Introduction to Computer Security

Chapter 23: Internet Authentication Applications

Chi-Yu Li (2019 Spring)
Computer Science Department
National Chiao Tung University

Outline

- Kerberos
- X.509
- Public-Key Infrastructure

Kerberos

- Initially developed at MIT
  - A software utility available both in the public domain and in commercially supported versions
  - The defacto standard for remote authentication

- A trusted third party authentication service
  - Clients and servers trust Kerberos to mediate their mutual authentication
  - Requires that
    - A user proves his or her identity for each service invoked
    - Optionally, servers prove their identity to clients

Security Issues between Clients and Servers?

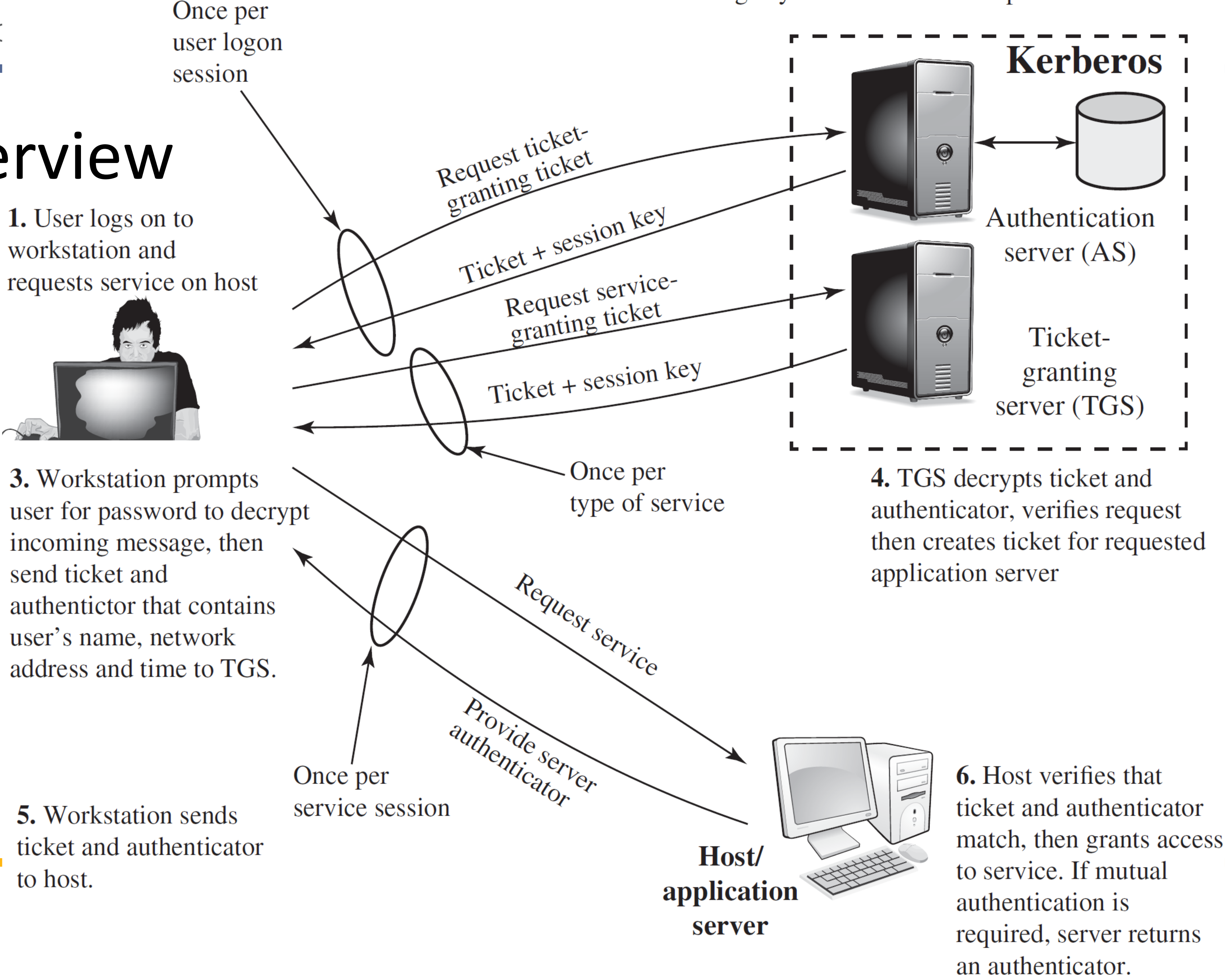
- In an unprotected network environment, any client can apply to any server for server
- Obvious security risk?
  - Impersonation: an opponent can pretend to be another client and obtain unauthorized privileges on server machines
- How do servers counter this threat?
  - Confirm the identities of clients
  - But, each server is required to do this for each client/server interaction

Security Issues between Clients and Servers? (Cont.)

