"Cryptography and Network Security", by William Stallings

Chapter 13- Digital Signatures and Authentication Protocols

(Lecture slides by Lawrie Brown)

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

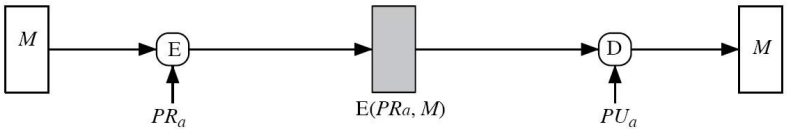
Digital Signature Properties

- must depend on the message signed
- must use information unique to sender
  - to prevent both forgery and denial
- must be relatively easy to produce
- must be relatively easy to recognize & verify
- be computationally infeasible to forge
  - with new message for existing digital signature
  - with fraudulent digital signature for given message
- be practical save digital signature in storage

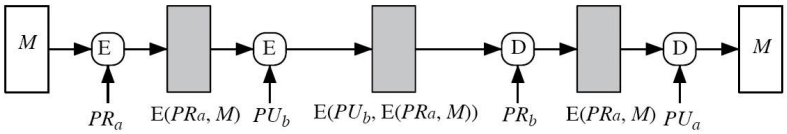
Direct Digital Signatures

- involve only sender & receiver
- assumed receiver has sender's public-key
- digital signature made by sender signing entire message or hash with private-key
- can encrypt using receivers public-key
- important that sign first then encrypt message & signature

Direct Digital Signatures



(c) Public-key encryption: authentication and signature



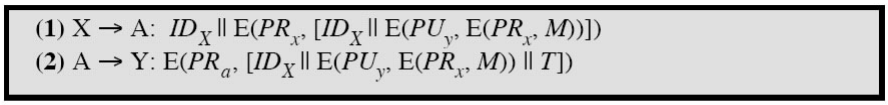
(d) Public-key encryption: confidentiality, authentication, and signature

Weakness: Security depends on sender's private-key

Arbitrated Digital Signatures

- involves use of arbiter A
  - validates any signed message
  - then dated and sent to recipient
- requires suitable level of trust in arbiter
- can be implemented with either private or public-key algorithms
- arbiter may or may not see message

Arbitrated Digital Signatures



(c) Public-Key Encryption, Arbiter Does Not See Message

Notations:

X=sender	M=message
Y=recipient	T=time stamp
A=Arbiter	PR_X=X's private key
ID_X=ID of X	PU_Y=Y's public key
	PR_A=A's private key

Weakness: twice public-key encryptions on the message

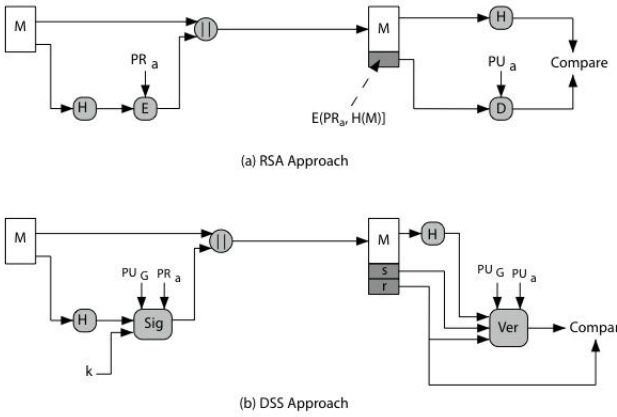
Digital Signature Standard (DSS)

- US Govt approved signature scheme
- designed by NIST & NSA in early 90's
- published as FIPS-186 in 1991
- revised in 1993, 1996 & then 2000
- uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
- FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants

Digital Signature Algorithm (DSA)

- creates a 320 bit signature
- with 512-1024 bit security
- smaller and faster than RSA
- a digital signature scheme only
- security depends on difficulty of computing discrete logarithms
- variant of ElGamal & Schnorr schemes

