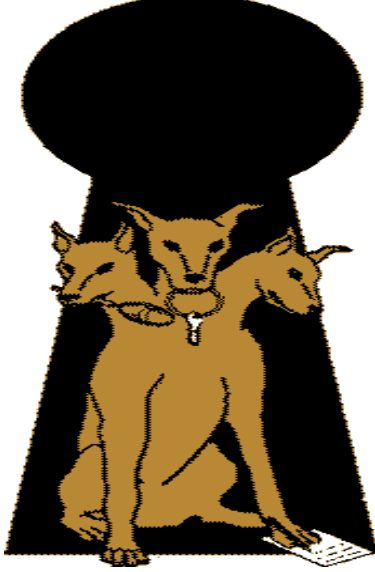
Authentication Applications

Authentication Applications

- authentication functions developed to support **application-level** authentication & digital signatures
- will consider Kerberos – a private-key authentication service
- then X.509 directory - public key authentication service



- Kerberos was developed for

Project Athena at the MIT

- Greek mythology → Kerberos (Cerberus)

was a three-headed dog who guarded

the gates of Hades. The three heads of the Kerberos protocol represent the following:

- the client or principal;
- the network resource, which is the application server that provides access to the network resource; and
- a key distribution center (KDC), which acts as Kerberos' trusted third-party authentication service.

Kerberos

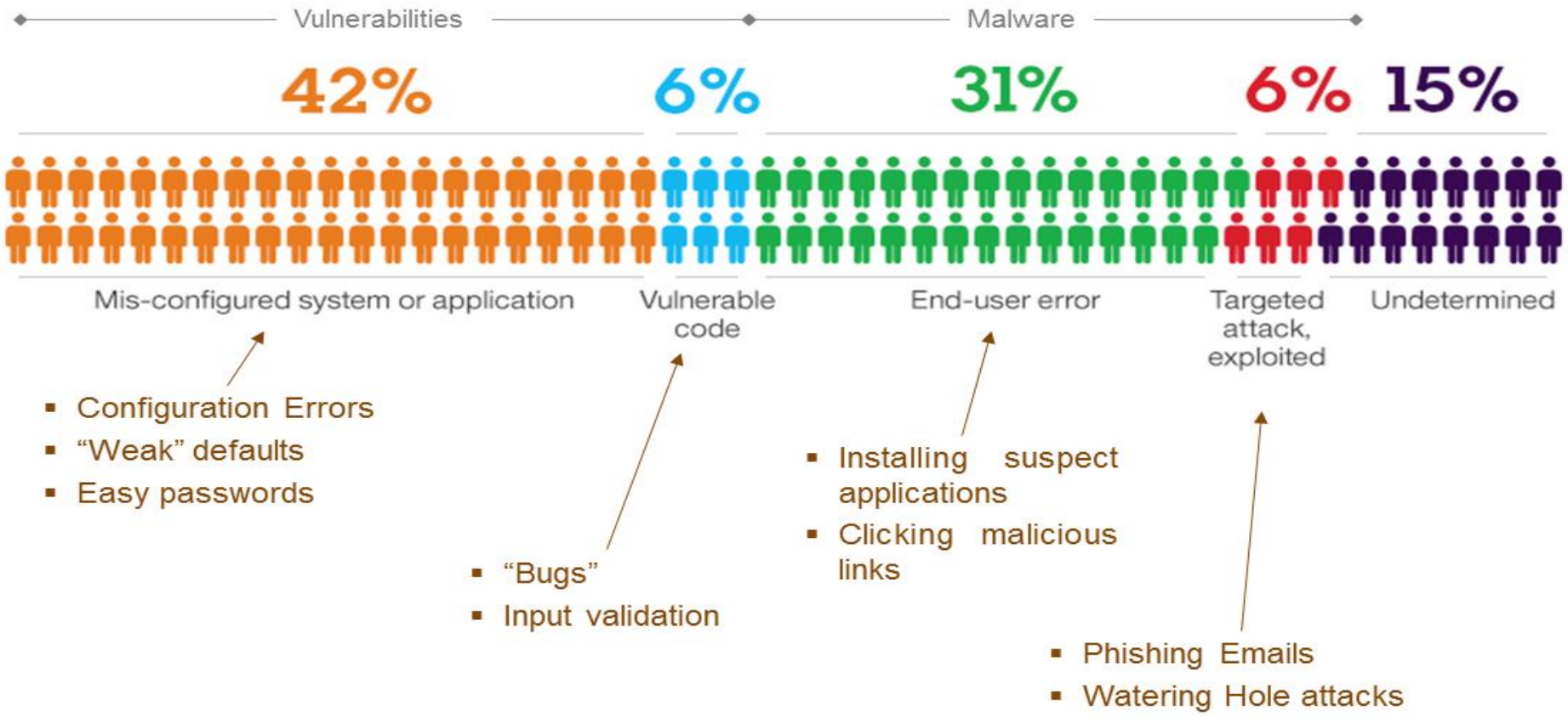
- Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.
- **trusted key server system** from MIT
- provides **centralised private-key third-party authentication in a distributed network**
  - allows users access to services distributed through network
  - without needing to trust all workstations
  - rather all trust a central authentication server
- Unlike most other authentication schemes Kerberos relies **exclusively on symmetric encryption**, making no use of public-key encryption.
- two versions in use: 4 & 5

Firewall vs. Kerberos?

- Firewalls make a *risky assumption*: that attackers are coming from the outside. In reality, attacks frequently come from within.

- Kerberos assumes that *network connections* (rather than servers and work stations) are the weak link in network security.

Why do Breaches Happen?



Source: *IBM Security Services 2013 Cyber Security Intelligence Index*

Kerberos Requirements

- Three approaches to security can be envisioned:

  1. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy **based on user identification (ID).**

  2. Require that **client systems authenticate themselves** to servers, but trust the client system **concerning the identity of its user**.

  3. Require the **user to prove identity for each service** invoked. Also **require that servers prove their identity to clients**.

- first published report identified its requirements as:

  1. Secure: A network eavesdropper should not be able to obtain the necessary information to impersonate a user.

  2. Reliable: Kerberos should be highly reliable and should employ distributed server architecture, with one system able to back up another.

  3. Transparent: Ideally, the user should not be aware that authentication takes place, beyond the requirement to enter a password.

  4. Scalable: The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture

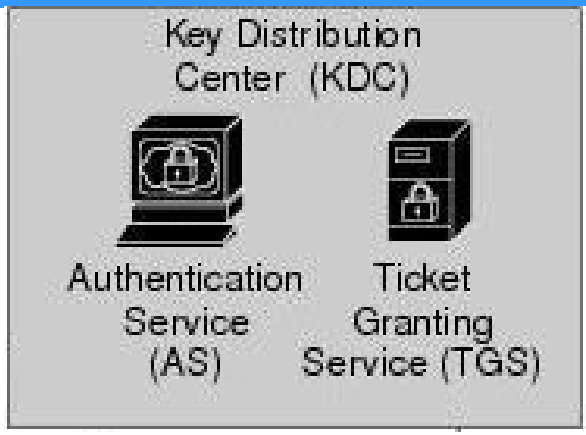
- implemented using an authentication protocol based on

Needham-Schroeder

Kerberos 4 Overview

- Third-party authentication scheme

(i.e. KDC involves two servers)



- 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

Kerberos 4 Overview-Simple Authentication Dialogue

(1) C → AS : *IDc* || *Pc* || *IDv*
(2) AS → C : *Ticket*
(3) C → V : *IDc* || *Ticket*
Ticket = *EKv [IDc* || *ADc* || *IDv]*

where
C = client
AS = authentication server
V = server
IDc = identifier of user on C
IDv = identifier of V
Pc = password of user on C
ADc= network address of C
Kv = secret encryption key shared by AS and V
|| = concatenation

Kerberos 4 Overview-Simple Authentication Dialogue

- AS – Knows passwords of all users and stores in central database
- AS shares **unique secret key with each server** and is distributed in secure manner
- User logs on and requests for server V
- Client module on user's workstation sends message to AS as in 1)
- AS checks its database for valid **password+permission** to access server V
- Once confirmed AS sends ticket to C as in 2)
- With this ticket C can apply to V for service
- **Encryption do not allow C or opponent to change ticket.**

A More Secure Authentication Dialogue

Scheme:

- **Avoid plain text passwords**
- **Introduction of TGS**

Once per user logon session:

(1) C → AS: *IDc* || *IDtgs*

(2) AS → C: *EKc* [*Ticket*$_{tgs}$.]