- Alternative: using an authentication server (AS)
  - Once the AS has verified the user's identity, it can pass this information on to an application server
- How to do all this in a secure way?

Kerberos Overview

- AS shares a unique secret key with each server
- Session key: one-time encryption key
- Ticket and session key are both encrypted using the user's password as the encryption key
- Password is never passed over the network

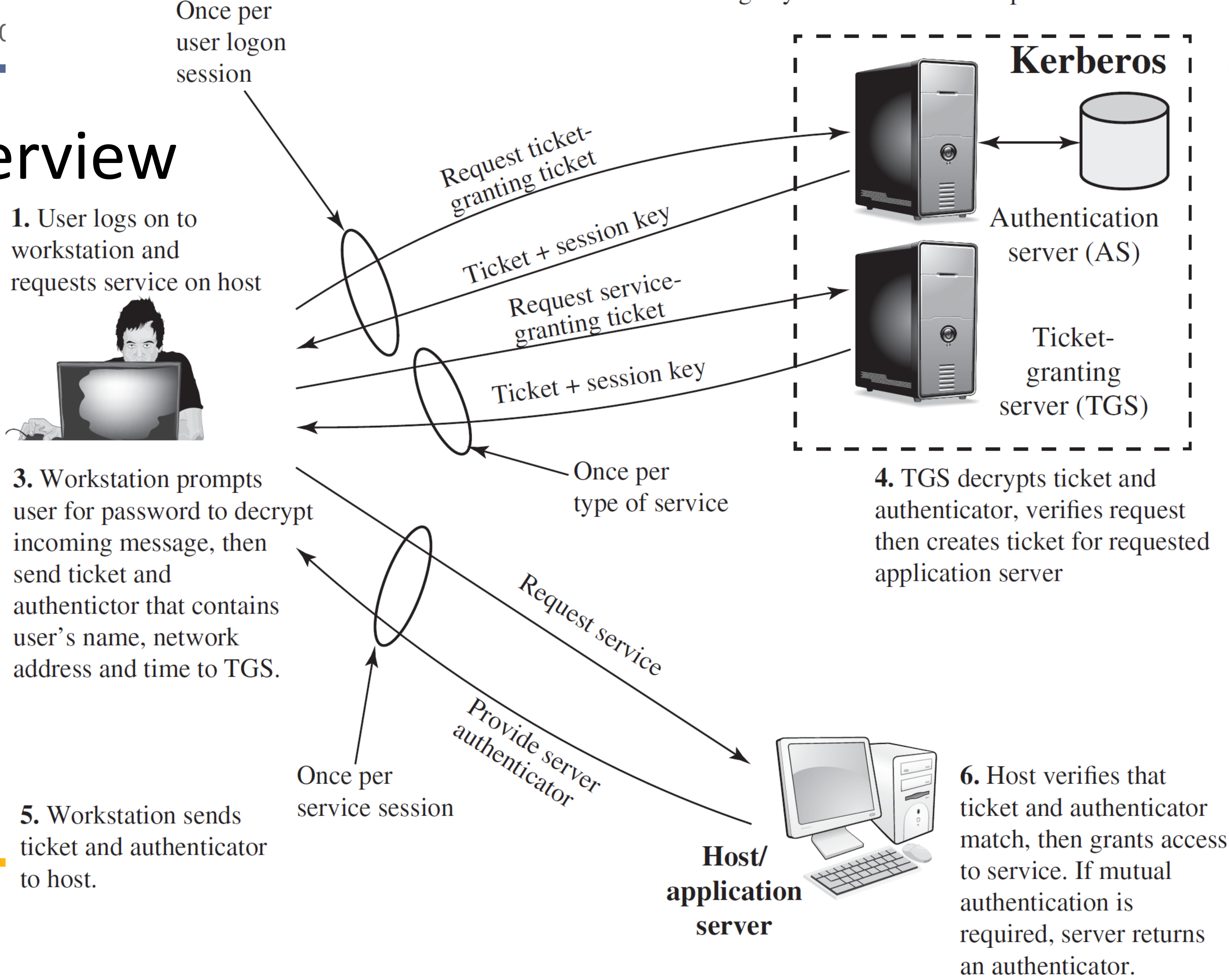


2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

Kerberos Overview

- Ticket: a set of credentials
  - ➢ User's ID, server's ID, a timestamp, a lifetime
- Entire ticket is encrypted using a secret DES key shared by the AS and the server
- Why TGS?
  - ➢ Query the user for his password for each service
    Inconvenient!
  - ➢ Store the password in memory for the duration of the logon session
    Security risk!

