

## CS 6v81 - Network Security

*Kerberos and Public Key Infrastructure (PKI)*

## Kerberos

### Kerberos

- ▶ 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
- ▶ Two versions in use: 4 & 5

### Kerberos Requirements

- ▶ Its first report identified requirements as:
  - ▶ Secure
  - ▶ Reliable
  - ▶ Transparent
  - ▶ Scalable
- ▶ Implemented using an authentication protocol based on Needham-Schroeder

### Kerberos

- ▶ Advantages
  - ▶ Secure authentication
  - ▶ Single sign-on to access multiple resources
  - ▶ Secure data flow
- ▶ Applications benefiting from Kerberos
  - ▶ telnet, ftp
  - ▶ BSD rtools (rlogin, rcp, rsh)
  - ▶ NFS
  - ▶ Others (pine, eudora, etc)

5

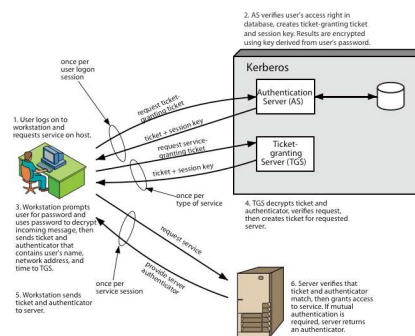
### Kerberos v4 Overview

- ▶ 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

## Kerberos v4 Dialogue

1. Obtain ticket granting ticket from AS
  - ▶ Once per session
2. Obtain service granting ticket from TGT
  - ▶ For each distinct service required
3. Client/server exchange to obtain service
  - ▶ On every service request

## Kerberos 4 Overview

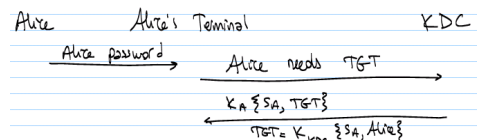


## Kerberos keys

- ▶ Each PRINCIPAL shares a master key with KDC
  - ▶  $K_A$  – Alice's master key used for initial authentication
  - ▶  $S_A$  – Alice's session key, created after initial authentication, used instead of  $K_A$
- ▶  $K_{AB}$  – Alice-Bob session key
- ▶ Ticket Granting Tickets (TGTs)
  - ▶ Issued to Alice by KDC after login
  - ▶ Contains  $S_A$  encrypted with  $K_{KDC}$
  - ▶ Used to obtain session key  $K_{AB}$

9

## Logging into the network



- ▶ does not protect against dictionary attacks with eavesdropping

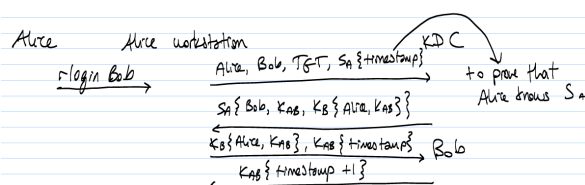
10

## Logging into the network

- ▶ The terminal/workstation
  - ▶ Converts Alice's password into a DES key
  - ▶ When receives the credentials from the server
    - ▶ Decrypts them using this DES key
  - ▶ If decrypts correctly, authentication is successful
  - ▶ Discards Alice's master key, retains TGT and  $S_A$
  - ▶ TGT contains all the information KDC needs about Alice's session; hence KDC can work w/o remembering any volatile data

11

## Accessing a remote principal



- ▶ Afterwards, the traffic between Alice and Bob can be
  - ▶ Unprotected
  - ▶ Authenticated
  - ▶ Encrypted and authenticated

12

## Replicated KDCs

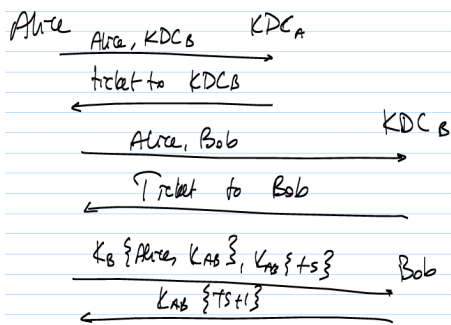
- ▶ A single KDC would be
  - ▶ Performance bottleneck
  - ▶ Single point of failure
- ▶ Have multiple replicas of the KDC with the database and the masterkey
- ▶ Any replica can serve as KDC for authentication
- ▶ Only one KDC (the master copy) handles the additions and deletions of principals (for consistency)

