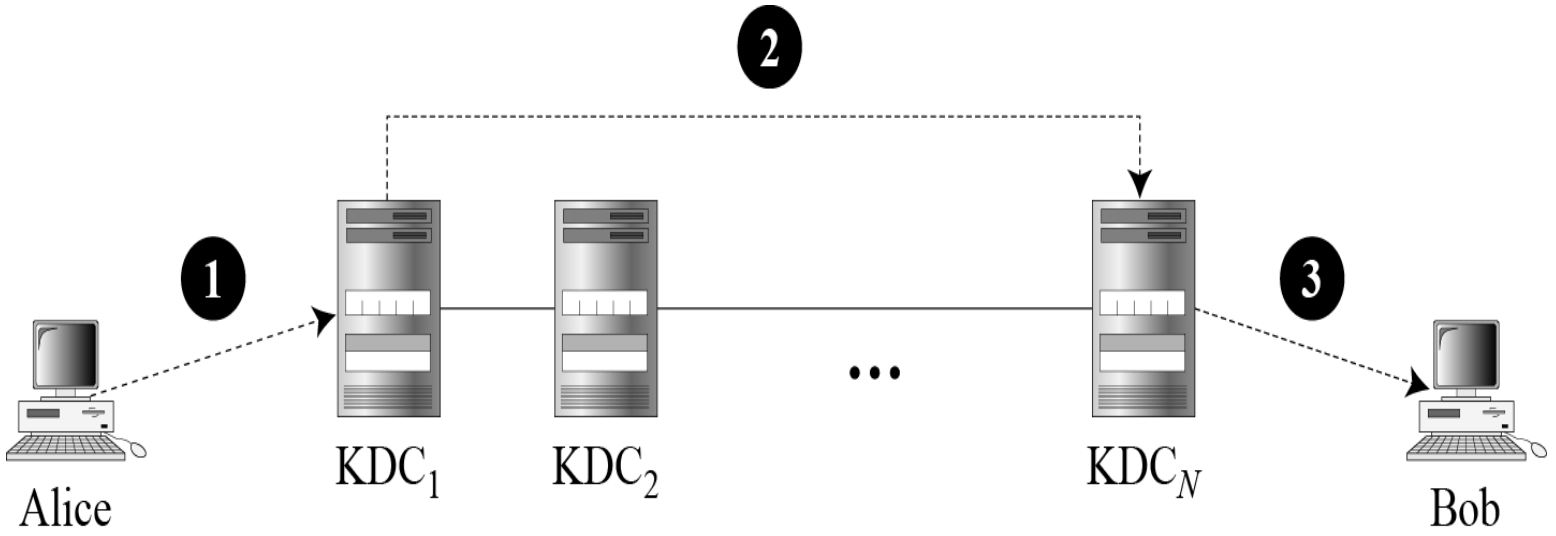
Symmetric-key cryptography is more efficient than asymmetric-key cryptography for enciphering large messages. Symmetric-key cryptography, however, needs a shared secret key between two parties. The distribution of keys is another problem.


CAN YOU REALLY
KEEP A SECRET...
imgflip.com

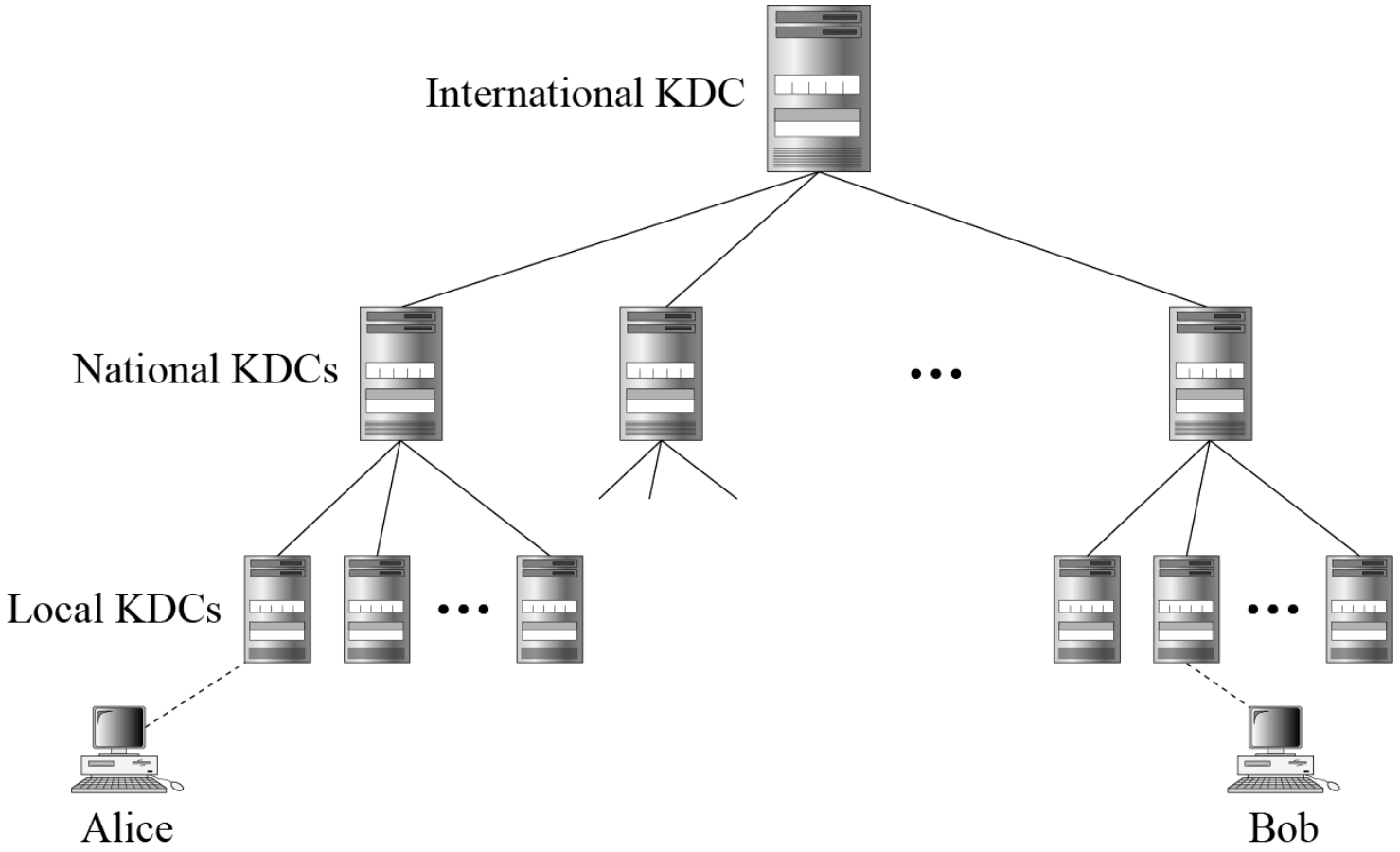
Key-Distribution Center: KDC



Flat Multiple KDCs.