Digital Signature Algorithm (DSA)



DSA Key Generation

- have shared global public key values (p,q,g):
  - choose a large prime p: $2^{L-1} < P < 2^{L}$ and q where L= 512 to 1024 bits and is a multiple of 64
    - and q is a prime factor of $(p-1)$: $2^{159} < q < 2^{160}$
  - commute $g = h^{(p-1)/q} \bmod p$
    - where $h<p-1$, $h^{(p-1)/q} \pmod p > 1$
- users choose private & compute public key:
  - choose $x<q$
  - compute $y = g^{x} \pmod p$

DSA Signature Creation

- to **sign** a message M the sender:
  - generates a random signature key k, $k<q$
  - nb. k must be random, be destroyed after use, and never be reused
- then computes signature pair:

  $r = (g^{k} \pmod p) \pmod q$

  $s = (k^{-1}.H(M) + x.r) \pmod q$
- sends signature (r,s) with message M

DSA Signature Verification

- having received M & signature (r,s)
- to **verify** a signature, recipient computes:

$w = s^{-1} \pmod q$

$u1 = (H(M).w) \pmod q$

$u2 = (r.w) \pmod q$

$v = (g^{u1}.y^{u2} \pmod p) \pmod q$

- if v=r then signature is verified
- see book web site for details of proof why

Authentication Protocols

- used to convince parties of each others identity and to exchange session keys
- one-way or mutual authentication
- key issues are
  - confidentiality – to protect session keys
  - timeliness – to prevent replay attacks

Replay Attacks

- where a valid signed message is copied and later resent
  - simple replay
  - repetition that can be logged
  - repetition that cannot be detected
  - backward replay without modification
- countermeasures include
  - use of sequence numbers (generally impractical)
  - timestamps (needs synchronized clocks)
  - challenge/response (using unique nonce)

Mutual Authentication

- Sender and receiver are to be online at the same time.
- There are symmetric encryption based and public-key encryption-base approaches (protocols)

Symmetric Encryption Based Authentication Protocols

- as discussed previously can use a two-level hierarchy of keys
- usually with a trusted Key Distribution Center (KDC)
  - each party shares own master key with KDC
  - KDC generates session keys used for connections between parties
  - master keys used to distribute these to them

Needham-Schroeder Protocol

- original third-party key distribution protocol
- for session between A B mediated by KDC
- protocol overview is:
  1. A->KDC: ID_A || ID_B || N_1
  2. KDC -> A: E_{Ka}[Ks || ID_B || N_1 || E_{Kb}[Ks||ID_A]]
  3. A -> B: E_{Kb}[Ks||ID_A]
  4. B -> A: E_{Ks}[N_2]
  5. A -> B: E_{Ks}[f(N_2)]

Needham-Schroeder Protocol

- used to securely distribute a new session key for communications between A & B
- but is vulnerable to a replay attack if an old session key has been compromised
  - then message 3 can be resent convincing B that is communicating with A
- modifications to address this require:
  - timestamps (Denning 81)
  - using an extra nonce (Neuman 93)

Public-Key Encryption Based Authentication Protocols

- have a range of approaches based on the use of public-key encryption
- need to ensure have correct public keys for other parties
- using a central Authentication Server (AS)
- various protocols exist using timestamps or nonces

Denning AS Protocol

- Denning 81 presented the following:
  - **1.** A -> AS: ID_A || ID_B
  - **2.** AS -> A: $E_{PRas}[ID_A||PU_a||T]$ || $E_{PRas}[ID_B||PU_b||T]$
  - **3.** A -> B: $E_{PRas}[ID_A||PU_a||T]$ || $E_{PRas}[ID_B||PU_b||T]$ || $E_{PUb}[E_{PRas}[K_s||T]]$
- note session key is chosen by A, hence AS need not be trusted to protect it
- timestamps prevent replay but require synchronized clocks