Once per type of service:

(3) C → TGS: *IDc* || *IDv* || *Ticket*$_{tgs.}$

(4) TGS → C: *Ticket*$_v$

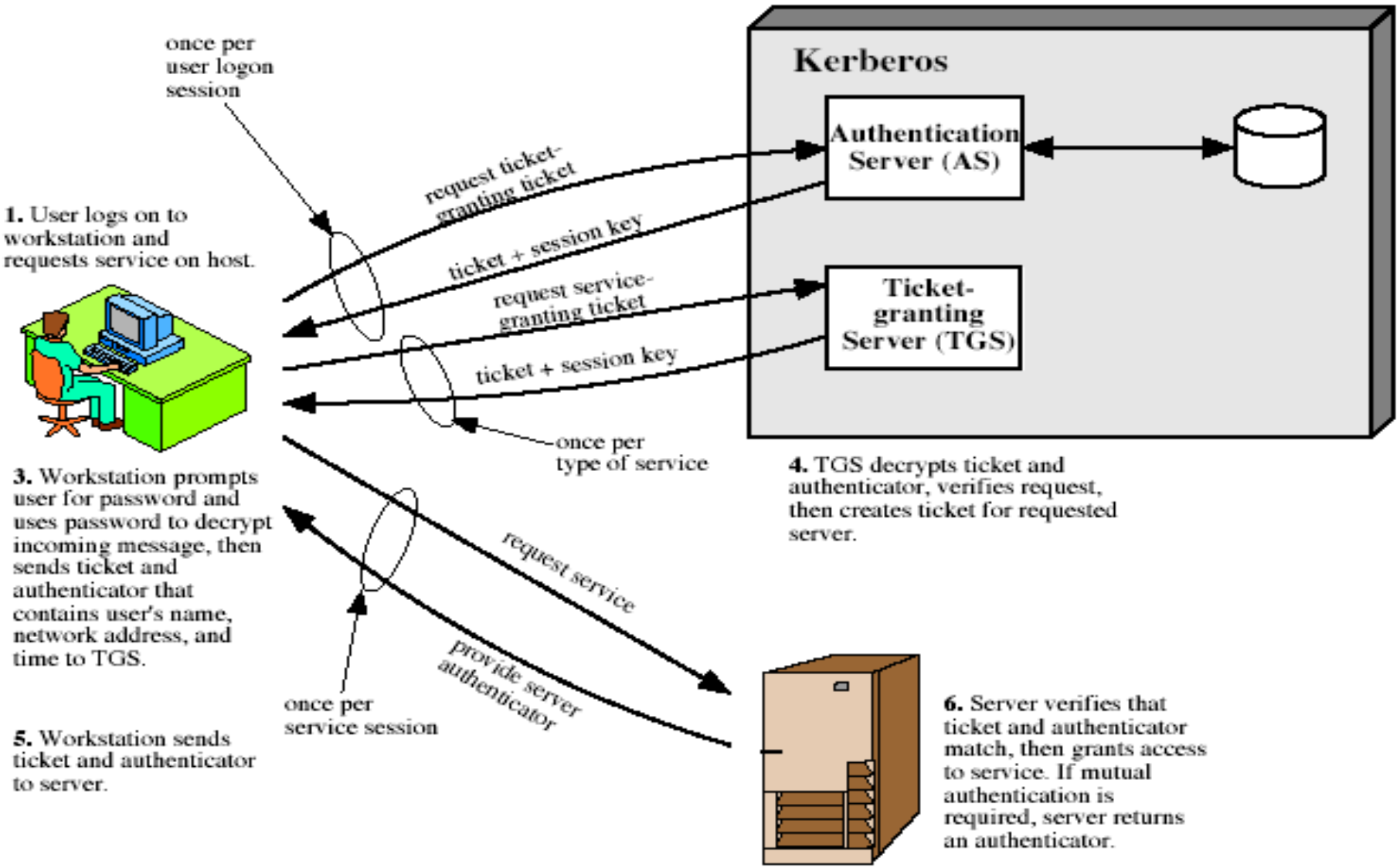
Once per service session:

(5) C → V: *IDc* || *Ticket*$_v$

Ticket$_{tgs}$ = *EKtgs* [*IDc* || *ADc* || *IDtgs* || *TS1I Lifetime1]*

Ticket$_v$ = *EKv* [*IDc* || *ADc* || *IDv* || *TS2* || *Lifetime2]*

Kerberos 4 Overview



A More Secure Authentication Dialogue

Once per user logon session:

(1) C → AS: *IDc* || *IDtgs*

(2) AS → C: *EKc [Ticket$_{tgs}$.]*

• On behalf of user client requests TGT to AS, indicating a request to use the TGS service

• AS responds with a ticket encrypted with a key derived from user's password

• When this response arrives at client, asks for user password, generates key, tries to decrypt the message

• With correct password successful decryption and hence the recovery.

Ticket$_{tgs}$ = EKtgs [IDc || *ADc* || *IDtgs* || *TS1I Lifetime1]*

• **Encryption with secret key known only to AS & TGS**

A More Secure Authentication Dialogue

Once per type of service:

(3) C → TGS: $IDc \parallel IDv \parallel Ticket_{tgs.}$

(4) TGS → C: $Ticket_v$

$Ticket_{tgs} = EKtgs\,[IDc \parallel ADc \parallel IDtgs \parallel TS1 \parallel Lifetime1]$

$Ticket_v = EKv\,[IDc \parallel ADc \parallel IDv \parallel TS2 \parallel Lifetime2]$

- For user client requests a service
- TGS decrypts and checks for lifetime not expired
- **If the user is permitted access to V, TGS issues a ticket to grant access to required service**

A More Secure Authentication Dialogue

Once per service session:

(5) C → V: $IDc \parallel Ticket_v$

$Ticket_v = EKv\ [IDc \parallel ADc \parallel IDv \parallel TS2 \parallel Lifetime2]$

• With particular service granting ticket, client can gain access to corresponding service.

Kerberos Realms

- a Kerberos environment consists of:
  - a Kerberos server
  - a number of clients, all registered with server
  - application servers, sharing keys with server
- this is termed a realm
  - typically a **single administrative domain**
- if have multiple realms, their Kerberos servers must share keys and trust

