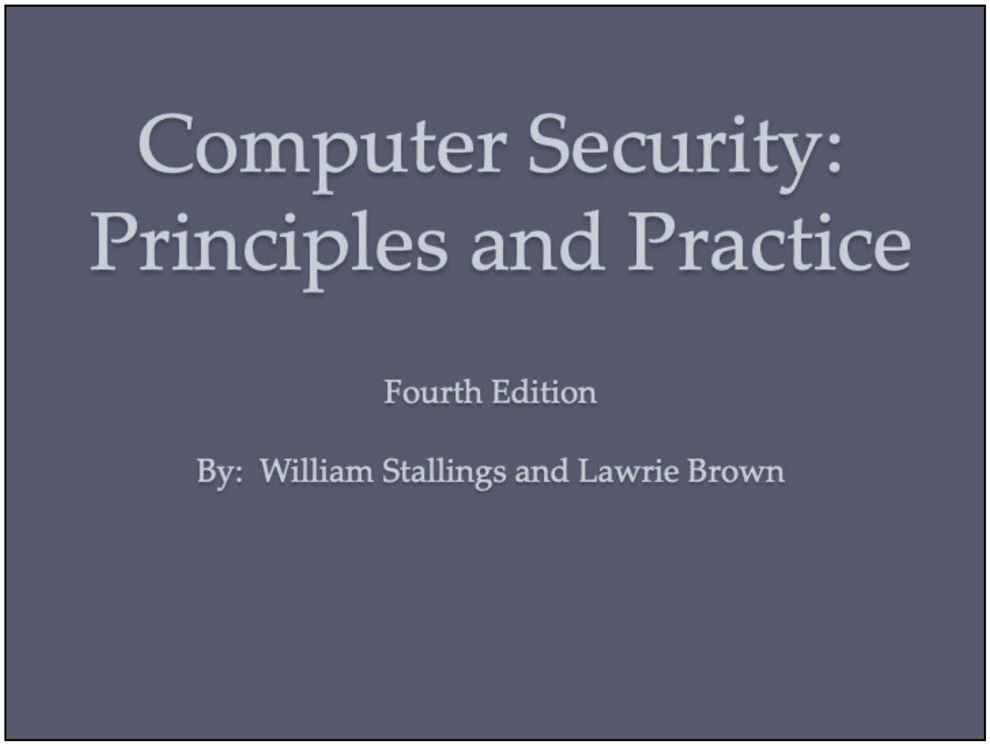
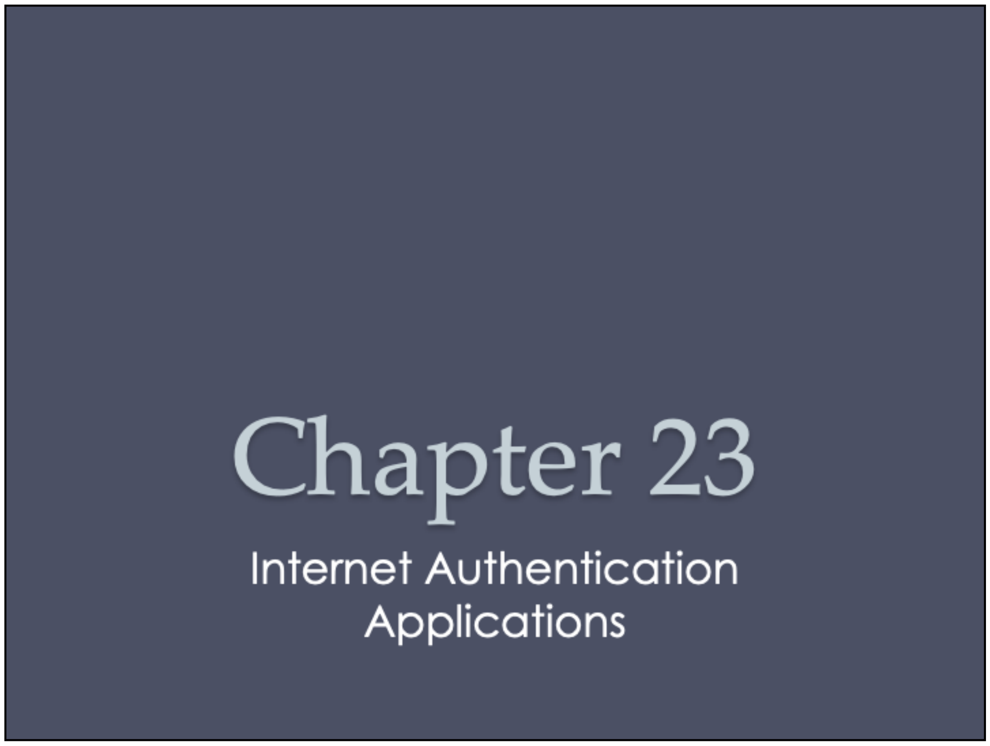