Hierarchical Multiple KDCs

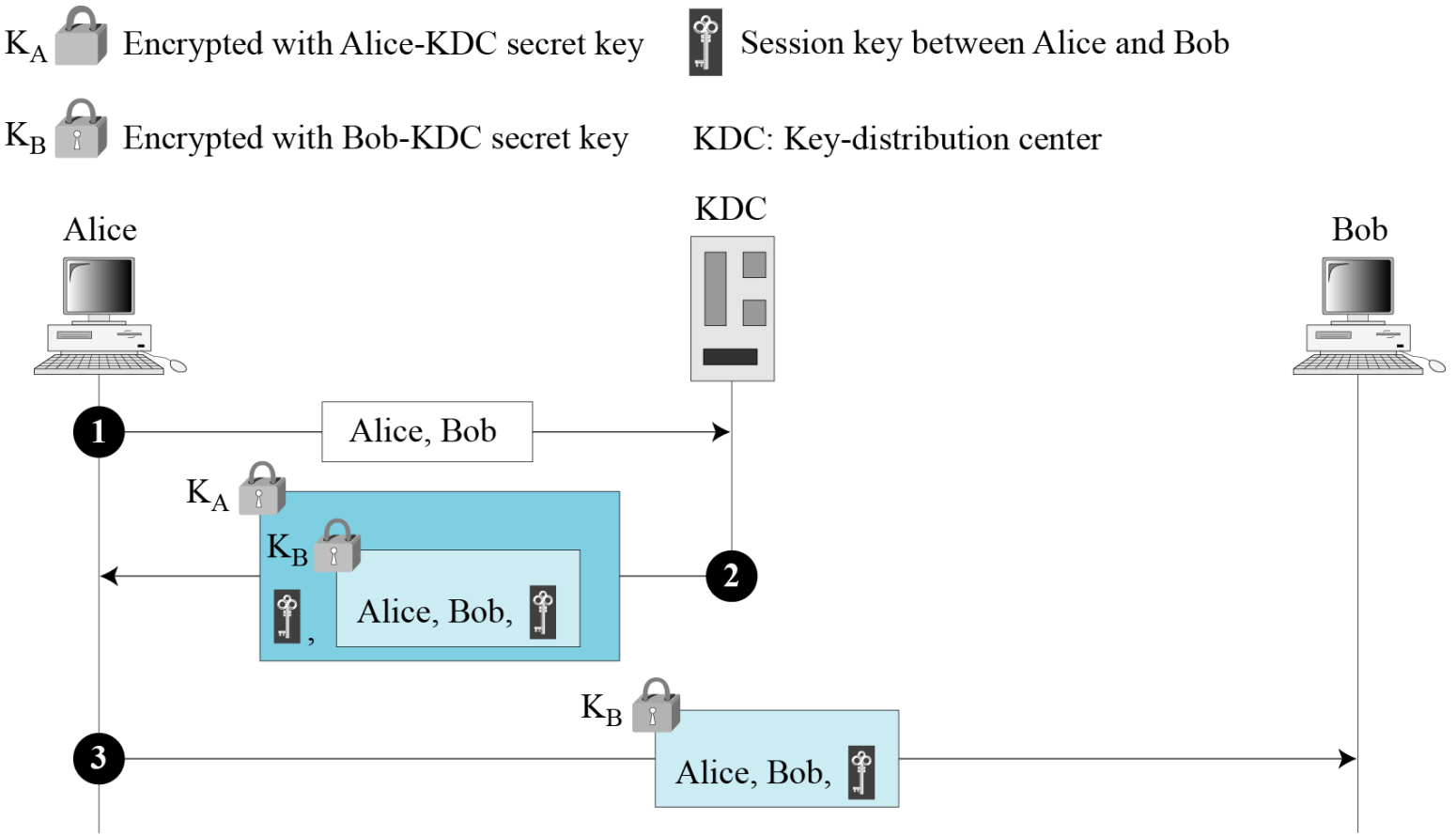


Session Keys

A KDC creates a secret key for each member. This secret key can be used only between the member and the KDC, not between two members.

A session symmetric key between two parties is used only once

A Simple Protocol Using a KDC

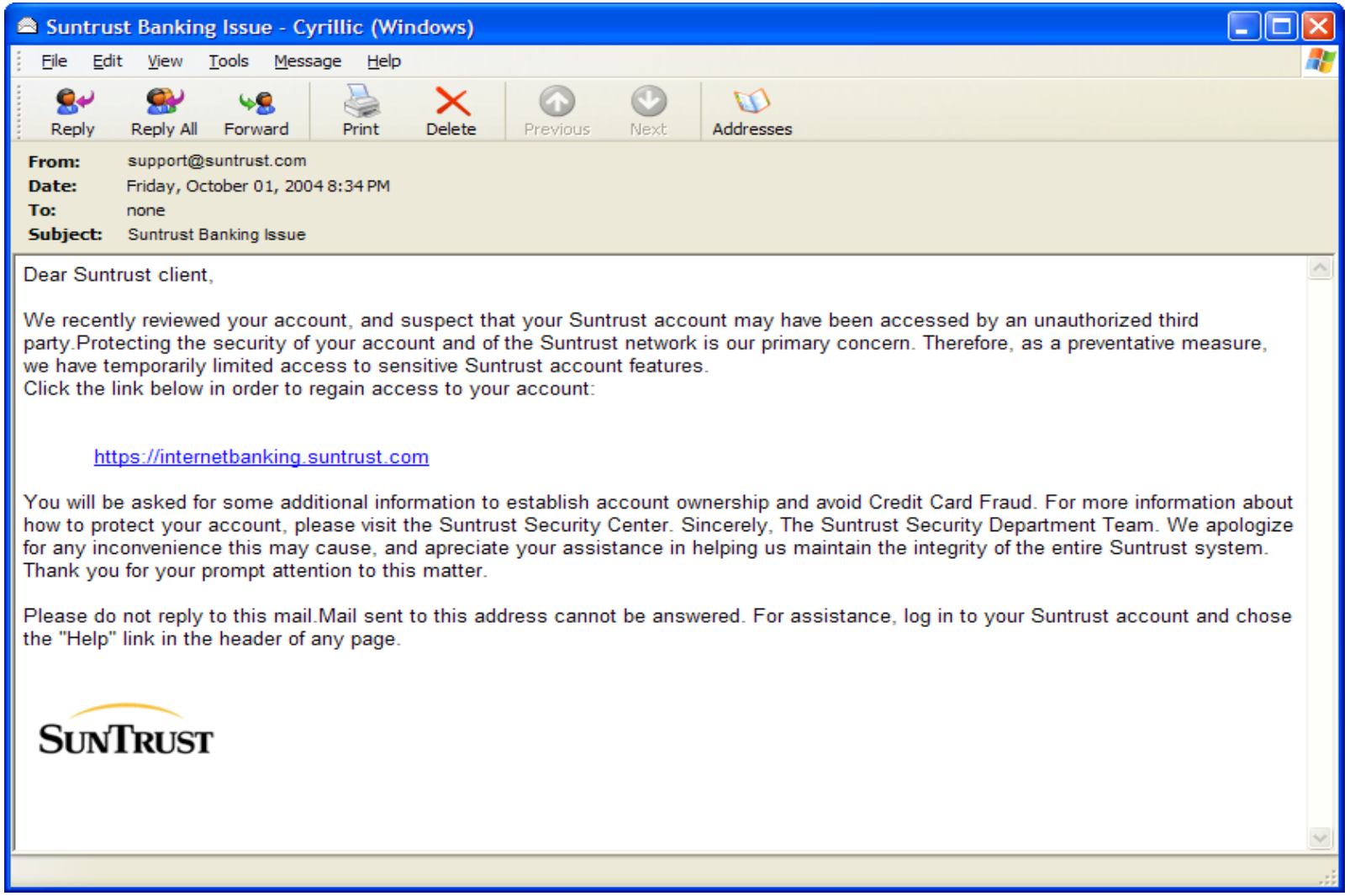



FINALLY
3% MILK

KERBEROS

Kerberos is an authentication protocol, and at the same time a KDC, that has become very popular. Several systems, including Windows 2000, use Kerberos. Originally designed at MIT, it has gone through several versions.

Motivation


Suntrust Banking Issue - Cyrillic (Windows)
File Edit View Tools Message Help
Reply Reply All Forward Print Delete Previous Next Addresses
From: support@suntrust.com
Date: Friday, October 01, 2004 8:34 PM
To: none
Subject: Suntrust Banking Issue
Dear Suntrust client,
We recently reviewed your account, and suspect that your Suntrust account may have been accessed by an unauthorized third party.Protecting the security of your account and of the Suntrust network is our primary concern. Therefore, as a preventative measure, we have temporarily limited access to sensitive Suntrust account features.
Click the link below in order to regain access to your account:
https://internetbanking.suntrust.com
You will be asked for some additional information to establish account ownership and avoid Credit Card Fraud. For more information about how to protect your account, please visit the Suntrust Security Center. Sincerely, The Suntrust Security Department Team. We apologize for any inconvenience this may cause, and apreciate your assistance in helping us maintain the integrity of the entire Suntrust system. Thank you for your prompt attention to this matter.
Please do not reply to this mail.Mail sent to this address cannot be answered. For assistance, log in to your Suntrust account and chose the "Help" link in the header of any page.
SUNTRUST