Kerberos Realms

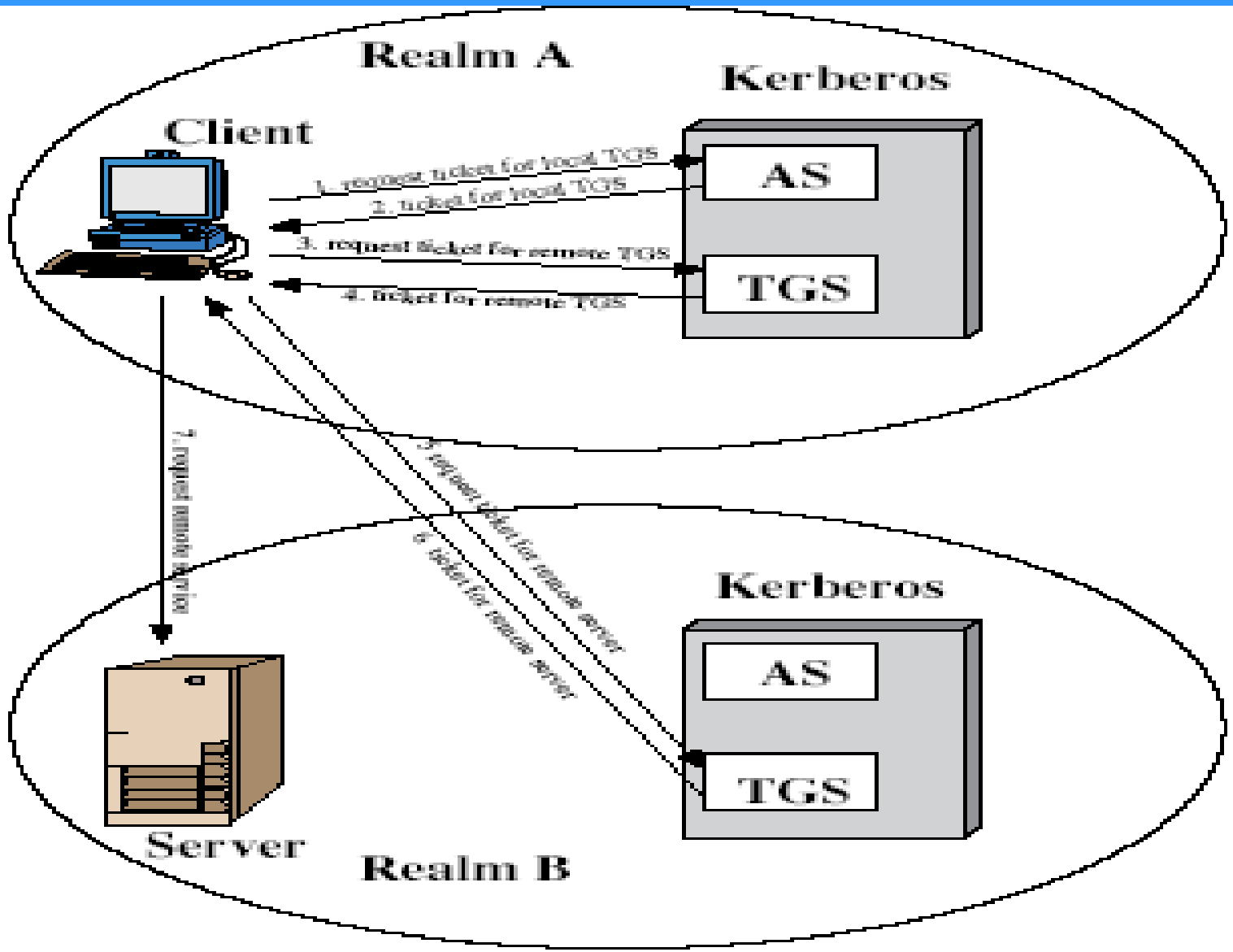


Figure 14.2 Request for Service in Another Realm

Kerberos Realms

(1) $C \rightarrow AS$: $ID_c \| ID_{tgs} \| TS_1$

(2) $AS \rightarrow C$: $E_{K_c}[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}]$

(3) $C \rightarrow TGS$: $ID_{tgsrem} \| Ticket_{tgs} \| Authenticator_c$

(4) $TGS \rightarrow C$: $E_{K_{c,tgs}}[K_{c,tgsrem} \| ID_{tgsrem} \| TS_4 \| Ticket_{tgsrem}]$

(5) $C \rightarrow TGS_{rem}$: $ID_{vrem} \| Ticket_{tgsrem} \| Authenticator_c$

(6) $TGS_{rem} \rightarrow C$: $E_{K_{c,tgsrem}}[K_{c,vrem} \| ID_{vrem} \| TS_6 \| Ticket_{vrem}]$

(7) $C \rightarrow V_{rem}$: $Ticket_{vrem} \| Authenticator_c$

1) Request ticket for local TGS ***2) Ticket for Local TGS***

3) Request ticket for remote TGS ***4) Ticket for remote TGS***

5) Request ticket for remote server ***6) Ticket for remote server***

7) Request remote server

Kerberos Version 5

- **Realms :** Indicates the realm of the user
- **Options:** Used to request that certain flags be sent in the returned ticket.
- **Times:** Used by the client to request following time settings

  from: the desired start time for the requested ticket

  till: The requested expiration time for the requested ticket

  rtime: Requested renew-till time
- **Nonce:** Random value to be repeated in message to assure that the response is fresh and has not been replayed by an opponent
- **Flags:** Reflect status of the ticket and requested options.

Kerberos Version 5

- developed in mid 1990's
- provides improvements over v4
- specified as Internet standard RFC 1510
- **Differences between version 4 and version 5**

Due to Environmental Shortcomings:

1. Encryption system dependence: Version 4 requires the use of DES. In version 5, ciphertext is tagged with an encryption type identifier so that any encryption technique may be used. Encryption keys are tagged with a type and a length.

2. Internet protocol dependence: Version 4 requires the use of Internet Protocol (IP) addresses. Other address types, such as the ISO network address, are not accommodated. Version 5 network addresses are tagged with type and length, allowing any network address type to be used.

3. Message byte ordering: In version 4, the sender of a message employs a byte ordering of its own choosing. In version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.

Kerberos Version 5

4. Ticket lifetime: Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. Thus, the maximum lifetime that can be expressed 2^8 X 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.

5. 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.

6. Interrealm authentication: In version 4, interoperability among *N* realms requires on the order of N^2 Kerberos-to-Kerberos relationships. Version 5 supports a method that requires fewer relationships.

Kerberos Version 5

Due to Technical Deficiencies:

1. **Double encryption:** Messages that tickets provided to clients are encrypted twice, once with the secret key of the target server and then again with a secret key known to the client. The second encryption is not necessary and is computationally wasteful.

2. **PCBC encryption:** Encryption in version 4 makes use of a nonstandard mode of DES known as propagating block chaining (PCBC)-vulnerable to an attack. Version 5 uses standard CBC mode to be used for encryption.

3. **Session keys:** Each ticket includes a session key that is used by the client to encrypt the authenticator sent to the service associated with that ticket. In addition, the session key may subsequently be used by the client and the server to protect messages passed during that session-replay attack possible. In version 5, it is possible for a client and server to negotiate a sub session key, which is to be used only for that one connection.

4. **Password attacks:** Both versions are vulnerable to a password attack.

Weaknesses and Solutions

If TGT stolen, can be used to access network services	Only a problem until ticket expires in a few hours.
Subject to dictionary attack	Timestamps require hacker to guess in 5 minutes.
Critical to compromise Authentication Server	Physical protection for the server.

However, Scalability is still major limitation……

The Competition: SSL

SSL	Kerberos
Uses public key encryption	Uses private key encryption
Is certificate based (asynchronous)	Relies on a trusted third party (synchronous)
Ideal for the WWW	Ideal for networked environments
Key revocation requires Revocation Server to keep track of bad certificates	Key revocation can be accomplished by disabling a user at the Authentication Server
Certificates sit on a users hard drive (even if they are encrypted) where they are subject to being cracked.	Passwords reside in users' minds where they are usually not subject to secret attack.
Uses patented material, so the service is not free. Netscape has a profit motive in wide acceptance of the standard.	Kerberos has always been open source and freely available.

Kerberos brute-force

Brute-forcing Kerberos is comparatively simpler because:

- No domain account is needed to conduct the attack; just connectivity to the KDC.
- Kerberos pre-authentication errors are not logged in Active Directory with a normal Logon failure events
- even if the password is wrong, Kerberos indicates whether the username is correct or not; makes possible to discover user accounts(ASREPRoast attack)
- However, by carrying out a brute-force attack it is also possible to **block user accounts**

ASREPRoast

- The ASREPRoast attack looks for users without Kerberos pre-authentication

(pre-authentication→ security feature which offers protection against password-guessing attacks.)

- That means that anyone can send an AS_REQ request to the KDC on behalf of any of those users, and receive an AS_REP message. This message contains a chunk of data encrypted with the original user key, derived from its password.
- by using this message, the user password could be cracked offline.

X.509 Authentication Service

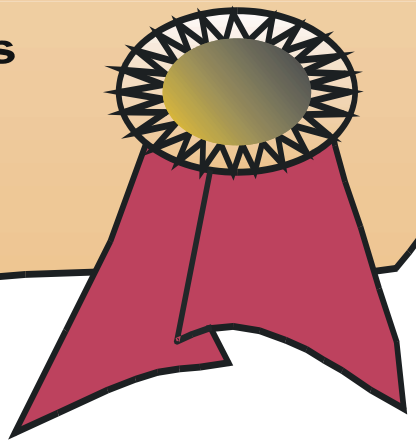
- part of CCITT X.500 directory service standards
  - distributed servers maintaining some info database
- defines framework for authentication services
  - directory may store public-key certificates
  - with public key of user
  - signed by certification authority
- also defines authentication protocols
- uses public-key crypto & digital signatures
  - algorithms not standardised, but RSA recommended

X.509 Authentication Service

- X.509 is the Internationally accepted standard
  - how to construct a public key certificate,
  - has gone through several versions.
- used by S/MIME secure email, SSL/TLS secure Internet links (eg for secure web).