Computer Security: Principles and Practice

Fourth Edition

By: William Stallings and Lawrie Brown

Lecture slides prepared for “Computer Security: Principles and Practice”, 4/e, by William Stallings and Lawrie Brown, Chapter 23 “Internet Authentication Applications”.

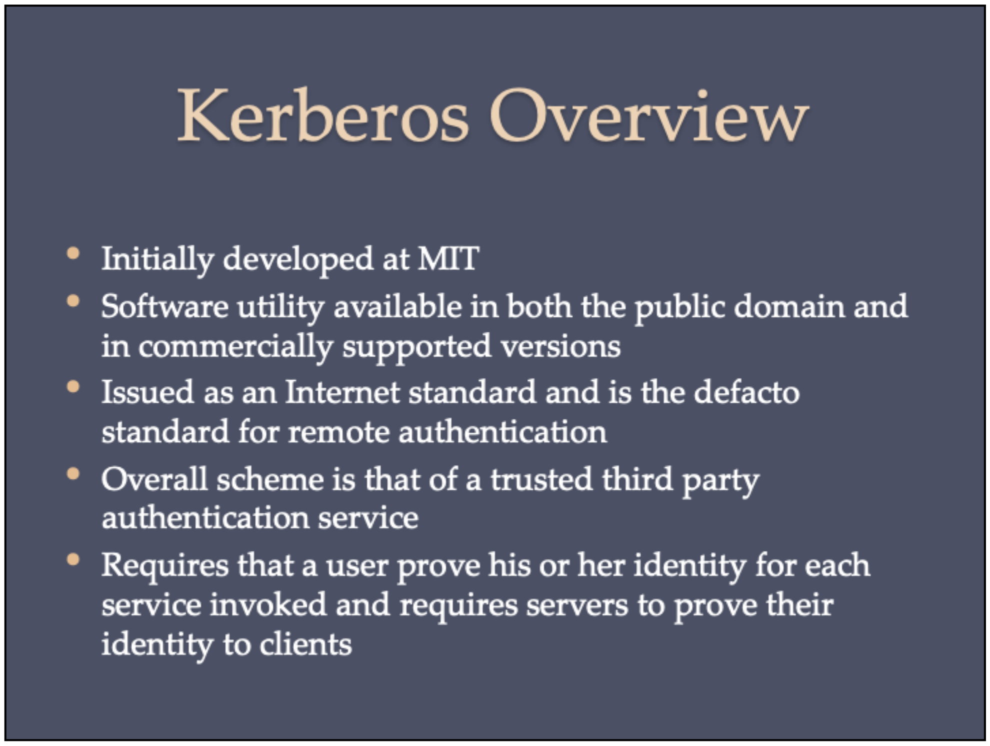


Chapter 23

Internet Authentication Applications

This chapter examines some of the authentication functions that have been developed to support network-based authentication and digital signatures.