Kerberos

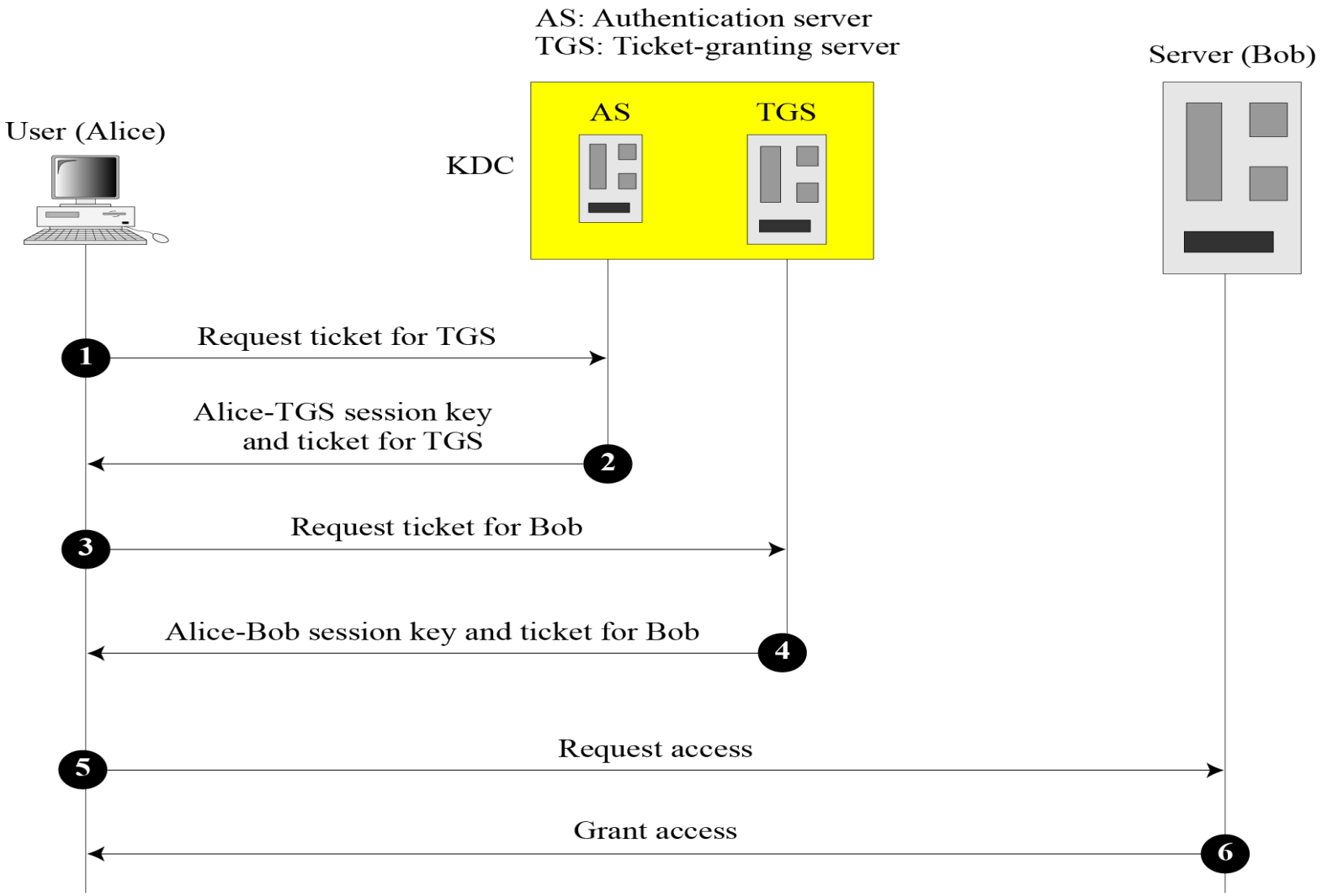
A practical authentication service

Kerberos: three headed dog in Greek mythology, the guardian of the entrance of Hades

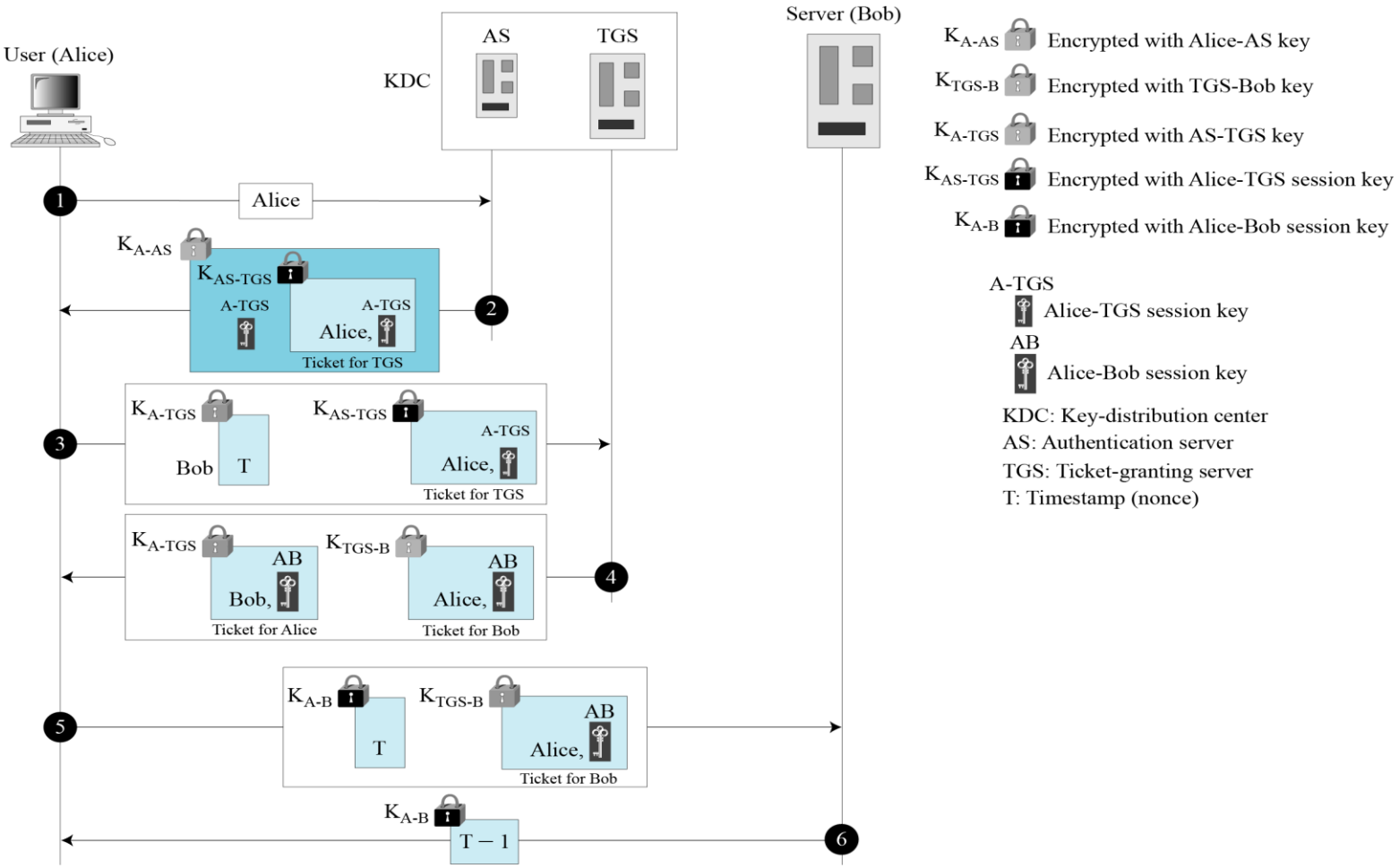
Those three heads in security: AAA (Authentication, Accounting, Audit)

However, in Kerberos the last two heads never implemented

Kerberos servers



Kerberos example



A Simple Authentication Dialogue without TGT

where

C= client Alice , AS= authentication server ,V= Bob server

ID_C= identifier of user on C ,ID_V= identifier of V Bob

P_C= password of user on C Bob, AD_C= network address of C Alice

K_v= secret encryption key shared by AS and V Bob

the user logs on to a workstation and requests access to server V.

The client module C in the Alice user's workstation requests the user's password and then sends a message to the AS that includes the Alice user's ID, the Bob server's ID, and the user's password.

The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V Bob.

A Simple Authentication Dialogue

- an authentication server (AS)
  - that knows the passwords of all users and stores these in a centralized database.
  - shares a unique secret key with each server
  - These keys have been distributed physically or in some other secure manner.

(1) C $\longrightarrow$ AS: $ID_C||P_C||ID_V$
(2) AS $\longrightarrow$ C: Ticket
(3) C $\longrightarrow$ V: $ID_C||$Ticket

- Ticket = $E(K_v, [ID_C||AD_C||ID_V])$

A Simple Authentication Dialogue

- The AS creates a ticket that contains the user's ID Alice and network address (where Alice and Bob are located) and the server's ID Bob.
- This ticket is encrypted using the secret key shared by the AS and this server Bob.
- This ticket is then sent back to C Alice.
- C Alice sends a message to V Bob containing C's ID and the ticket.
- Bob V decrypts the ticket and verifies that the user ID Alice in the ticket is the same as the unencrypted user ID in the message.
- There are two ID’s one which verified by the AS encrypted with secret key and other which is coming with plaintext of Alice

A More Secure Authentication Dialogue

Two problems in previous dialogue :

a. each ticket can be used only once.

b. plaintext transmission of the password [message (1)].

a scheme for avoiding plaintext passwords and a new server, known as the ticket-granting server (TGS).

A More Secure Authentication Dialogue

Once per user logon session:

(1) C ⟶ AS: $ID_C||ID_{tgs}$

(2) AS ⟶ C: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) C ⟶ TGS: $ID_C||ID_V||Ticket_{tgs}$

(4) TGS ⟶ C: $Ticket_v$

Once per service session:

(5) C ⟶ V: $ID_C||Ticket_v$

$Ticket_{tgs} = E(K_{tgs}, [ID_C||AD_C||ID_{tgs}||TS1||Lifetime1])$

$Ticket_v = E(K_v, [ID_C||AD_C||ID_v||TS2||Lifetime2])$

A More Secure Authentication Dialogue

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID and password to the AS, together with the TGS ID, indicating a request to use the TGS service.

2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password. When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered.