CERTIFICATE

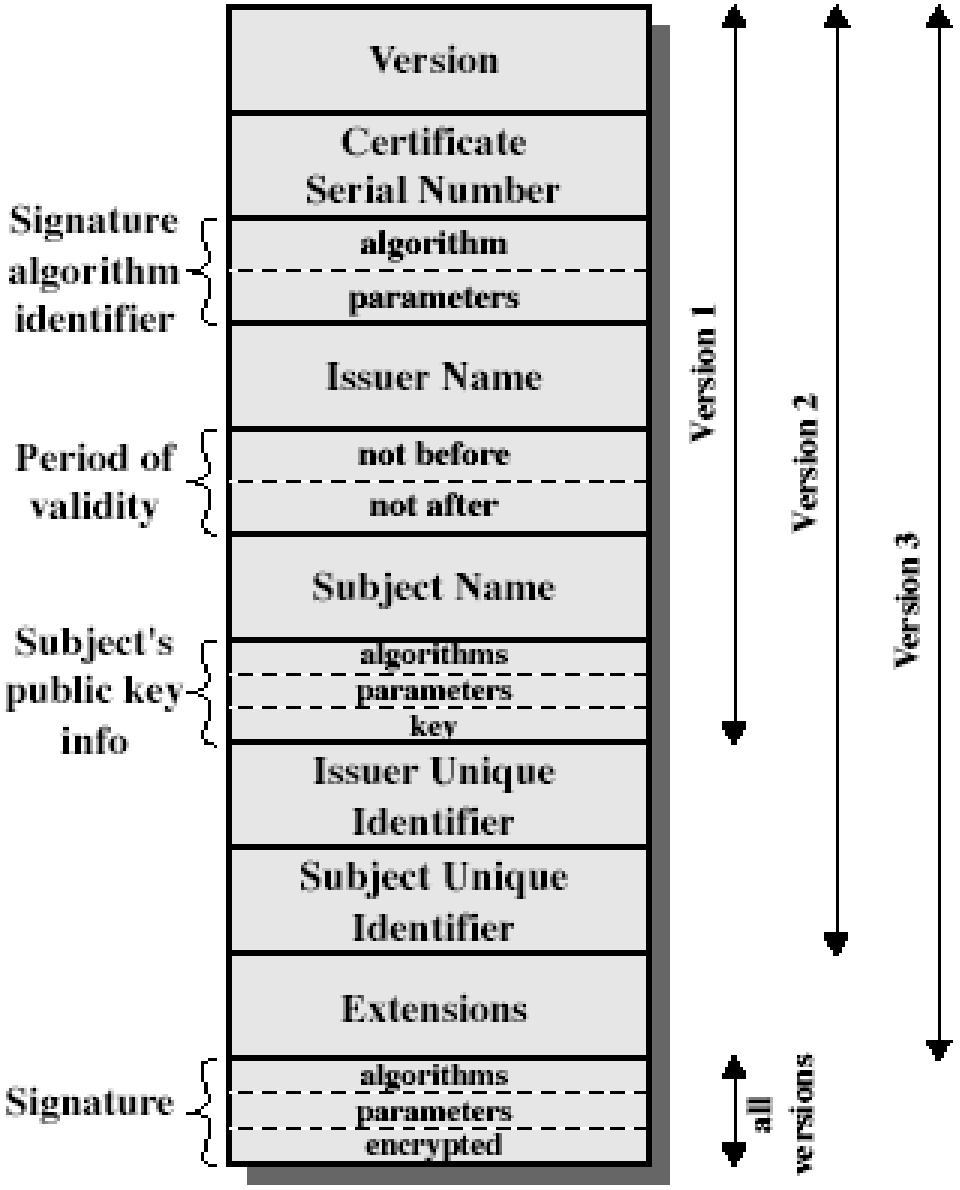
- Issuer (CA) Distinguished Name (DN) e.g. C=GB, O=Baltimore Technologies, OU=PSD
- Serial number (allocated by CA)
- Validity period (typically a year)
- User (Subject) Distinguished Name
- User Public Key parameters e.g. RSA
- User Public Key
- Extensions e.g.
  - Alternative user name (e.g. e-mail address)
  - Key usage e.g. digital signature, key encipherment
- Signing algorithm parameters e.g. SHA-1, MD5
- CA Signature