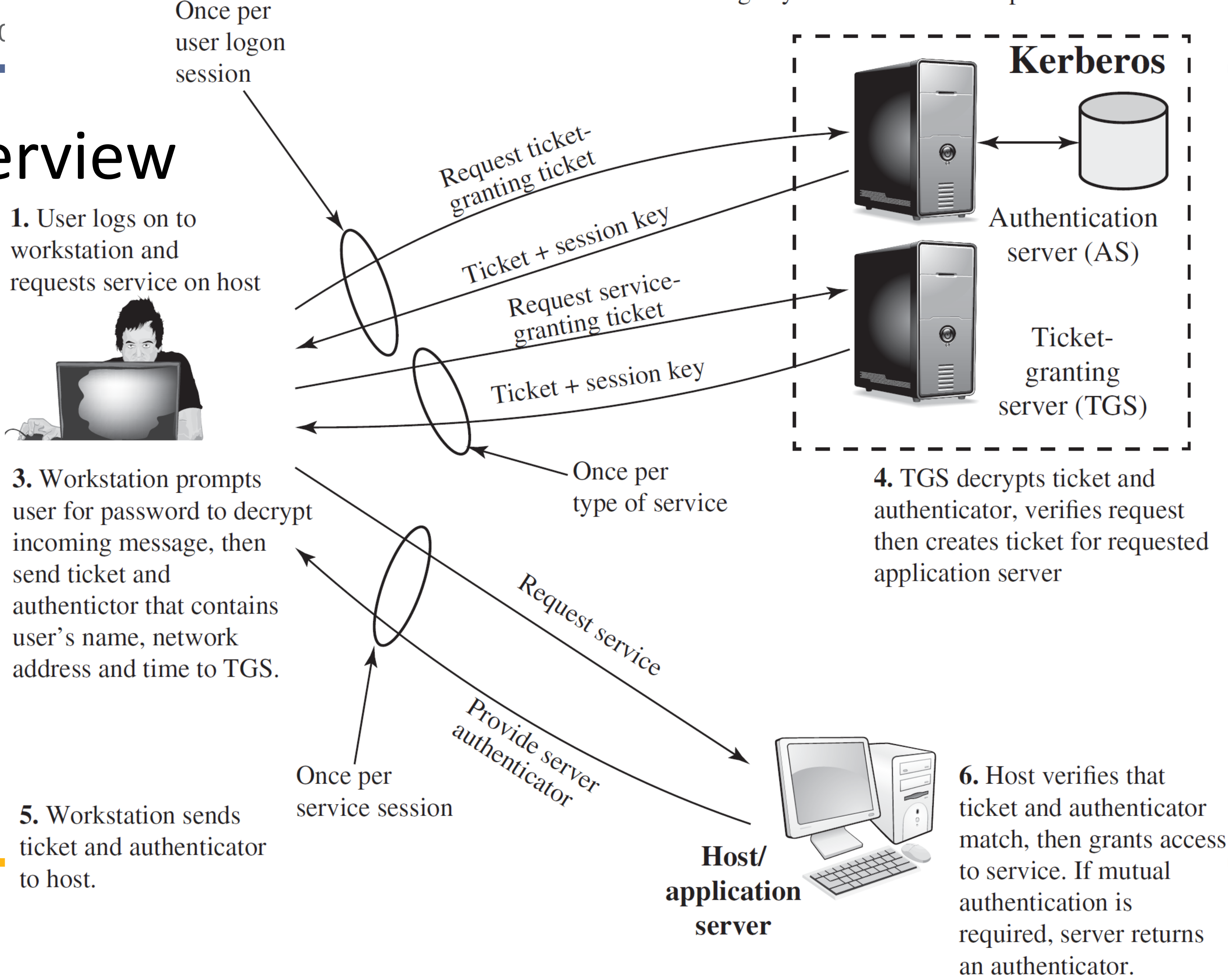
- TGS: A ticket to get more tickets!



Kerberos Overview

How to counter the following threats for ticket-granting ticket?

- Ticket may be stolen and reused
  - ➢ Timestamp
- Alteration of the ticket
  - ➢ Encrypted with a secret key known only to the AS and TGS
- User spoofing
  - ➢ Authentication based on the encryption with the user's password
- Replay attack
  - ➢ Authenticator, which is not reusable