Now that the client has a ticket-granting ticket, access to any server can be obtained with steps 3 and 4:

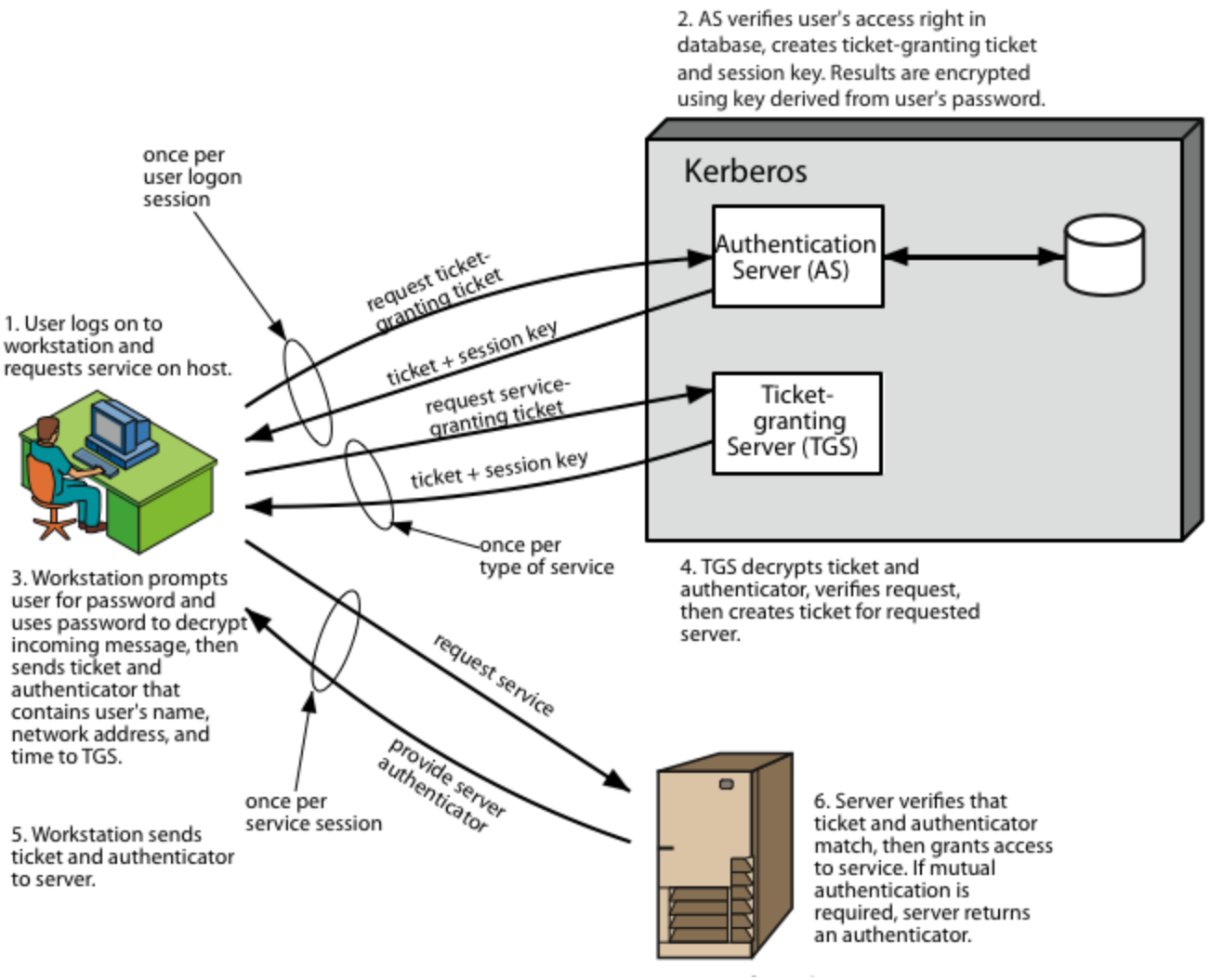
A More Secure Authentication Dialogue

3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.

4. The TGS decrypts the incoming ticket and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.

5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service-granting ticket. The server authenticates by using the contents of the ticket.

Kerberos 4 Overview



The Version 4 Authentication Dialogue

Problems:

remain in previous dialogue the lifetime associated with the ticket-granting ticket deny the true service to the user.

Solution:

1. A network service (the TGS or an application service) must be able to prove that the person using a ticket is the same person to whom that ticket was issued.
2. servers authenticate themselves to users ($Authenticator_c$)

Kerberos 4 Overview

a basic third-party authentication scheme

have an Authentication Server (AS)

- users initially negotiate with AS to identify self
- AS provides a non-corruptible authentication credential (ticket granting ticket TGT)

have a Ticket Granting server (TGS)

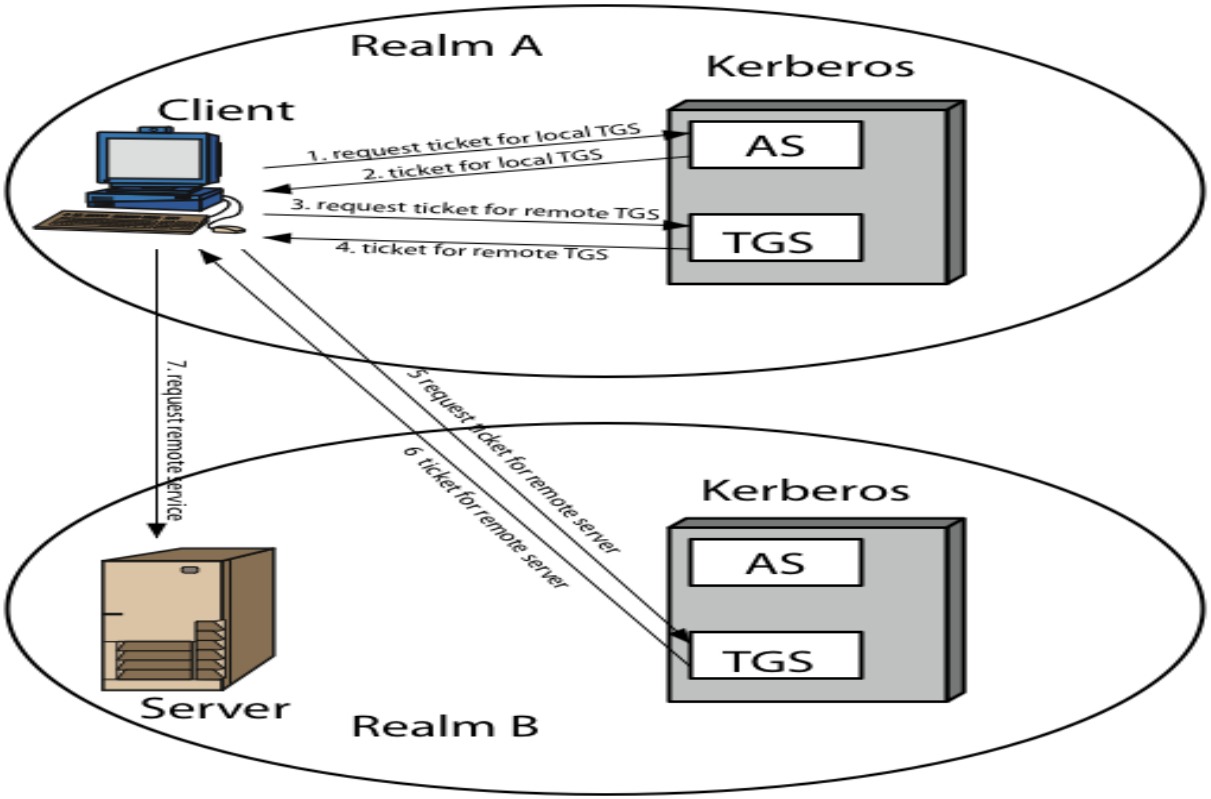
- users subsequently request access to other services from TGS on basis of users TGT

The Version 4 Authentication Dialogue

(1) C $\longrightarrow$ AS $ID_c||ID_{tgs}||TS_1$
(2) AS $\longrightarrow$ C $E(K_c,[K_{c,tgs}||ID_{tgs}||TS_2||Lifetime_2||Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs}||ID_c||AD_c||ID_{tgs}||TS_2||Lifetime_2])$
(a) Authentication Service Exchange to obtain ticket-granting ticket
(3) C $\longrightarrow$ TGS $ID_v||Ticket_{tgs}||Authenticator_c$
(4) TGS $\longrightarrow$ C $E(K_{c,tgs}, [K_{c,v}||ID_v||TS_4||Ticket_v])$
$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs}||ID_C||AD_C||ID_{tgs}||TS_2||Lifetime_2])$
$Ticket_v = E(K_v, [K_{c,v}||ID_C||AD_C||ID_v||TS_4||Lifetime_4])$
$Authenticator_c = E(K_{c,tgs}, [ID_C||AD_C||TS_3])$
(b) Ticket-Granting Service Exchange to obtain service-granting ticket
(5) C $\longrightarrow$ V $Ticket_v||Authenticator_c$
(6) V $\longrightarrow$ C $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)
$Ticket_v = E(K_v, [K_{c,v}||ID_c||AD_c||ID_v||TS_4||Lifetime_4])$
$Authenticator_c = E(K_{c,v},[ID_c||AD_C||TS_5])$
(c) Client/Server Authentication Exchange to obtain service

Kerberos Realms



Kerberos Version 5

developed in mid 1990's

specified as Internet standard RFC 1510

provides improvements over v4

- addresses environmental shortcomings
  - encryption alg, network protocol, byte order, ticket lifetime, authentication forwarding, interrealm auth
- and technical deficiencies
  - double encryption, non-std mode of use, session keys, password attacks

Kerberos Version 5

The minor differences between version 4 and version 5 are briefly listed below:

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.