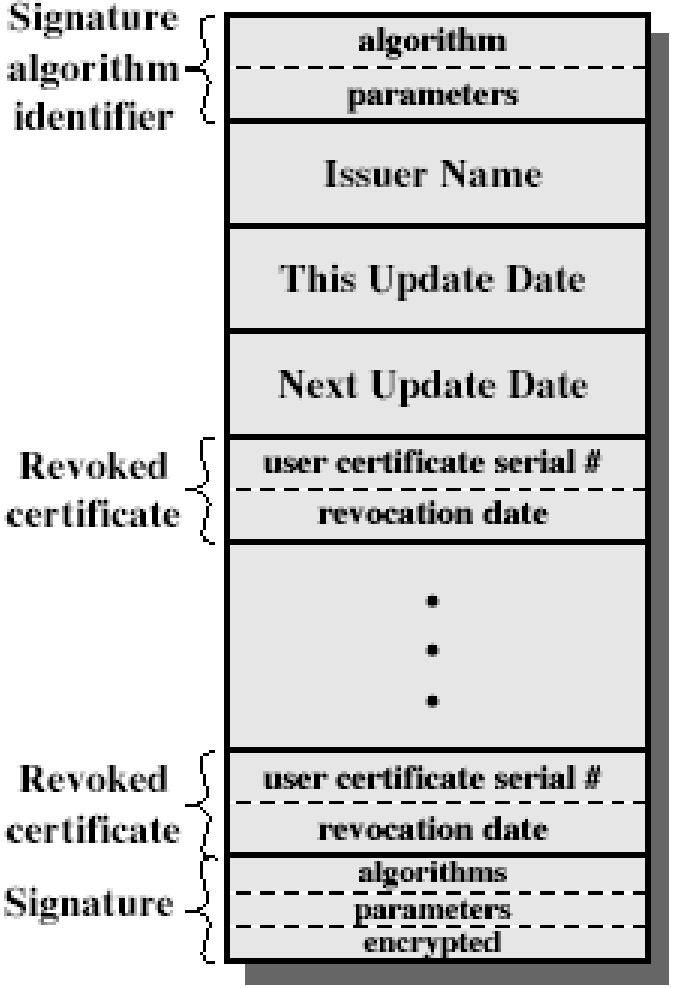
X.509 Certificates

- 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

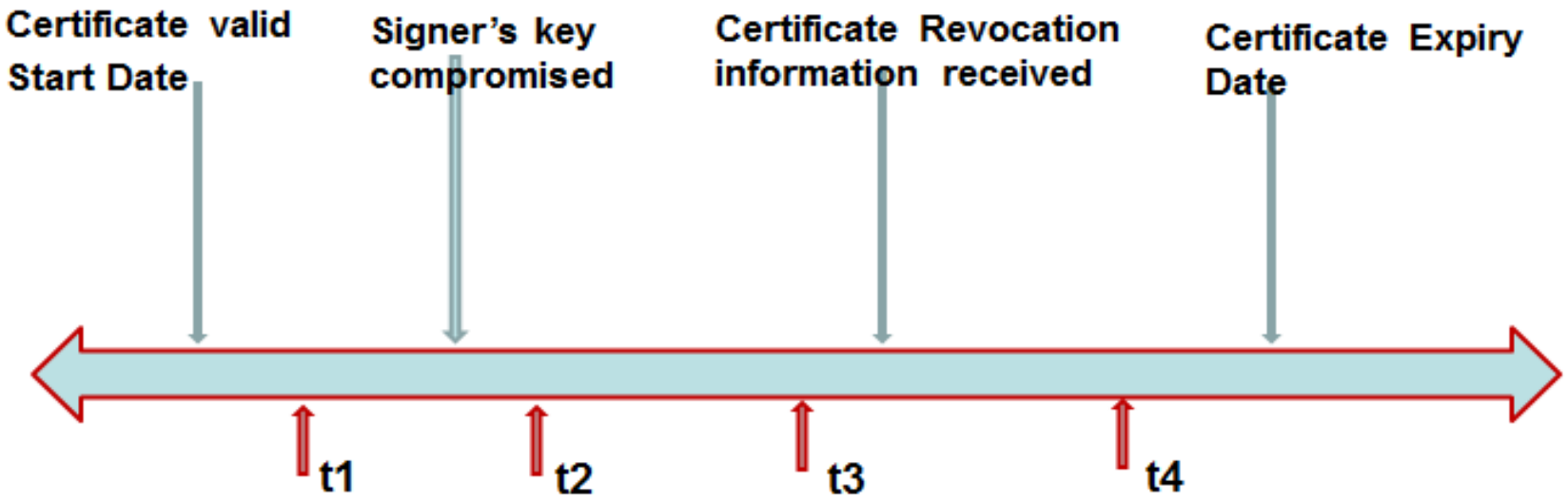
Certificate Revocation

- 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
- CAs maintain list of revoked certificates
  - the Certificate Revocation List (CRL)
- users should check certificates with CA’s CRL

Certificate revocation lists; relevant issues

- CRL do not operate in real times; speed of CRL updates may mismatch
- Each CA is responsible for updating and maintaining its own CRL; however non-reachability to CA cause availability or integrity issue
- revocations are done by overwriting the previous file; unavailability of historical data

CASE.. Certificate Revocation Lag Time Vs. Signature Acceptance



CASE	Signature Generation Time	Signature Verification Time	Does Verifier Accept Signature?	Should Verifier Accept Signature?
1	t1	t3	Y	Y
2	t1	t4	?	?
3	t2	t3	?	?
4	t2	t4	?	?

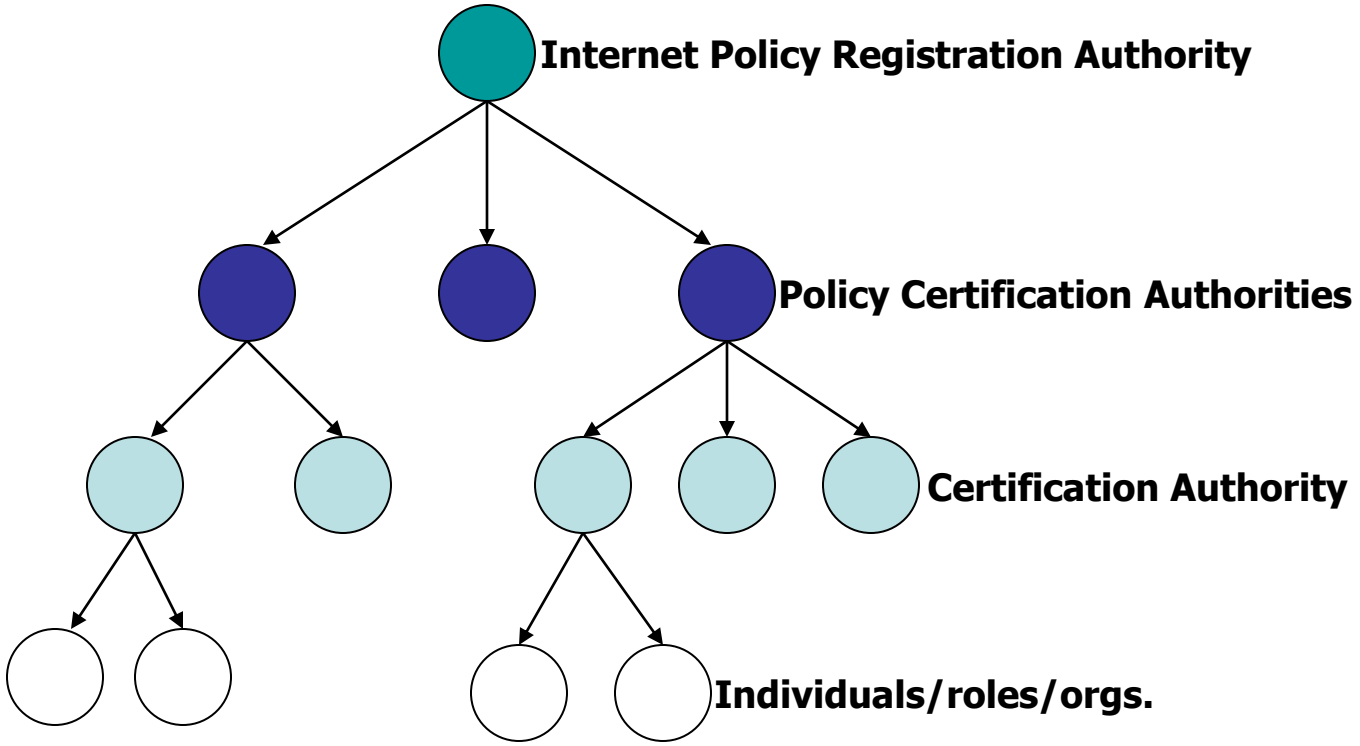
Obtaining a Certificate

- **any user with access to CA** can get any certificate from it
- **only CA can modify** a certificate
- because cannot be forged, **certificates can be placed in a public directory**

Advantages of CA Over KDC

- CA does not need to be on-line all the time!
- CA can be very simple computing device.
- If CA crashes, life goes on (except CRL).
- Certificates can be stored in an insecure manner!!
- Compromised CA cannot decrypt messages.
- Scales well.

Internet Certificate Hierarchy



X509 PKI – Technical View

Basic Components:

- Certificate Authority (CA)
- Registration Authority (RA)
- Certificate Distribution System

"Provider" Side

- PKI enabled applications

"Consumer" Side

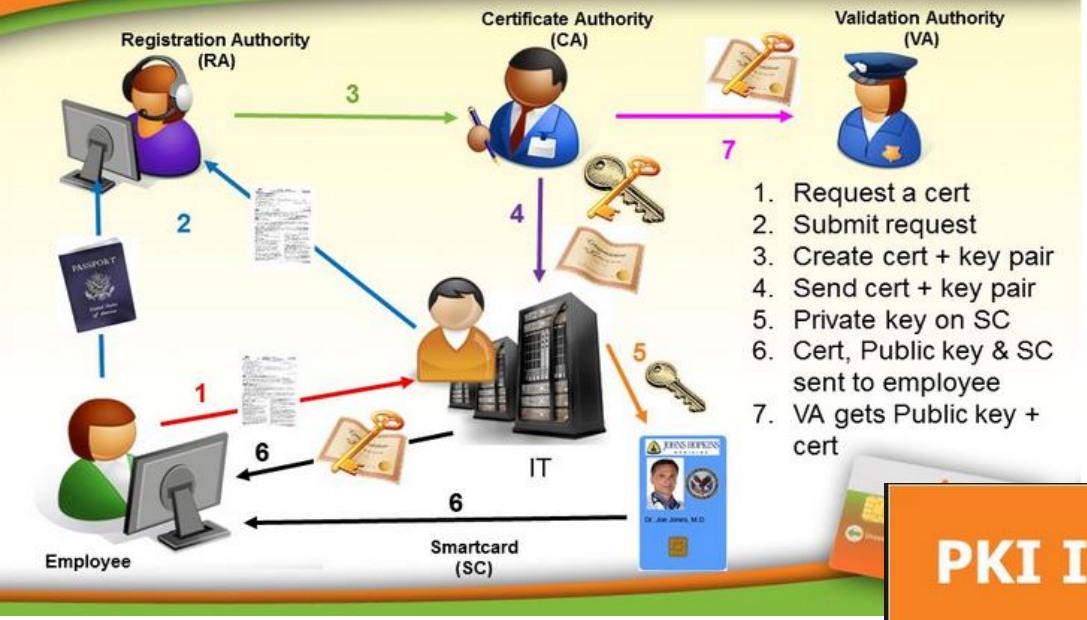
Basic Tasks: (CA)