One-Way Authentication

- required when sender & receiver are not in communications at same time (eg. email)
- have header in clear so can be delivered by email system
- may want contents of body protected & sender authenticated

Using Symmetric Encryption

- can refine use of KDC but can't have final exchange of nonces, vis:
  - **1.** A->KDC: ID_A || ID_B || N_1
  - **2**. KDC -> A: E_{Ka}[Ks || ID_B || N_1 || E_{Kb}[Ks||ID_A]]
  - **3.** A -> B: E_{Kb}[Ks||ID_A] || E_{Ks}[M]
- does not protect against replays
  - could rely on timestamp in message, though email delays make this problematic

Public-Key Approaches

- have seen some public-key approaches
- if confidentiality is major concern, can use:

  A->B: E_{PUb}[Ks] || E_{Ks}[M]
  - has encrypted session key, encrypted message
- if authentication needed use a digital signature with a digital certificate:

  A->B: M || E_{PRa}[H(M)] || E_{PRas}[T||ID_A||PU_a]
  - with message, signature, certificate

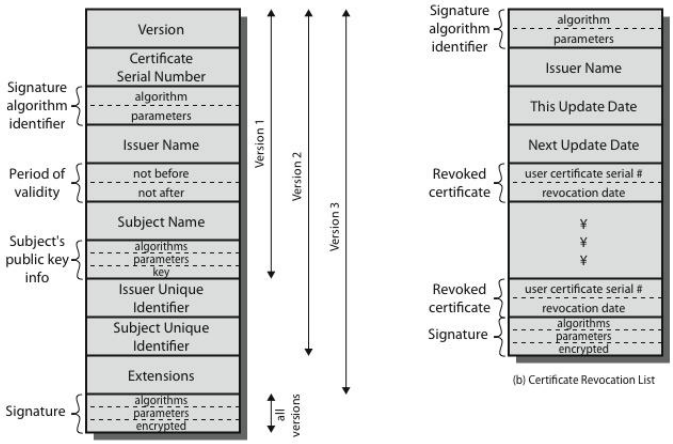
X.509 Authentication Service

- 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

- 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



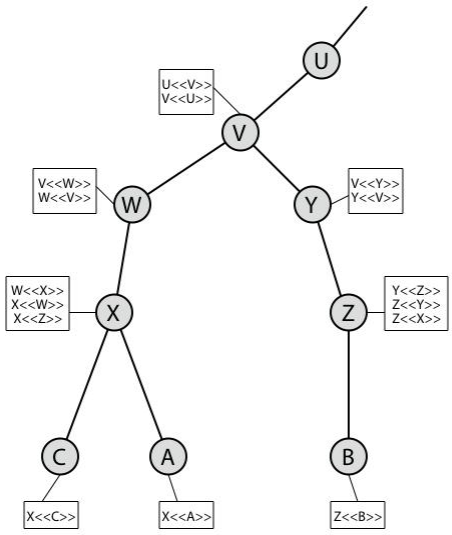
Obtaining a Certificate

- 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



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 certificates with CA's CRL

Authentication Procedures

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

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
- may include additional info for B
  - eg session key

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
- may include additional info for A

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

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

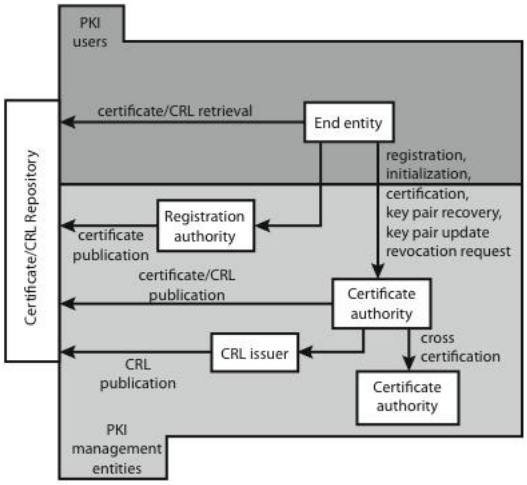
X.509 Version 3

- 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:
  - extension identifier
  - criticality indicator
  - extension value