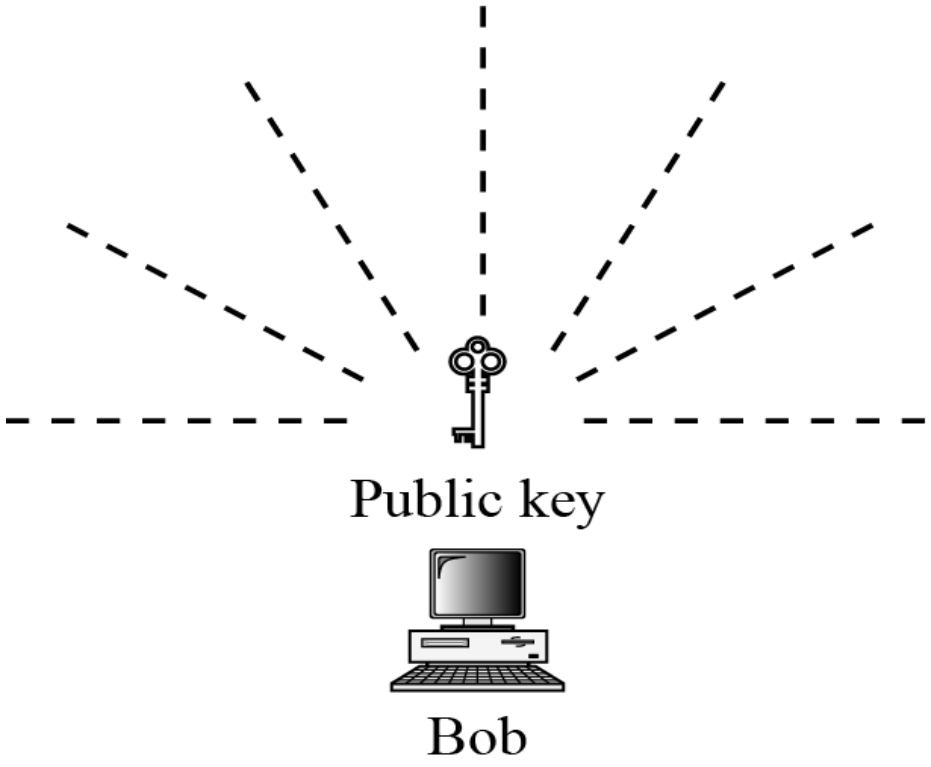

MAN IN THE MIDDLE ATTACK

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.

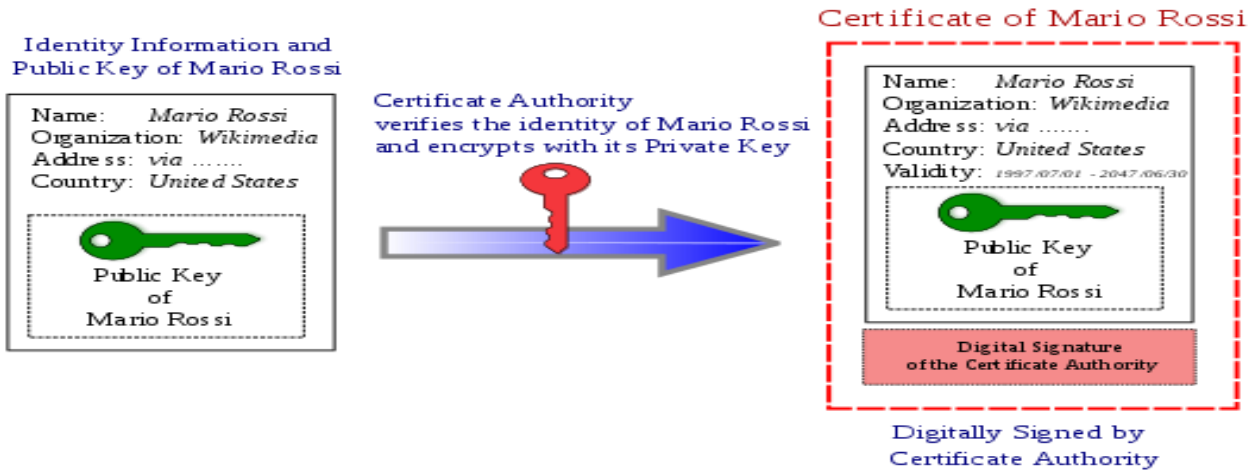
Public Announcement

Announcing a public key



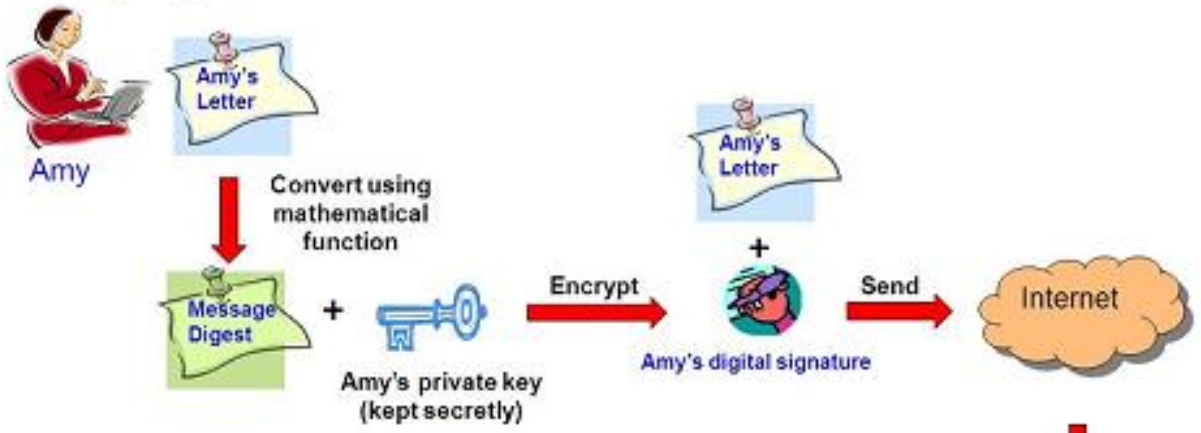
Certification Authority

- Certification authority generates the signature (using the sender's private key in the encryption process)
- This signature would be generated for the user public key and user identity