- Key Generation
- Digital Certificate Generation
- Certificate Issuance and Distribution
- Revocation
- Key Backup and Recovery System
- Cross-Certification

Basic Tasks: (RA)

- Registration of Certificate Information
- Face-to-Face Registration
- Remote Registration
- Automatic Registration
- Revocation

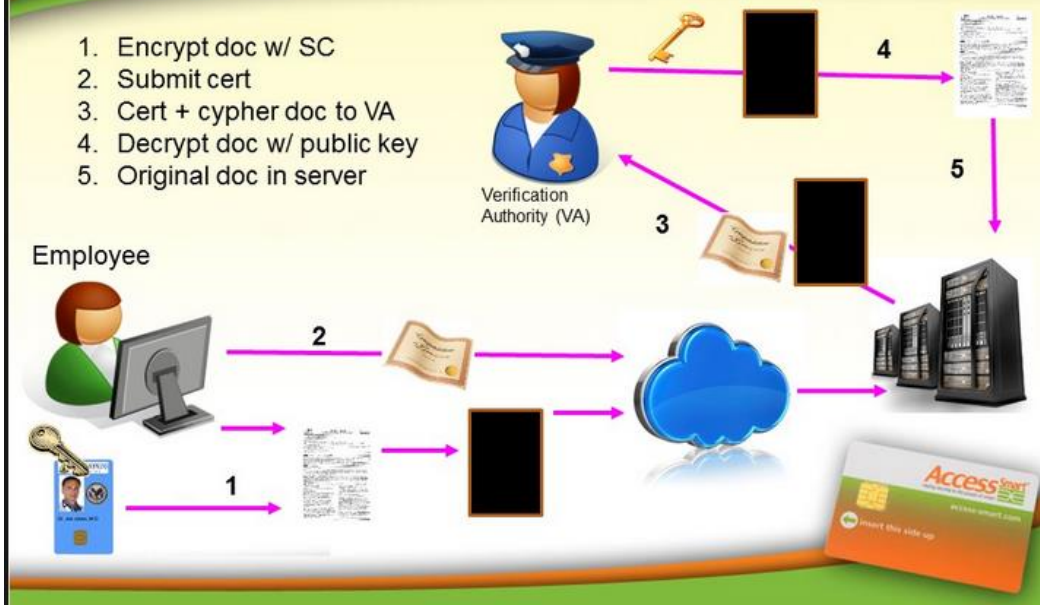
PKI Infrastructure - Issuance



1. Request a cert
2. Submit request
3. Create cert + key pair
4. Send cert + key pair
5. Private key on SC
6. Cert, Public key & SC sent to employee
7. VA gets Public key + cert

PKI Infrastructure - Usage

1. Encrypt doc w/ SC
2. Submit cert
3. Cert + cypher doc to VA
4. Decrypt doc w/ public key
5. Original doc in server



Types of certificates

- Organizational Certificates
  Principal's affiliation with an organization
- Residential certificates
  Principal's affiliation with an address
- Personal Certificates
  Principal's Identity
- Principal need not be a person. It could be a role.

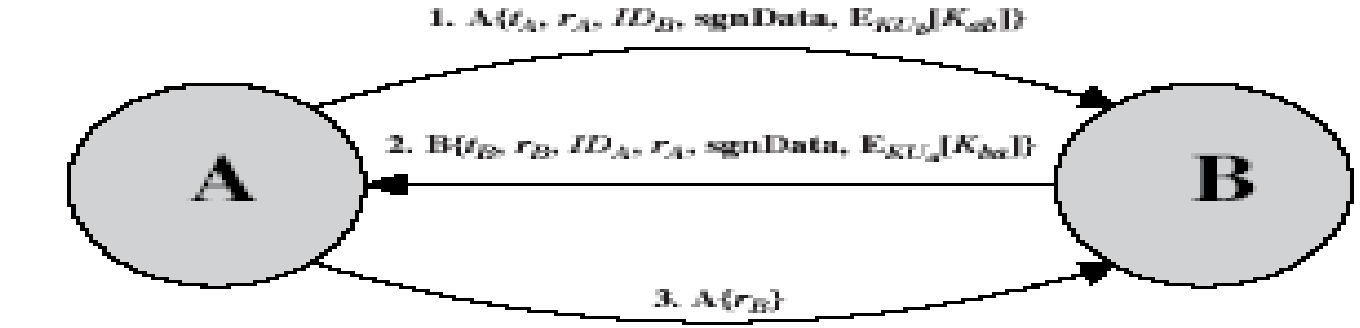
Authentication Procedures

- X.509 includes three alternative authentication procedures:
  - One-Way Authentication
  - Two-Way Authentication
  - Three-Way Authentication
- all use public-key signatures

Authentication Procedures

(a) One-way authentication

(b) Two-way authentication



(c) Three-way authentication

One-Way Authentication

- 1. message (A→ B) used to establish
  - the identity of A and that message is from A
  - message was intended for B
  - integrity & originality of message
- message must include timestamp, nonce, B's identity and is signed by A

Two-Way Authentication

- 2. messages (A→B, B→A) which also establishes in addition:
  - the identity of B and that reply is from B
  - that reply is intended for A
  - integrity & originality of reply
- reply includes original nonce from A, also timestamp and nonce from B

Three-Way Authentication

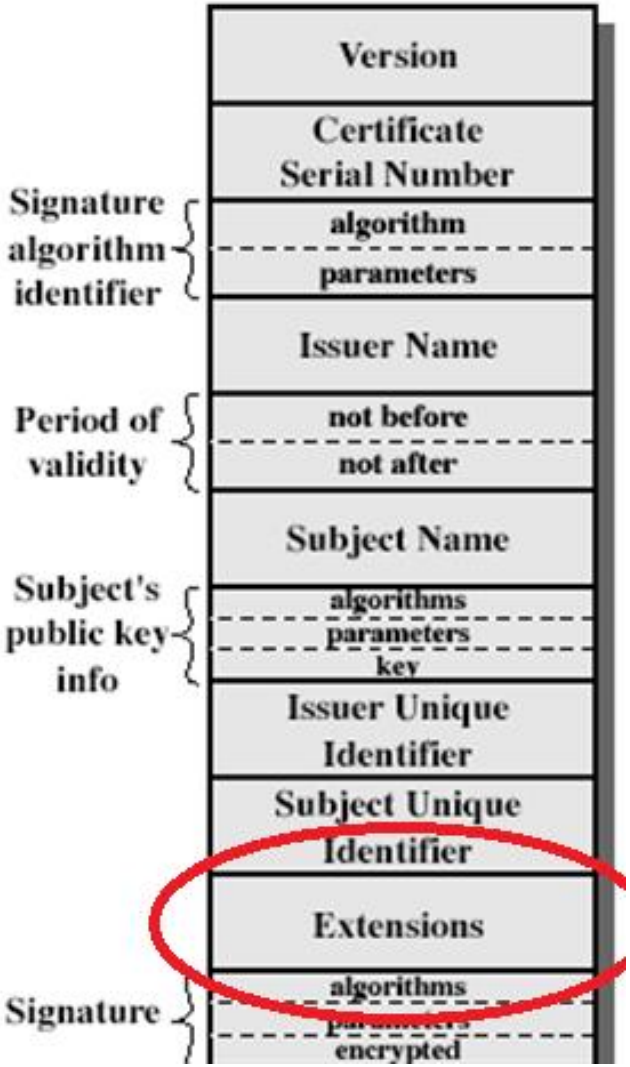
- 3. messages (A→B, B→A, A→B) which enables above authentication **without** synchronized clocks
- has reply from A back to B containing signed copy of nonce from B
- means that timestamps need not be checked or relied upon

Problems with PKI - Conflicts

- X.509, and PGP remain silent on conflicts.
- They assume CA's will ensure that no conflicts arise.
- But in practice conflicts may exist –
  - John A. Smith and John B. Smith may live at the same address