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

Public Key Infrastructure



30.4 Key Management

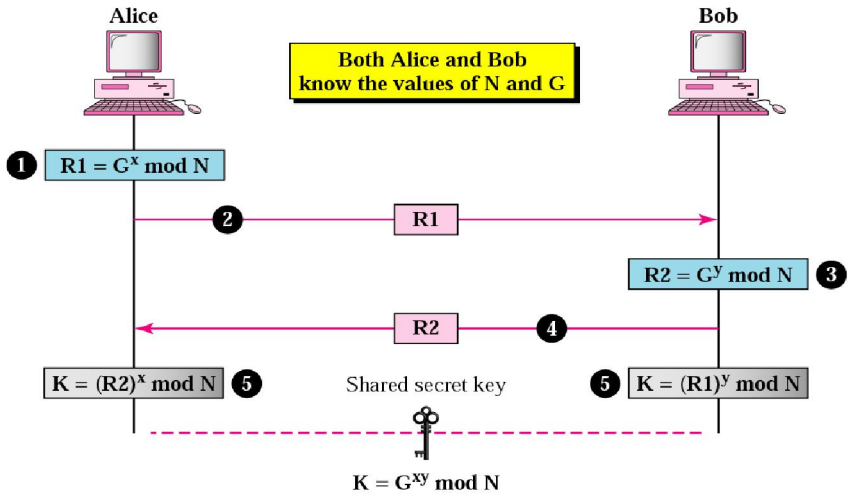
Symmetric-Key Distribution

Public-Key Certification

Note:

A symmetric key between two parties is useful if it is used only once; it must be created for one session and destroyed when the session is over.

Figure 30.11 Diffie-Hellman method



The symmetric (shared) key in the Diffie-Hellman protocol is
$K = G^{xy} \bmod N$.

Example 2

Assume G = 7 and N = 23. The steps are as follows:

1. Alice chooses x = 3 and calculates R1 = $7^3 \bmod 23 = 21$.
2. Alice sends the number 21 to Bob.
3. Bob chooses y = 6 and calculates R2 = $7^6 \bmod 23 = 4$.
4. Bob sends the number 4 to Alice.
5. Alice calculates the symmetric key K = $4^3 \bmod 23 = 18$.
6. Bob calculates the symmetric key K = $21^6 \bmod 23 = 18$.

The value of K is the same for both Alice and Bob;
$G^{xy} \bmod N = 7^{18} \bmod 23 = 18$.