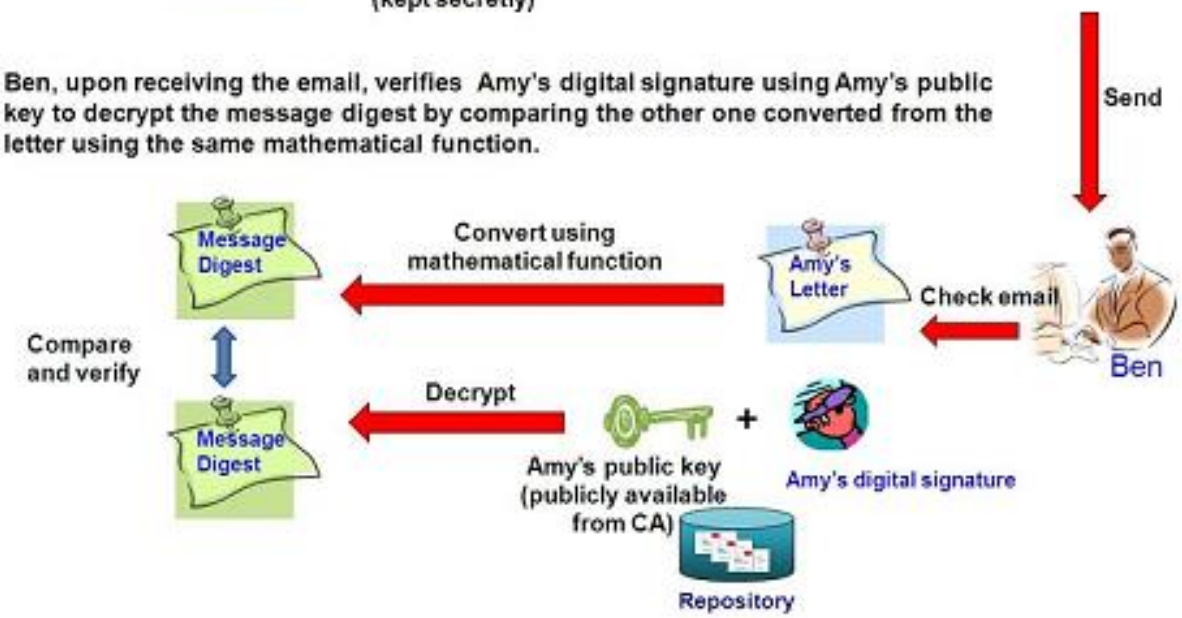


Digital Signature

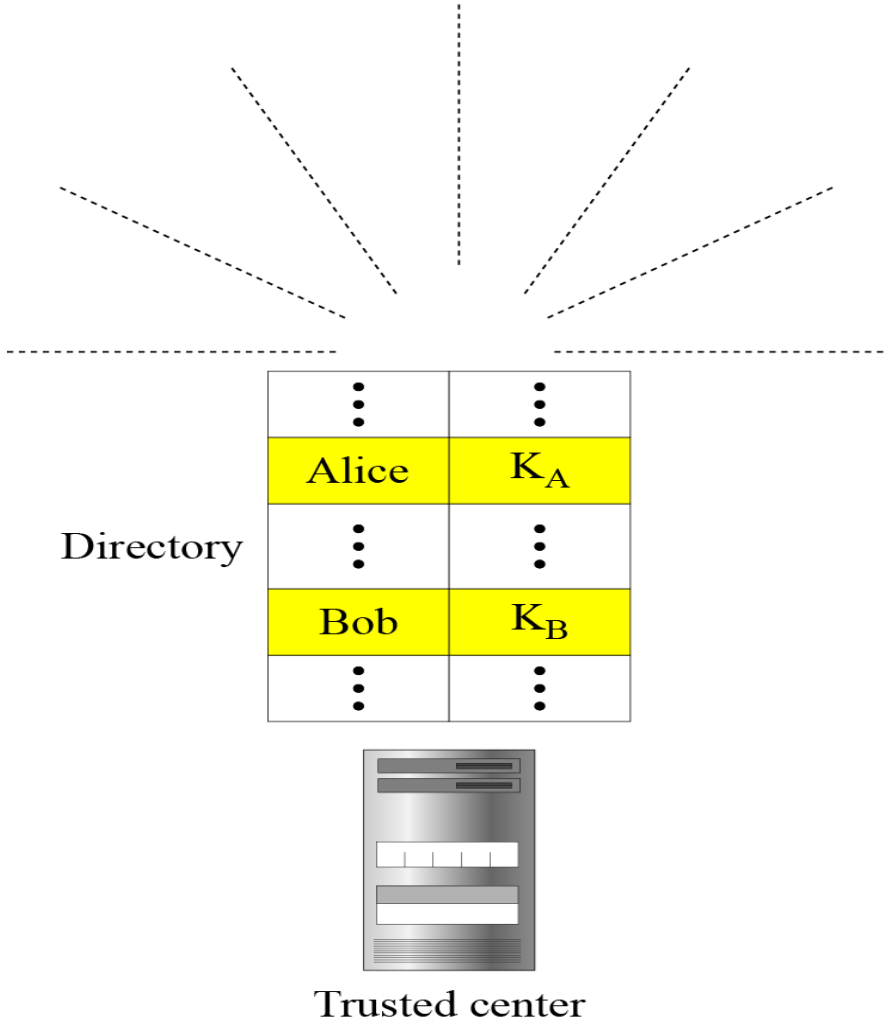
1. Amy converts her letter into a message digest by using a mathematical function. She then creates her digital signature by encrypting the message digest using her private key. Her letter, together with her digital signature are sent to Ben via email.



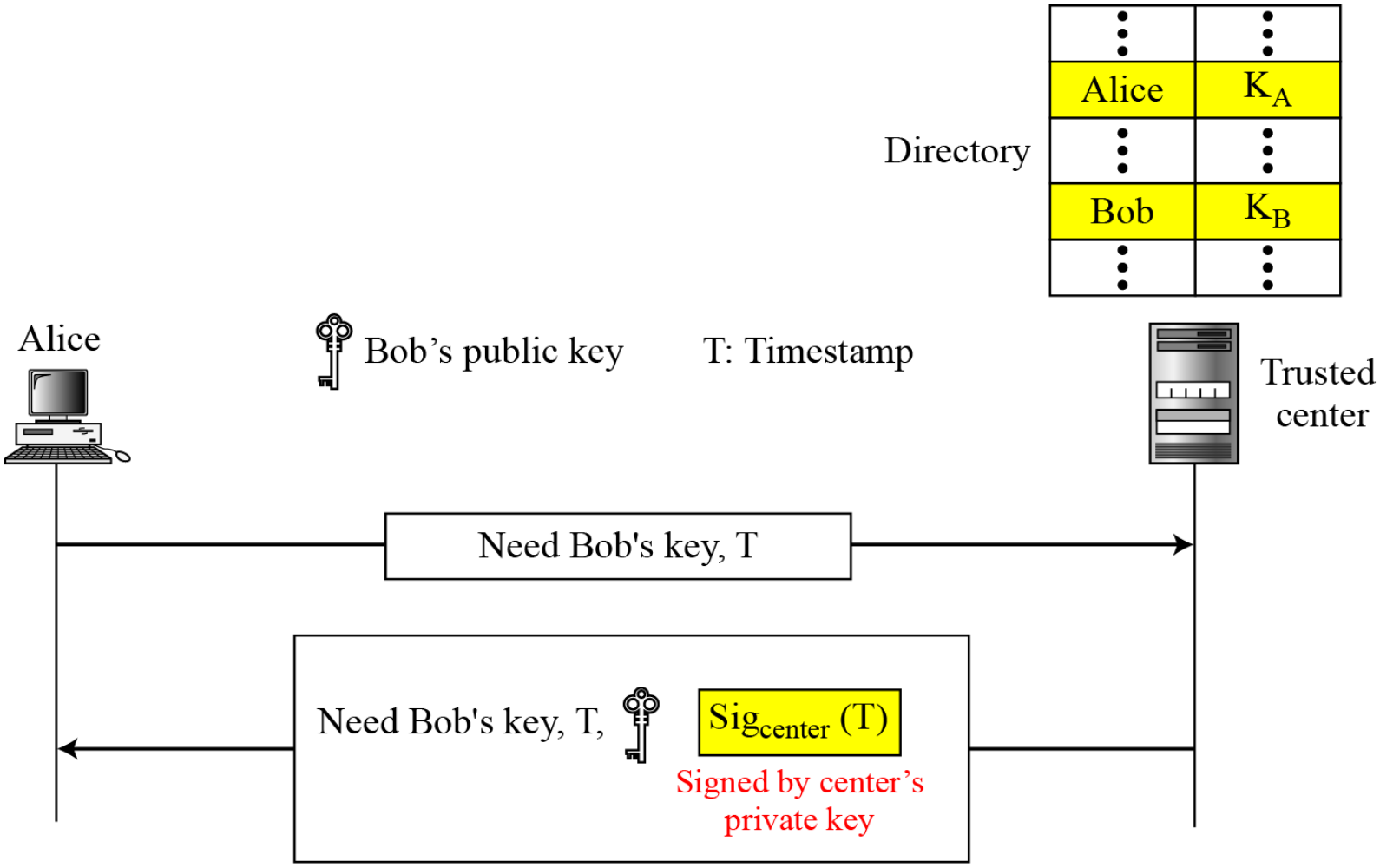
2. Ben, upon receiving the email, verifies Amy's digital signature using Amy's public key to decrypt the message digest by comparing the other one converted from the letter using the same mathematical function.