13

## Kerberos Realms

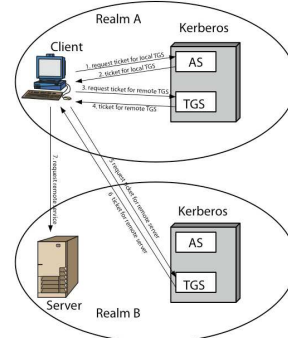
- ▶ A Kerberos environment consists of:
  - ▶ A Kerberos server
  - ▶ A number of clients, all registered with server
  - ▶ Application servers, sharing keys with server
- ▶ This is termed a realm
  - ▶ Typically a single administrative domain
- ▶ If have multiple realms, their Kerberos servers must share keys and trust

## Kerberos Realms



15

## Kerberos Realms



## Encryption for privacy and integrity

- ▶ Recall that one crypto operation is not enough to ensure privacy and integrity together
- ▶ Kerberos uses modified CBC to help integrity along with privacy – PCBC

$$c_{n+1} = m_{n+1} \oplus c_n \oplus m_n$$

← EXTEND

- ▶ Ensures that if  $c_i$  is tampered, rest of the blocks will be garbled
- ▶ For integrity
  - ▶ Append a recognizable text to end of the msg if any tampering of  $c_i$ s, then the recognizable text will be garbled as well

17

## Kerberos Version 5

- ▶ Developed in mid 1990's as IETF 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
  - ▶ Has public key extensions (e.g., SESAME, Win2000)

## Public Key Infrastructure (PKI)

### PKI

- ▶ A system to securely distribute and manage public keys
- ▶ Important for trust management for wide area network applications, e.g., for e-commerce
- ▶ A PKI consists of
  - ▶ Certificates
    - ▶ [I vouch that xyz is Bob's public key]<sub>sign</sub>
  - ▶ Certification authority (CA)
  - ▶ Certification repositories
  - ▶ A certificate revocation mechanism
- ▶ If have a certificate, a private key, and public key of CA, can authenticate
- ▶ Many design models are possible
  - ▶ Monopoly, oligarchy, anarchy, etc.

20

### Monopoly model

- ▶ A single organization is the CA for everyone
- ▶ Shortcomings
  - ▶ No such universally trusted organization
  - ▶ Requires everyone to authenticate physically with the same CA
  - ▶ Recovery from compromise is difficult
    - ▶ Due to single embedded public key of CA on all products
  - ▶ Once established, the CA can abuse its position
  - ▶ Requires perfect security at the CA
  - ▶ Getting certificate from a remote CA is vulnerable to attacks and/or is expensive

21

### Monopoly with registration authorities

- ▶ CA trusts other organizations (RAs) to check identities and do the initial authentication
- ▶ RA helps get the certificate for you from the CA
- ▶ Solves the problem of obtaining certificates in first place, other problems remain
  - ▶ Still monopoly pricing
  - ▶ Still cannot change CA
- ▶ RA can be incorporated in other models, too

22

### Delegated CAs

- ▶ Root CA certifies lower-level CAs to certify others
- ▶ All verifiers trust the root CA and verify certificate chains beginning at the root
  - ▶ i.e., the root CA is the trust anchor of all verifiers
- ▶ Example: A national PKI where a root CA certifies institutions, ISPs, universities who in turn certify their members
- ▶ Limitations are similar to monopoly with RAs

23

### Oligarchy model

- ▶ Many root CAs exist trusted by verifiers
- ▶ The model of web security
- ▶ Solves the problem of single authority – e.g., excessive pricing
- ▶ Disadvantages
  - ▶ N security sensitive sites instead of one
    - ▶ Compromise of any may put the entire system into a risk
  - ▶ Users can be tricked into trusting fake CAs (depending on implementation)
  - ▶ Users cannot tell if trust anchors in use are good or bad

24

## Anarchy model

- ▶ Users decide on whom to trust and how to authenticate their public keys
- ▶ Certificates issued by arbitrary parties can be stored in public databases which can be searched to find a path of trust to a desired party
- ▶ Not scalable as public database grows significantly
  - ▶ Works well for informal non-sensitive applications, e.g., PGP