Figure 30.12 Man-in-the-middle attack

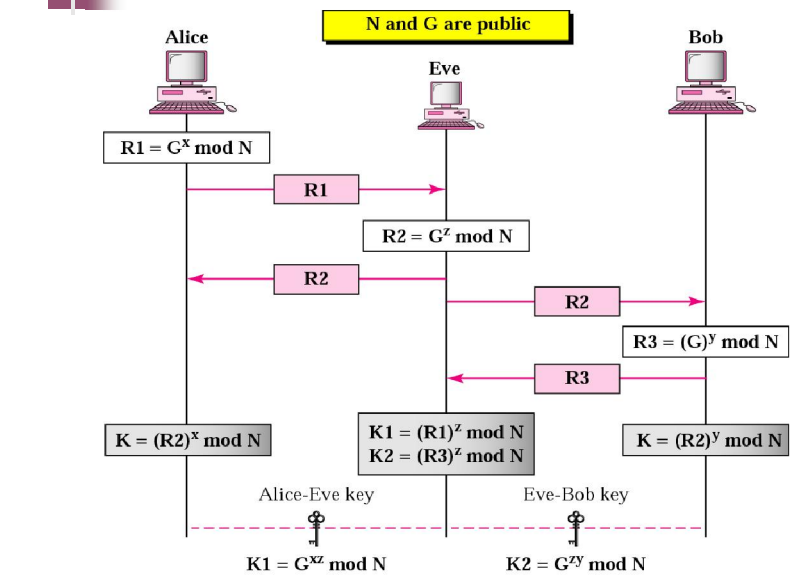


Figure 30.13 First approach using KDC

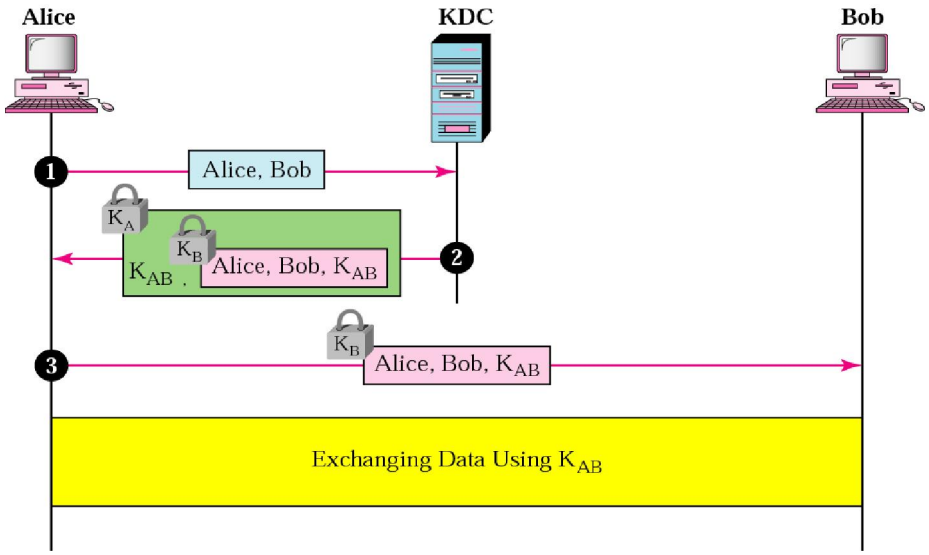


Figure 30.14 Needham-Schroeder protocol

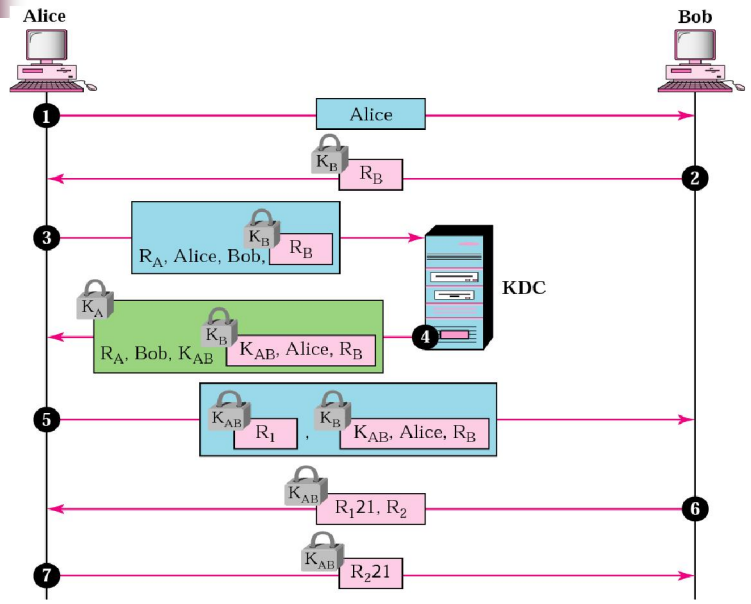
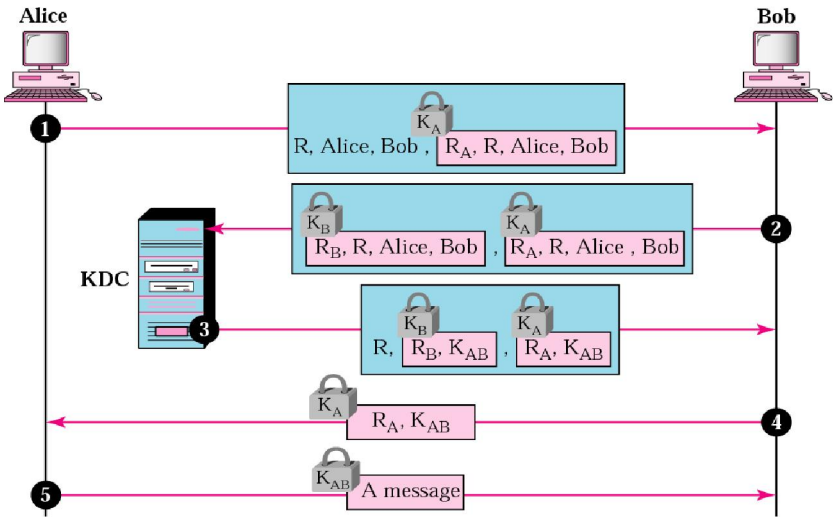