X.509 Version 3

- has been recognised that additional information is needed in a certificate
  - email/URL, policy details, usage constraints
- rather than explicitly naming new fields defined a general extension method
- extensions consist of:
  - key and policy information
  - certificate subject and issuer attributes
  - certificate path constraints



Certificate Extensions

- key and policy information
  - convey info about subject & issuer keys, plus indicators of certificate policy
- certificate subject and issuer attributes
  - support alternative names in alternative formats for certificate subject and/or issuer
- certificate path constraints
  - allow constraints on use of certificates by other CA's

Identity Based Encryption (IBE)

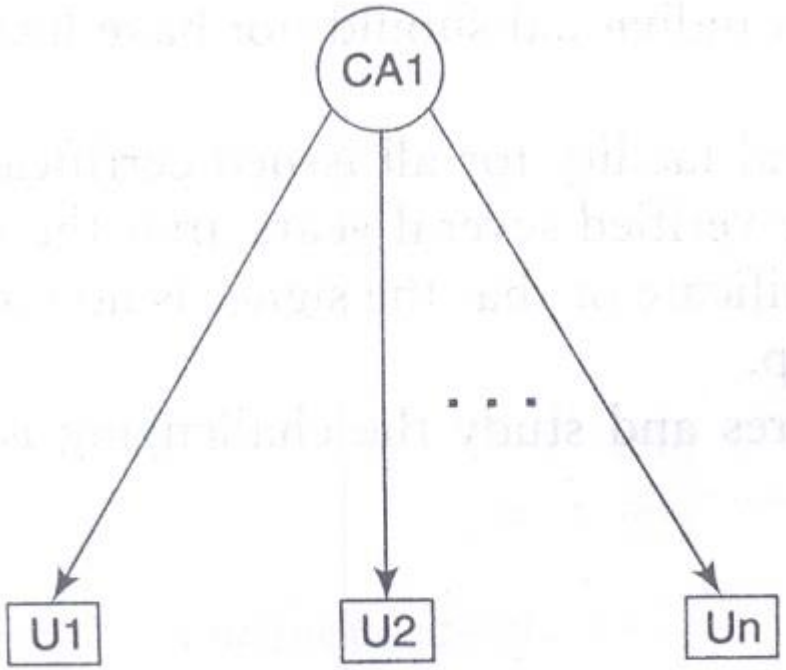
- Provides **verifiable way** of communicating the **public key**
- Public key can be computed as a function of person's **unique credential** e.g. mail ID
- Trusted Third Party (TTP) uses Private Key Generator (PKG) as:

- To obtain the Private key, user informs ID to PKG (xyz@walchandsangli.ac.in)
- The communicator can invoke public key only by knowing users mail ID.

The private key as well as public key are the functions of user's identity

PKI Architectures

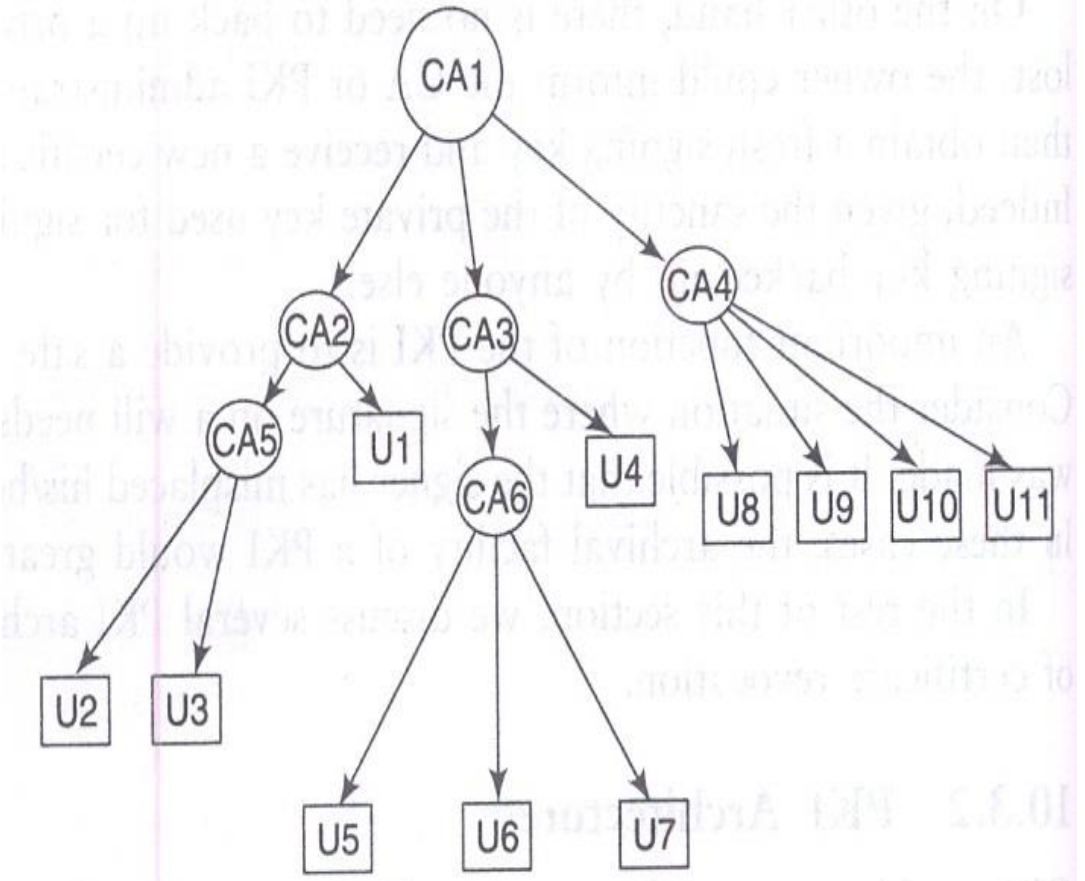
- Central Trust Relationship
- Requires Background checking
- **Problem of scalability**



(a) PKI with single CA

PKI Architectures

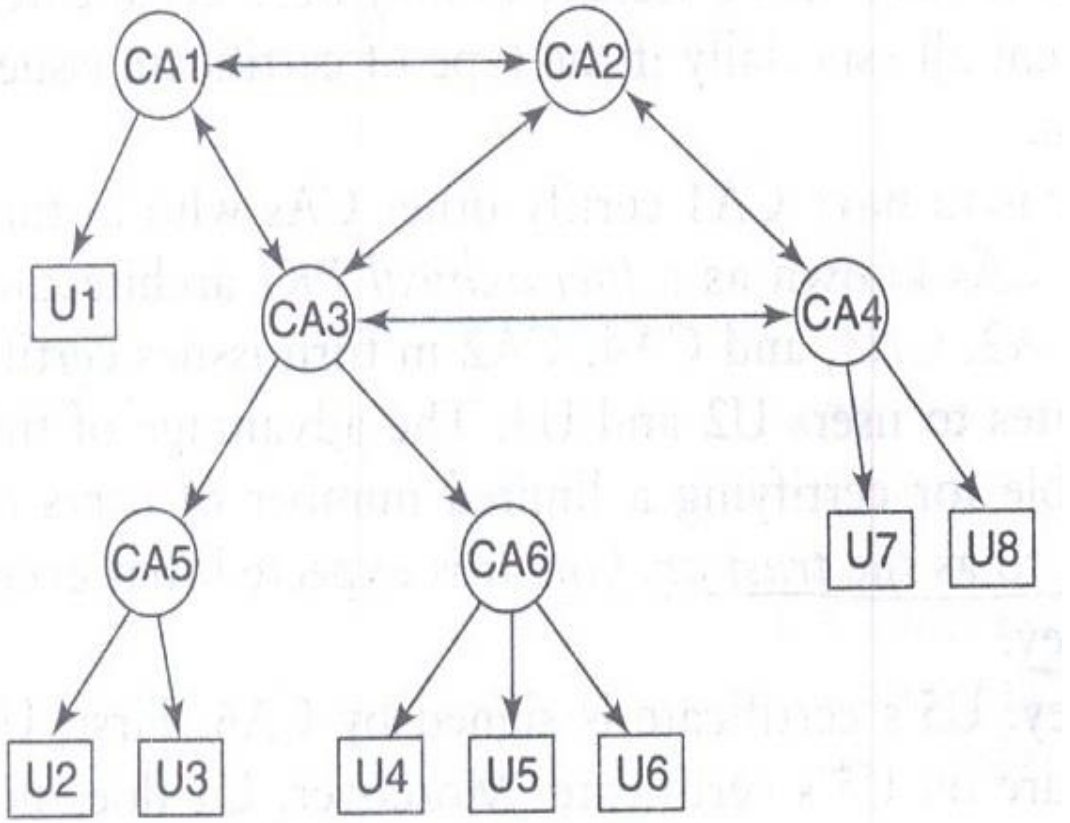
- Limited number of users for each CA
- The Root CA (CA1) is called as **Trust Anchor**
- Every node knows the public key of Trust Anchor