25

## Other models

- ▶ Top-down w/ name constraints
  - ▶ The root is known to everyone
  - ▶ Root CA delegates other CAs each of which is responsible for its own name space
  - ▶ Searching a path to a name is easy – from root downward
- ▶ Bottom-up with name constraints
  - ▶ The model is very similar to how DNS service is organized and works
  - ▶ Each organization has its CA responsible for names in its organization
  - ▶ Each parent-child in the hierarchy certifies each other
  - ▶ Uses cross-certificates among remote nodes

26

## Revocation

- ▶ Mechanisms to cancel certificates compromised before expiration
- ▶ **Certificate Revocation List (CRL)** – list of revoked certificates published periodically by the CA
- ▶ **Delta CRLs** – only the changes since the previous issue are published
- ▶ **Online Revocation Servers**
  - ▶ No CRL is published
  - ▶ Verifier queries a central server to check if a certificate has been revoked

27

## Bad list vs good list

- ▶ **Bad List** – keep a list of revoked certificates
  - ▶ If a bogus certificate is issued to someone w/o keeping a log of it, no one would know its existence
- ▶ **Good List** – keeping a list of valid certificates
  - ▶ Bogus certificates would not be honored
  - ▶ But would be a too large and dynamic list

28

## Directories and PKI

- ▶ **Directory** – distributed hierarchical database indexed by a hierarchical name
  - ▶ Each name indicates a node in the tree
  - ▶ Each node is a record storing info about the name
  - ▶ Info – includes IP address, certificates signed by this or by other principals
- ▶ DNS can be seen as a directory service with some very limited functionality – lookup only
- ▶ X.500 – a directory standard w/ limited deployment and use
- ▶ Directories could be used to facilitate certification management in PKIs

29

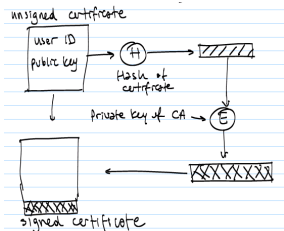
## Where to store certificates?

- ▶ **With subject or issuer or both?**
  - ▶ PKIX – must store w/ subject and may store w/ issues as well
- ▶ **Storing w/ subject**
  - ▶ Issuer may not have write access to subject's record
  - ▶ For a root CA w/ many children, more convenient for down-certificates from the CA to be stored in subjects' records
- ▶ **Storing w/ issuer**
  - ▶ If a key is compromised, the principal needs to inform everyone certifying his key → how to know cross-certifiers to inform them?
  - ▶ Helps find a path toward a target name from trust anchor

30

## X.509 and PKIX

- ▶ X.509 – a format to define certificates
  - ▶ Each certificate contains public key of a user and is signed w/ the private key of a trusted CA
  - ▶ Used in S/MIME, IPsec, SSL/TLS, etc.



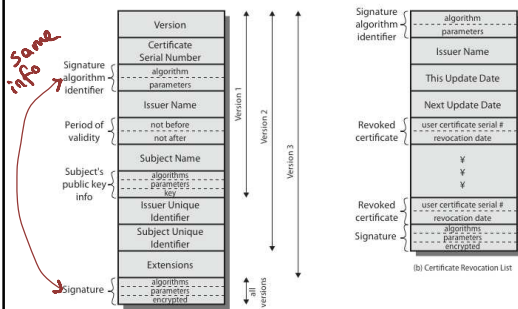
- ▶ PKIX – a profile for X.509, specifying which options should be supported

31

## X.509 authentication service

- ▶ Part of CCITT X.500 directory service standards
  - ▶ Distributed servers maintaining user 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 are widely used

## X.509 Certificates



## 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 CA's in hierarchy

## 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 maintains list of revoked certificates
  - ▶ The Certificate Revocation List (CRL)
- ▶ Users should check certificates with CA's CRL

## PKIX

- ▶ IETF proposed X.509 based model to deploy PKI
- ▶ Identifies several management functions to be supported
  - ▶ Registration, initialization, certification, key pair recovery, key pair update, revocation request, cross certification
- ▶ Accompanied w/ two management protocols to support above function – CMP and CMC

37

## PKIX