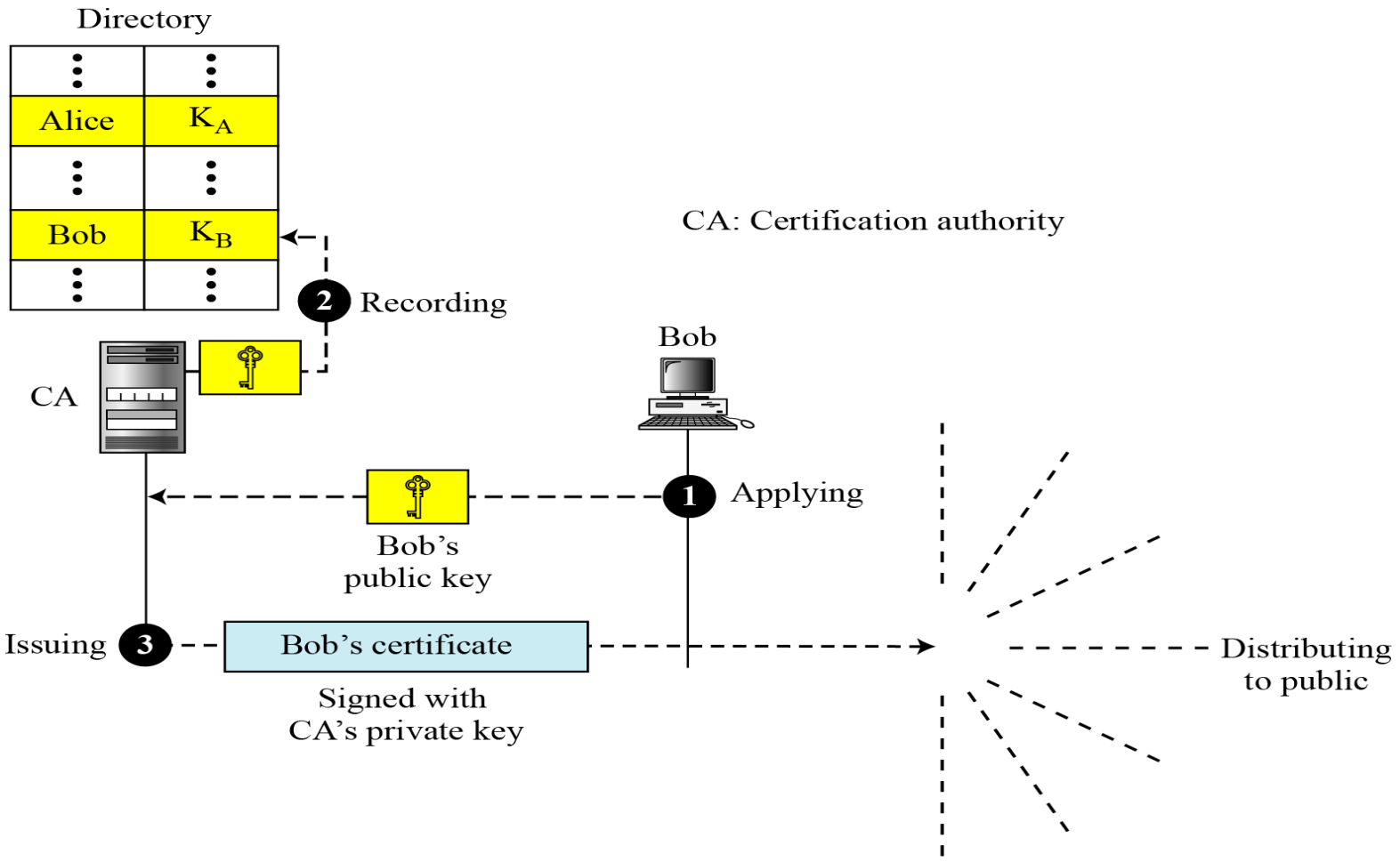
Trusted Center



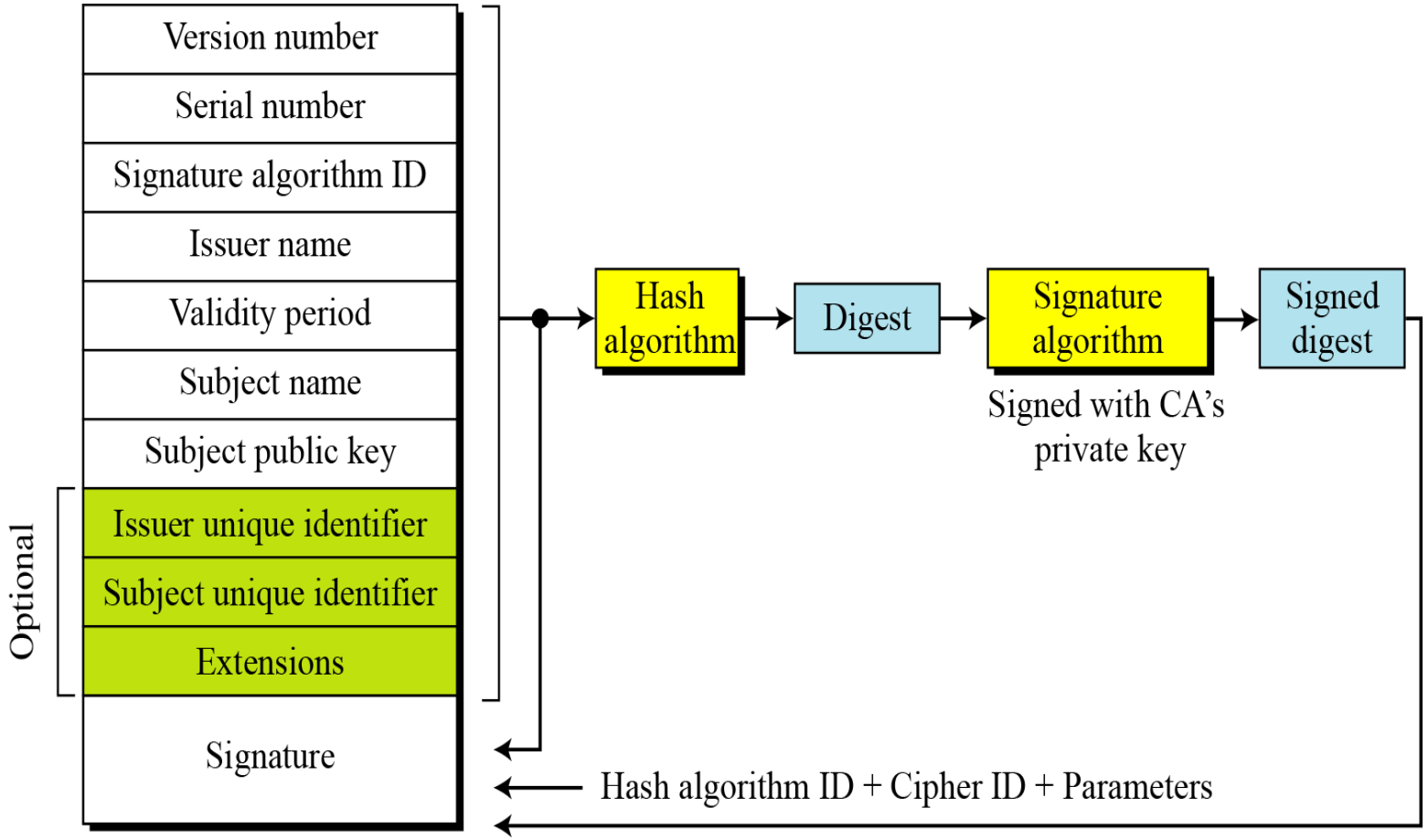
Controlled Trusted Center



Certification Authority



X.509-format of a Certificate



X.509 Authentication Service

part of CCITT X.500 directory service standards

- distributed servers maintaining some info database

defines framework for authentication services

- directory may store public-key certificates
- with public key of user
- signed by certification authority

also defines authentication protocols

uses public-key crypto & digital signatures

- algorithms not standardised, but RSA recommended

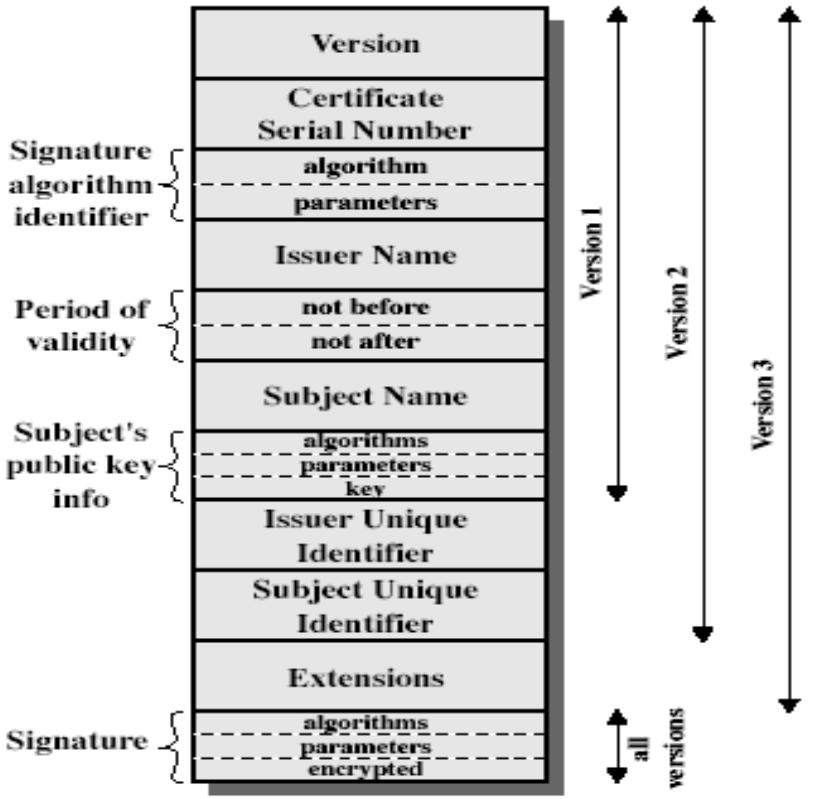
X.509 Certificates

issued by a Certification Authority (CA), containing:

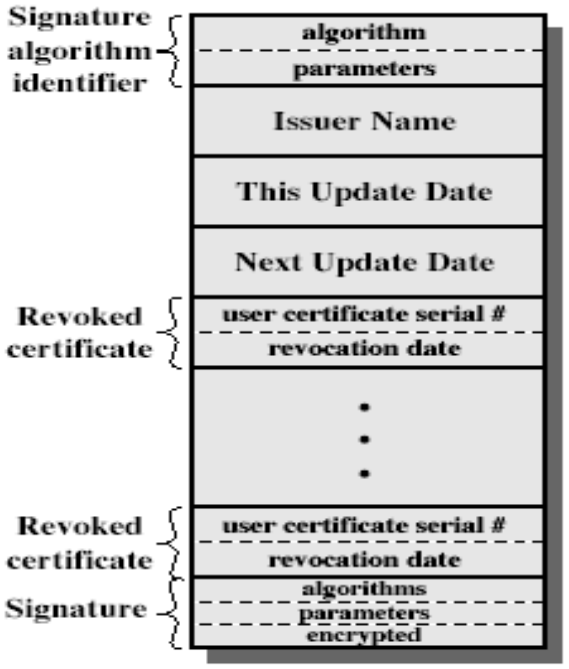
- version (1, 2, or 3)
- serial number (unique within CA) identifying certificate
- signature algorithm identifier
- issuer X.500 name (CA)
- period of validity (from - to dates)
- subject X.500 name (name of owner)
- subject public-key info (algorithm, parameters, key)
- issuer unique identifier (v2+)
- subject unique identifier (v2+)
- extension fields (v3)
- signature (of hash of all fields in certificate)

notation `CA<<A>>` denotes certificate for A signed by CA

X.509 Certificates



(a) X.509 Certificate



(b) Certificate Revocation List

Obtaining a Certificate

any user with access to CA can get any certificate from it

only the CA can modify a certificate

because cannot be forged, certificates can be placed in a public directory

CA Hierarchy

if both users share a common CA then they are assumed to know its public key

otherwise CA's must form a hierarchy

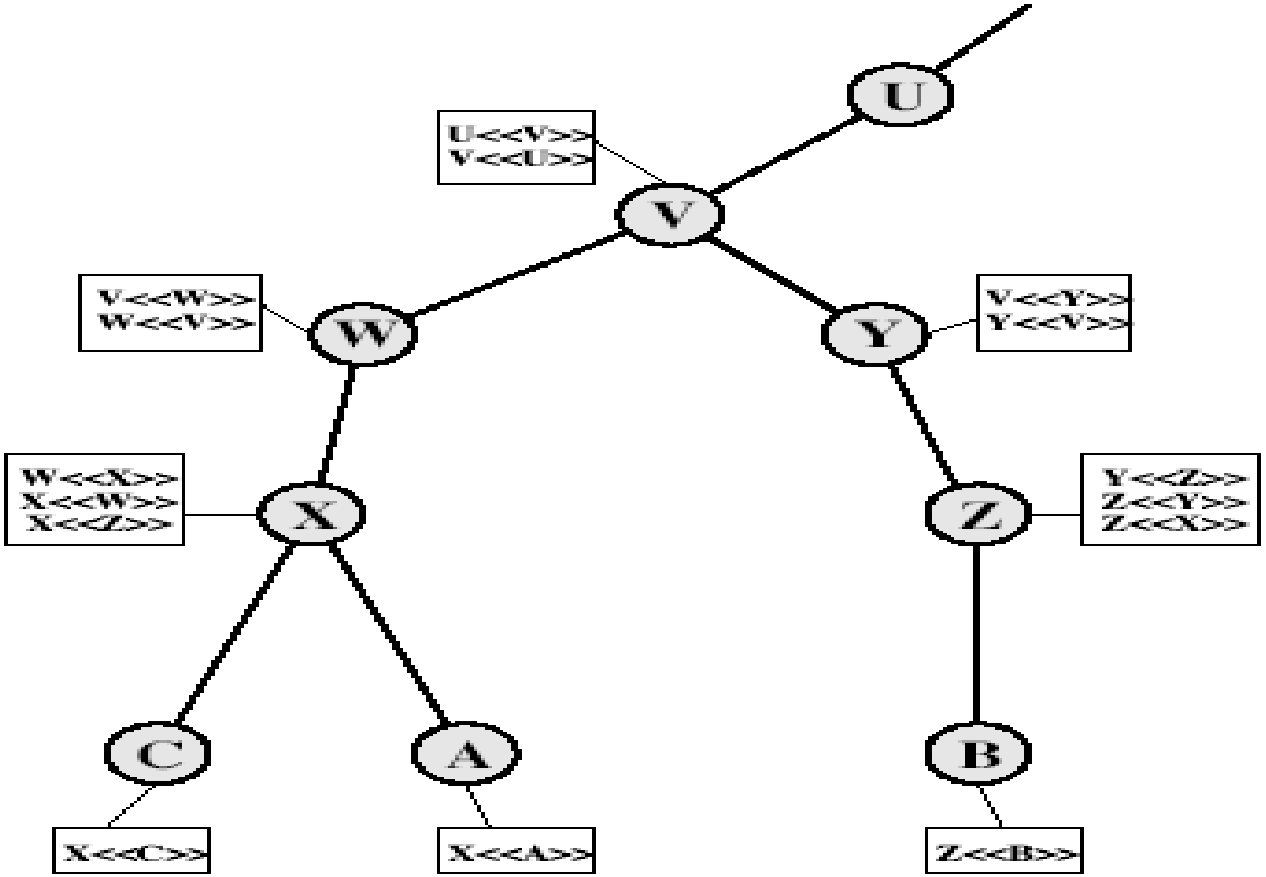
use certificates linking members of hierarchy to validate other CA's

- 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

CA Hierarchy Use



Track chains of certificates:
A acquires B certificate using chain: X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>
B acquires A certificate using chain: Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>

Certificate Revocation

certificates have a period of validity

may need to revoke before expiry, eg:

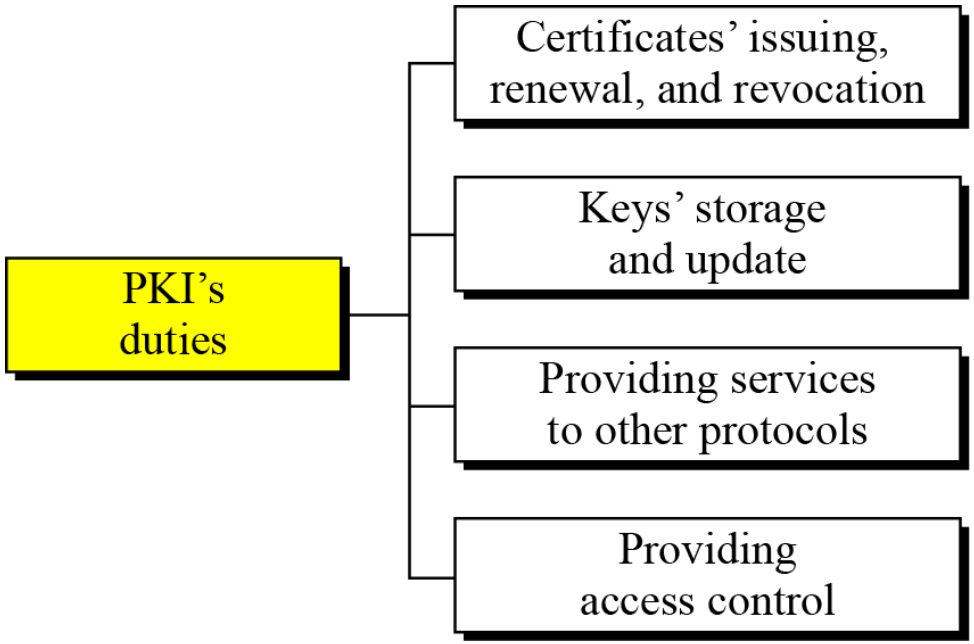
1. user's private key is compromised
2. user is no longer certified by this CA
3. CA's certificate is compromised

CA’s maintain list of revoked certificates

- the Certificate Revocation List (CRL)

users should check certs with CA’s CRL

Public-Key Infrastructures (PKI)



References

1. http://www.sfu.ca/~ljilja/ENSC427/News/Kurose_Ross/Chapter_8_V7.0_Accessible.pdf
2. Stallings, W., Cryptography and Network Security: Principles and Practice 0133354695, 9780133354690.
3. **A.K. Dewdney, The New Turning Omnibus, pp. 250-257, Henry Holt and Company, 2001.**
4. www.whatis.com (search for kerberos)
5. Bryant, W. Designing an Authentication System: A Dialogue in Four Scenes. http://web.mit.edu/kerberos/www/dialogue.html
6. Kohl, J.; Neuman, B. “The Evolotion of the Kerberos Authentication Service” http://web.mit.edu/kerberos/www/papers.html
7. http://www.isi.edu/gost/info/kerberos/

Lnu.se