(b) Hierarchical (tree-based) PKI architecture

PKI Architectures

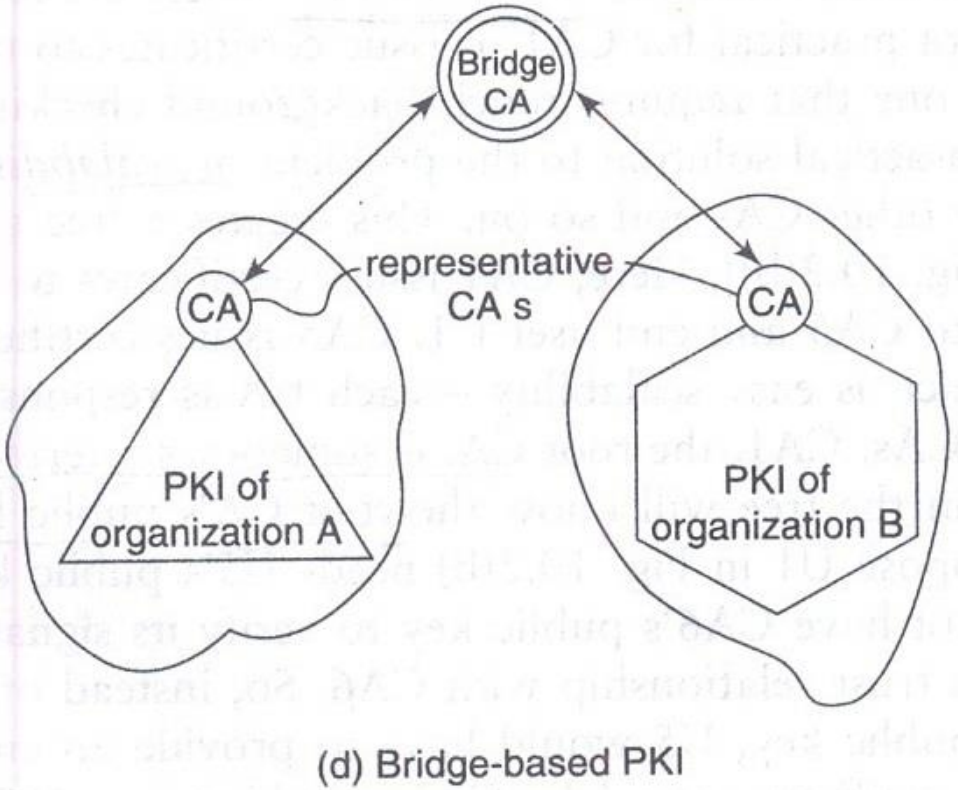
- Dense **Web of Trust**
- Bidirectional Relationship
- **Multiple paths between users** including certificate chain provides **greater resilience**



(c) Mesh-based PKI

PKI Architectures

- **Secure communications** between business partnership



(d) Bridge-based PKI

Questions in mind….

- Can a KDC/CA learn communication between Alice and Bob, to whom it issued keys/issued certificates?
- Alice uses her private key, public key pairs and a CA issued certificate. But is doubtful about key exposure to the opponent,

…… how to compromise the existing keys/ certificates?

CA certificate to match

```
$ openssl x509 -in thawte-ca-certificate.pem -noout -text
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
        OU=Certification Services Division,
        CN=Thawte Server CA/emailAddress=server-certs@thawte.com
    Validity
        Not Before: Aug  1 00:00:00 1996 GMT
        Not After : Dec 31 23:59:59 2020 GMT
    Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
        OU=Certification Services Division,
        CN=Thawte Server CA/emailAddress=server-certs@thawte.com
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c:
          68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
          85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06:
          6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2:
          6a:0c:44:38:cd:fe:be:e3:64:09:70:c5:fe:b1:6b:
          29:b6:2f:49:c8:3b:d4:27:04:25:10:97:2f:e7:90:
          6d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:
          5d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:
          3a:c2:b5:66:22:12:d6:87:0d
        Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Basic Constraints: critical
        CA:TRUE
  Signature Algorithm: md5WithRSAEncryption
    07:fa:4c:69:5c:fb:95:cc:46:ee:85:83:4d:21:30:8e:ca:d9:
    a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c8:48:
    3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
    4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9:
    8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5:
    e7:20:1b:8b:ca:a4:ab:8d:e9:51:d9:e2:4c:2c:59:a9:da:b9:
    b2:75:1b:f6:42:f2:ef:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e:
    70:47
```

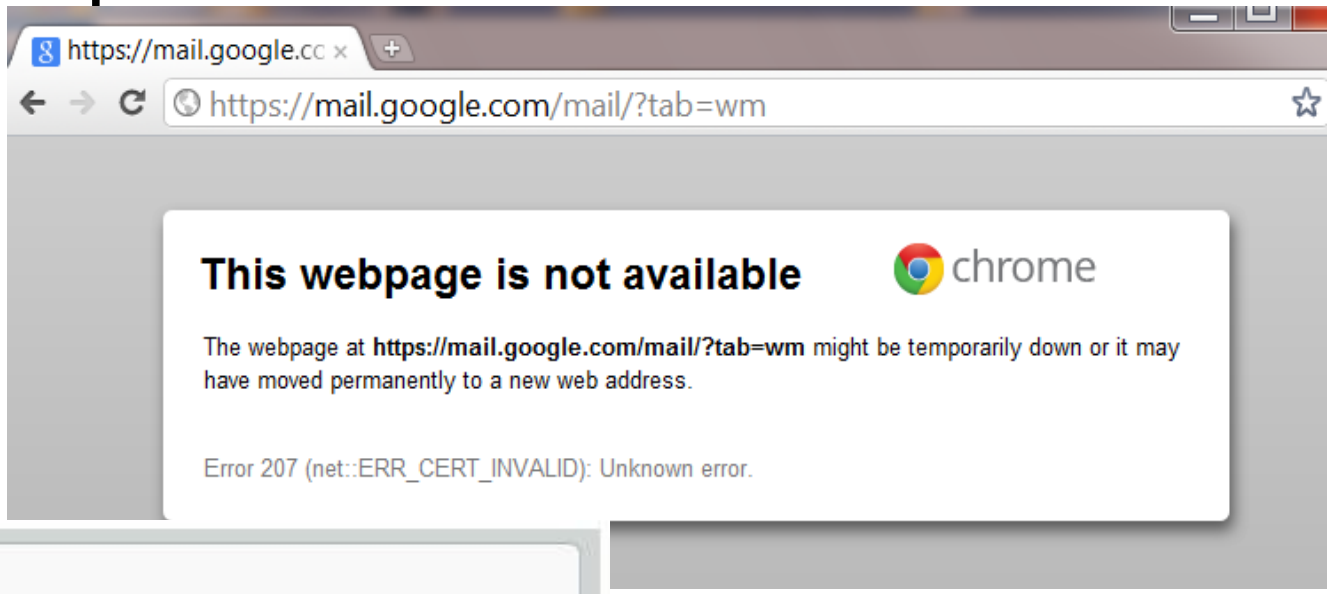
This is an example of a self-signed certificate, as the issuer and subject are the same.

Thawte is one of the root certificate authorities recognized by both Microsoft and Netscape

Challenge.. To fix the problems..

Survey ..case problem classification:

1) Trivial
2) Scary
3) Serious



Tutorial

Verification Engine:

Case COMODO/ VeriSign

- Installation
- Authentication & anti_phishing
- SSL Certification
- E-transactions