We begin by looking at one of the earliest and also one of the most widely used services: Kerberos. Next, we examine the X.509 public-key certificates. Then, we examine the concept of a public-key infrastructure (PKI).



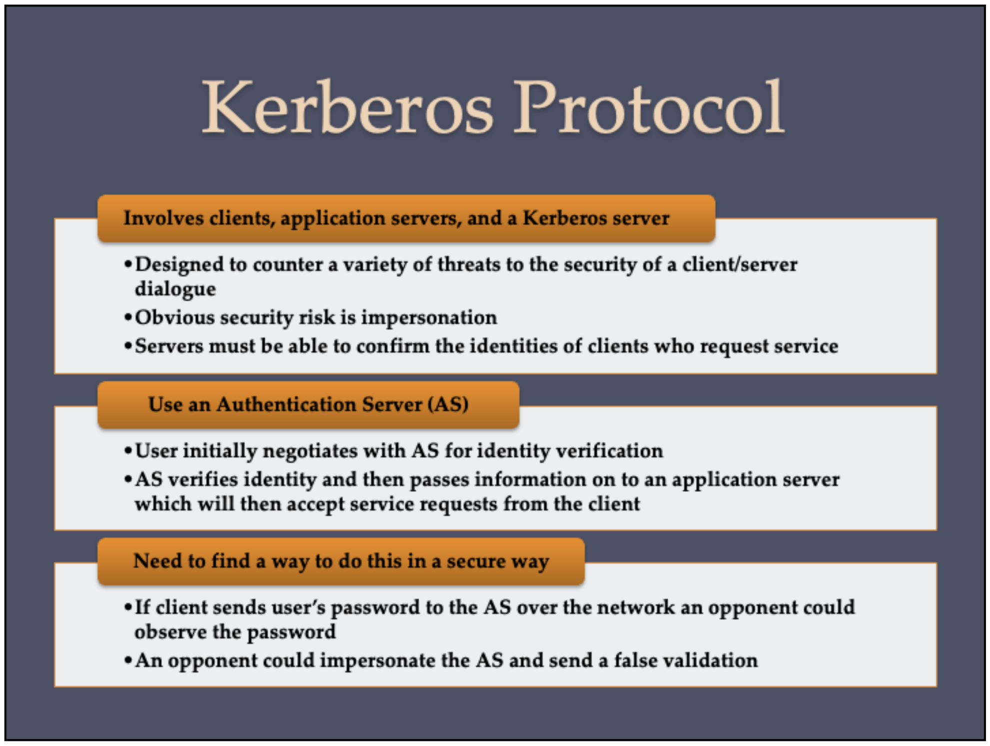
Kerberos Overview

- Initially developed at MIT
- Software utility available in both the public domain and in commercially supported versions
- Issued as an Internet standard and is the defacto standard for remote authentication
- Overall scheme is that of a trusted third party authentication service
- Requires that a user prove his or her identity for each service invoked and requires servers to prove their identity to clients

There are a number of approaches that organizations can use to secure networked servers and hosts. Systems that use one-time passwords thwart any attempt to guess or capture a user's password. These systems require special equipment such as smart cards or synchronized password generators to operate and have been slow to gain acceptance for general networking use. Another approach is the use of biometric systems. These are automated methods of verifying or recognizing identity on the basis some physiological characteristic, such as a fingerprint or iris pattern, or a behavioral characteristic, such as handwriting or keystroke patterns. Again, these systems require specialized equipment.

Another way to tackle the problem is the use of authentication software tied to a secure authentication server. This is the approach taken by Kerberos. Kerberos, initially developed at MIT, is a software utility available both in the public domain and in commercially supported versions. Kerberos has been issued as an Internet standard and is the de facto standard for remote authentication, including as part of Microsoft's Active Directory service.

The overall scheme of Kerberos is that of a trusted third-party authentication service. It is trusted in the sense that clients and servers trust Kerberos to mediate their mutual authentication. In essence, Kerberos requires that a user prove his or her identity for each service invoked and, optionally, requires servers to prove their identity to clients.