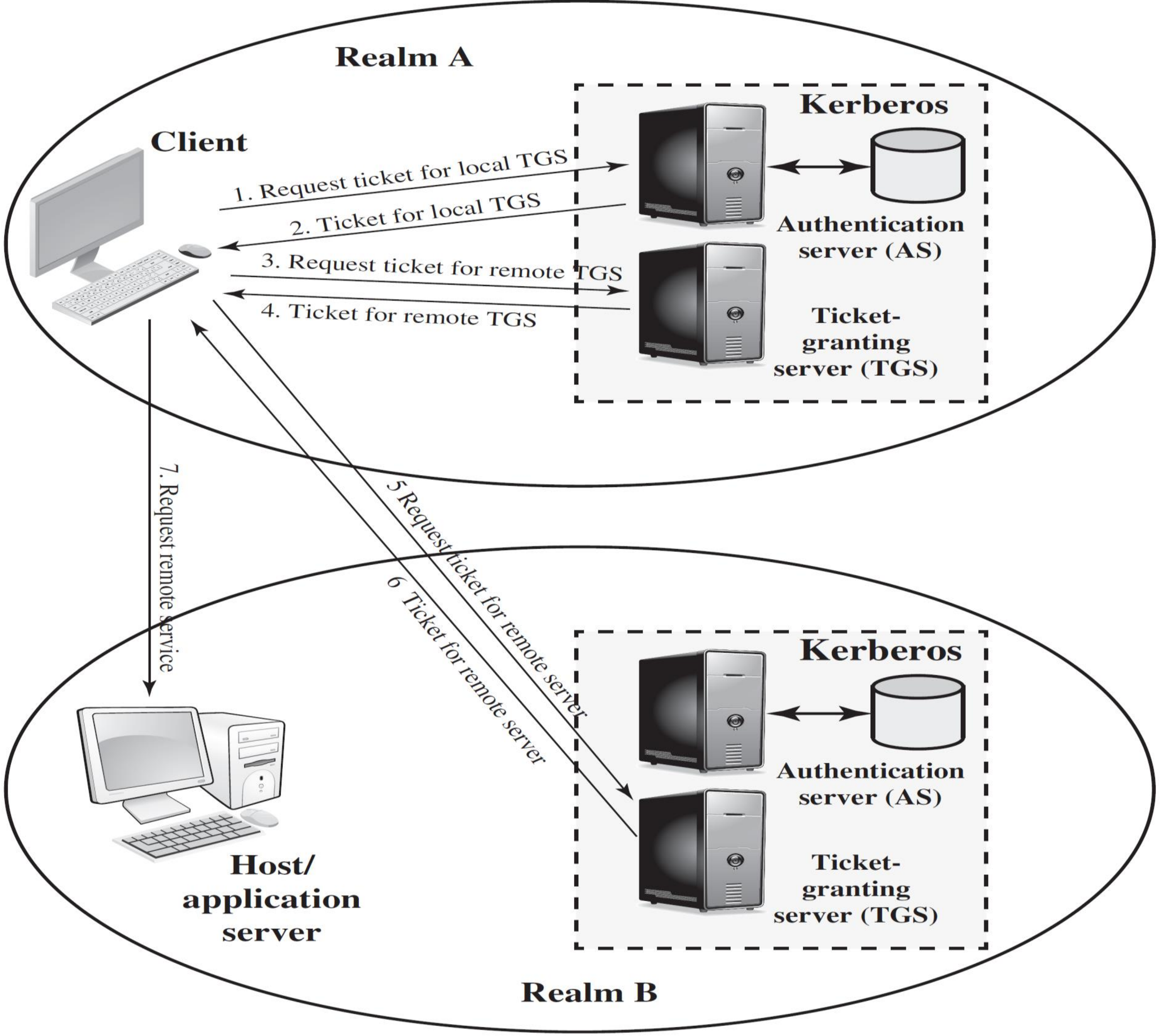


Kerberos Realms

- A Kerberos realm: full-service Kerberos environment
  - A Kerberos server
  - A number of clients
    - Each registers with the Kerberos server; the server holds a database for the user ID/password
  - A number of app servers
    - Each registers with the Kerberos server and shares a secret key with it
- Different realms
  - Networks of clients and server under different administrative organizations

Interrealm Authentication

- The Kerberos server in each realm shares a secret key with the server in the other realm
- Kerberos servers are registered with each other



Kerberos Versions 4 and 5

- Most widely used: Version 4 in late 1980s
- Version 5
  - Introduced in 1993; updated in 2005
  - Now widely implemented
    - Part of Microsoft's Active Directory service
    - Most current UNIX and Linux systems
    - Apple's Mac OS X
  - AES is the default choice (DES in v4)
  - Authentication forwarding
    - Enabling a client to access a server and have that server access another sever on behalf of the client