Figure 30.15 Otway-Rees protocol

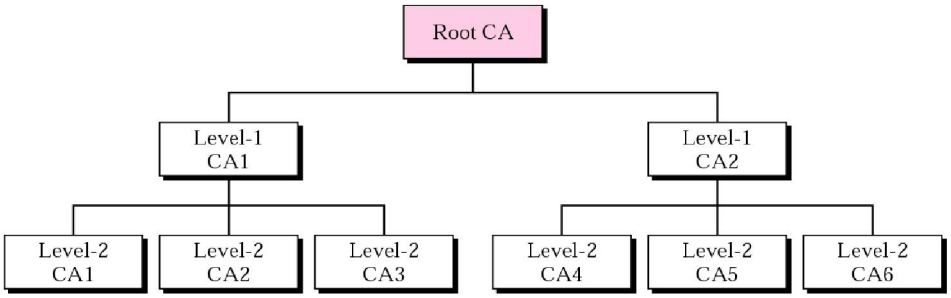


In public-key cryptography, everyone has access to everyone's public key.

Table 30.1 X.500 fields

Field	*Explanation*
Version	Version number of X.509
Serial number	The unique identifier used by the CA
Signature	The certificate signature
Issuer	The name of the CA defined by X.509
Validity period	Start and end period that certificate is valid
Subject name	The entity whose public key is being certified
Public key	The subject public key and the algorithms that use it

Figure 30.16 PKI hierarchy



30.5 Kerberos

Servers

Operation

Using Different Servers

Version 5

Realms

Figure 30.17 Kerberos servers

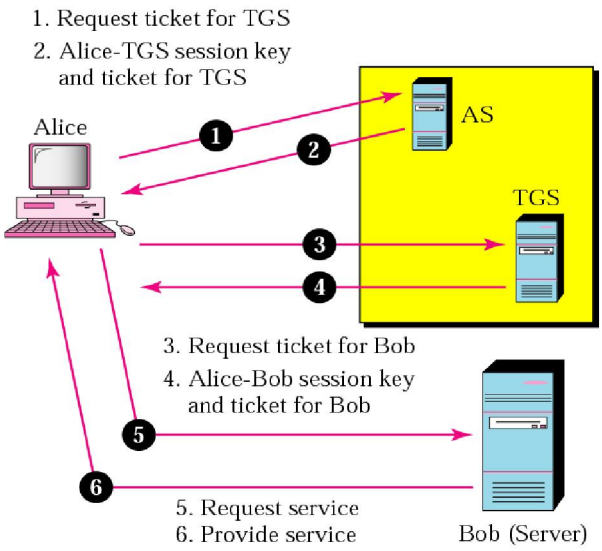


Figure 30.18 *Kerberos example*