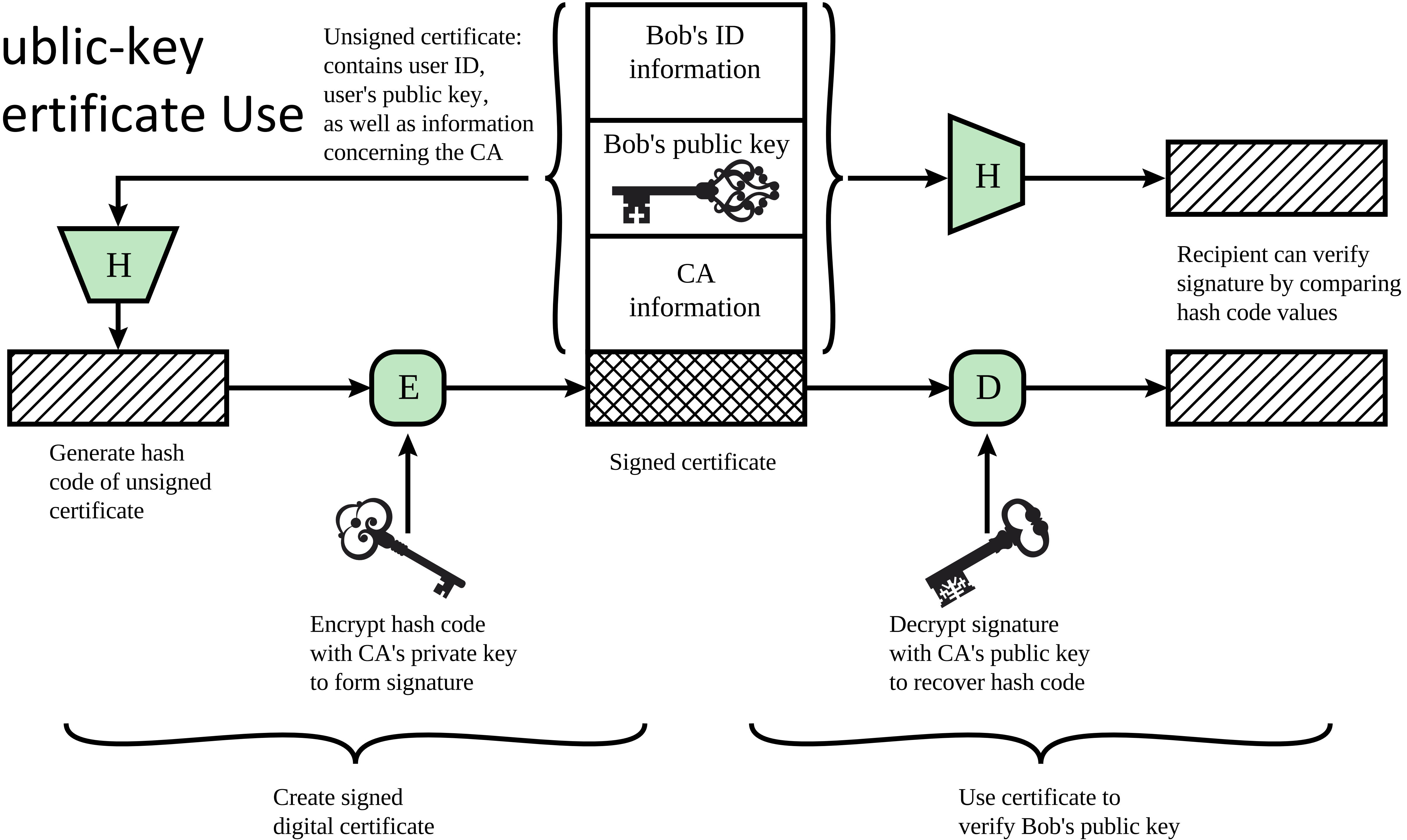
Performance Issues

- Very little performance impact in a large-scale environment
  - if the system is properly configured
  - The amount of traffic needed for the granting ticket: modest
- Does it require a dedicated platform?
  - Not wise to run it on the same machine as a resource-intensive app
  - Its security is best assured by placing it on a separate, isolated machine
- How about using multiple realms to maintain performance?
  - Probably not
  - The motivation of multiple realms is administrative

X.509: Public-key Certificate

- A certificate
  - linking a public key with the identity of the key's owner
  - the whole block signed by a trusted third party
- Third party: certificate authority (CA)
  - trusted by the user community
  - e.g., government agency, financial institution
- User can present his public key to the authority in a secure manner and obtain a certificate

Public-key Certificate Use

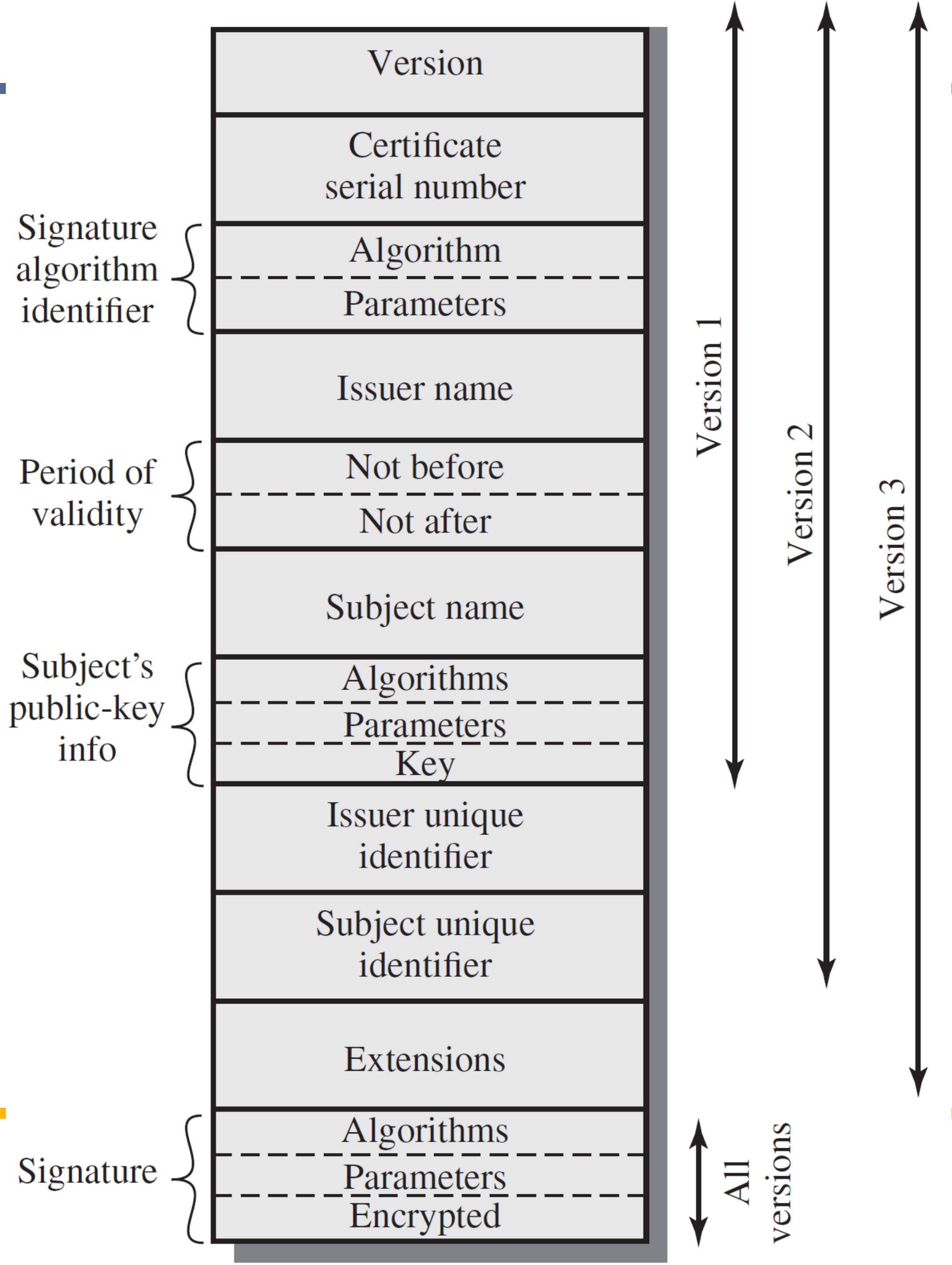


X.509

- Specified in RFC 5280
- Most widely accepted format for public-key certificates
- Used in most network security apps
  - IPSec, SSL, TLS, S/MIME, etc.

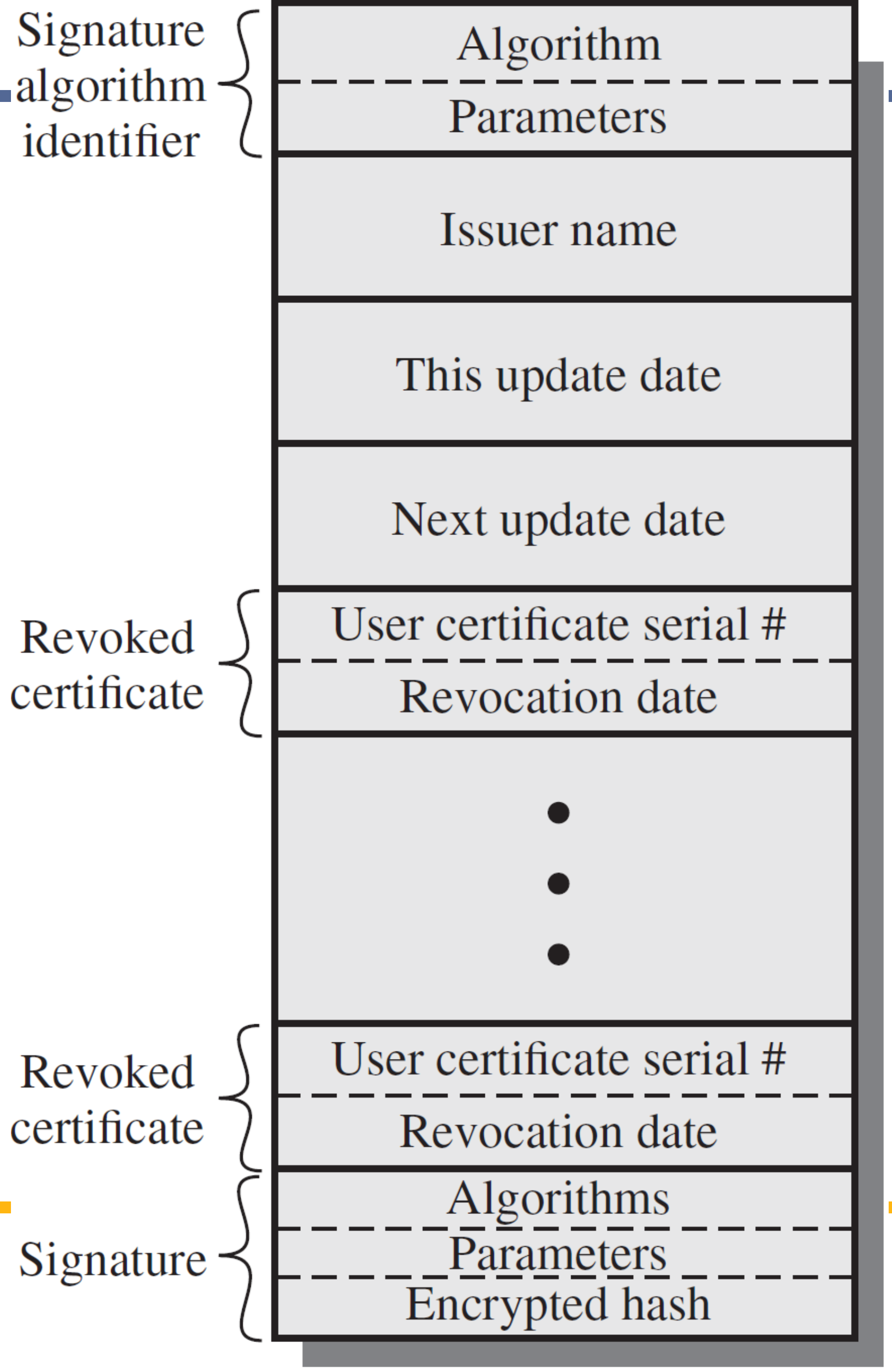
- Usage restriction
  - “Key Usage” and “Extended Key Usage” extensions specify a set of approved uses

- What if the certificate has been compromised? How to revoke it?



X.509 (Cont.)

- A certificate revocation list (CRL)
  - signed by the issuer
  - containing a serial number of a certificate and the revocation date
- When receiving a certificate, an app should determine whether it has been revoked
  - checking the current CRL for its issuing CA
  - very few apps do this due to high overheads
- The recent Heartbleed Open SSL bug has dramatically highlighted deficiencies with the use of CRLs



X.509 (Cont.)

- A lightweight protocol for the revocation in RFC 6960
  - including in recent versions of most common Web browsers
  - "Authority Information Access" extension in a certificate: specify the server access
- Hash signature
  - MD5 collisions are found in 2012: depreciated
  - SHA-a collisions are discovered in 2017
  - Using SHA-2 and SHA-3 now

Public-Key Infrastructure (PKI)

- RFC 4949: the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography
- Principal objective: enable secure, convenient, and efficient acquisition of public keys
- Current X.509 PKI implementations: trust store
    - A large list of CAs and their public keys

CAs in Trust Store

- Either directly sign "end-user" certificates
- Or sign a small number of Intermediate-CAs
  - They in turn sign "end-user" certificates
- All the hierarchies are very small, and all are equally trusted
- Automatically verified certificate: acquiring it from one of those CAs
  - Alternatively: use either a self-signed certificate or a certificate signed by some other CA
    - Such certificates are initially recognized as "untrusted"
    - The user presented with stark warnings about accepting such certificates

Issues with the PKI Model

- Issue 1: Reliance on the user to make an informed decision when there is a problem verifying a certificate

- Issue 2: Assuming that all of the CAs in the “trust store” are equally trusted, equally well managed, and apply equivalent policies
  - Compromise of the DigiNotar CA in 2011
    - Fraudulent issue of certificates for many well-known organizations
    - Bankrupt later that year
  - Compromise of the Comodo CA in 2011
    - A small number of fraudulent certificates issued
  - Iranian government: mount a “man-in-the-middle” attack
    - On the secured communications of many of their citizens

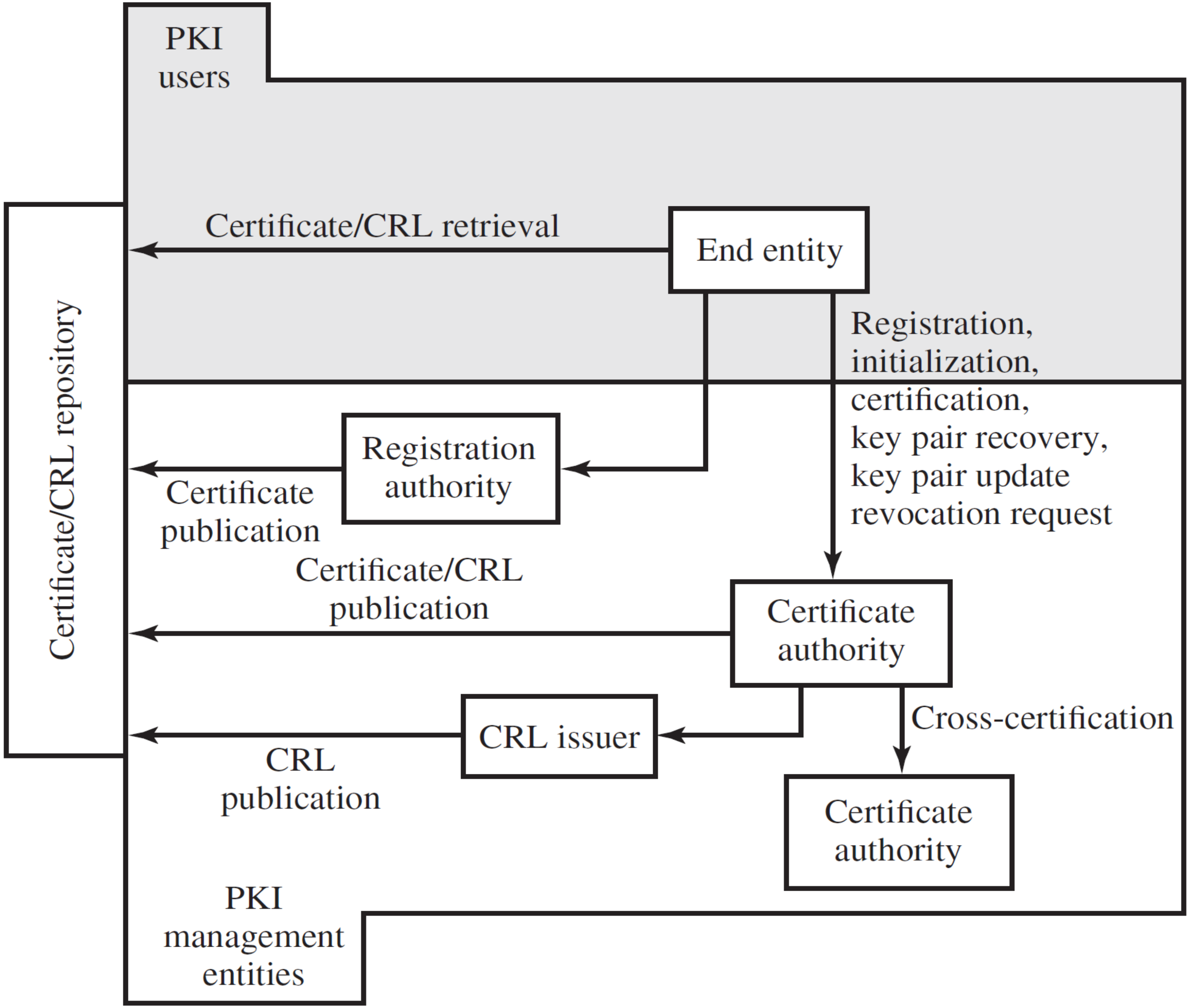
Issues with the PKI Model (Cont.)

- Issue 3: different implementations, in the various web browsers and OS, use different “trust stores,”
  - Present different security views to users

Improve the X.509 Certificates

- Recognize that many apps do not require formal linking of a public key to a verified identity
  - In many web apps, all users really need is to know that if they visit the same secure site
  - i.e., same site and same key as when they previously visited
- Improvement 1: confirming continuity in time
  - Apps keep a record of certificate details for all sites they visit
  - e.g., Google Chrome
- Improvement 2: confirming continuity in space
  - Using a number of widely separated "network notary servers" that keep records of certificates for all sites they view
  - e.g., Firefox "Perspectives" plugin; notary servers: Google Certificate Catalog

Public Key Infrastructure X.509 (PKIX)



Questions?