Kerberos Protocol

Involves clients, application servers, and a Kerberos server

- Designed to counter a variety of threats to the security of a client/server dialogue
- Obvious security risk is impersonation
- Servers must be able to confirm the identities of clients who request service

Use an Authentication Server (AS)

- User initially negotiates with AS for identity verification
- AS verifies identity and then passes information on to an application server which will then accept service requests from the client

Need to find a way to do this in a secure way

- If client sends user's password to the AS over the network an opponent could observe the password
- An opponent could impersonate the AS and send a false validation

Kerberos makes use of a protocol that involves clients, application servers, and a Kerberos server. That the protocol is complex reflects that fact that there are many ways for an opponent to penetrate security. Kerberos is designed to counter a variety of threats to the security of a client/server dialogue.

The basic idea is simple. In an unprotected network environment, any client can apply to any server for service. The obvious security risk is that of impersonation. An opponent can pretend to be another client and obtain unauthorized privileges on server machines. To counter this threat, servers must be able to confirm the identities of clients who request service. Each server can be required to undertake this task for each client/server interaction, but in an open environment, this places a substantial burden on each server. An alternative is to use an **authentication server (AS)** that knows the passwords of all users and stores these in a centralized database. Then the user can log onto the AS for identity verification. Once the AS has verified the user's identity, it can pass this information on to an application server, which will then accept service requests from the client.

The trick is how to do all this in a secure way. It simply won't do to have
the client send the user's password to the AS over the network: An opponent
could observe the password on the network and later reuse it. It also won't do for
Kerberos to send a plain message to a server validating a client: An opponent could
impersonate the AS and send a false validation.

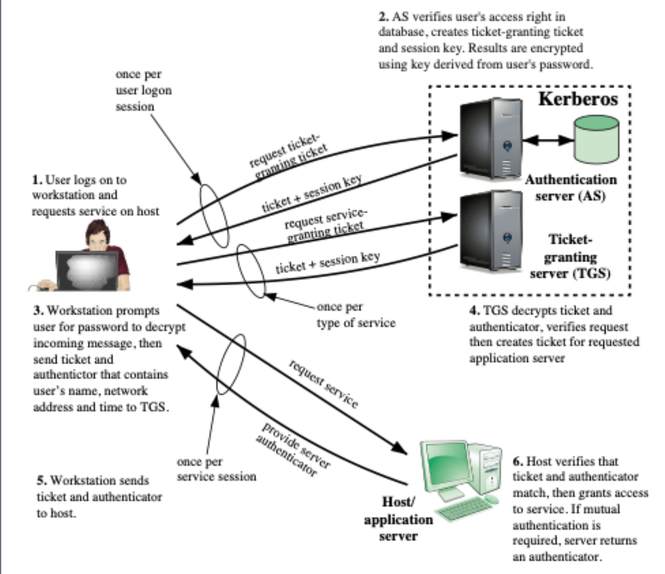

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
once per user logon session
Kerberos
request ticket-granting ticket
ticket + session key
Authentication server (AS)
1. User logs on to workstation and requests service on host
request service-granting ticket
ticket + session key
Ticket-granting server (TGS)
3. Workstation prompts user for password to decrypt incoming message, then send ticket and authentictor that contains user's name, network address and time to TGS.
once per type of service
4. TGS decrypts ticket and authenticator, verifies request then creates ticket for requested application server
request service
provide server authenticator
once per service session
5. Workstation sends ticket and authenticator to host.
Host/ application server
6. Host verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Figure 23.1 Overview of Kerberos

The way around this problem is to use encryption and a set of messages that accomplish the task (see Figure 23.1). The original version of Kerberos used the Data Encryption Standard (DES) as it's encryption algorithm.

The AS shares a unique secret key with each server. These keys have been distributed physically or in some other secure manner. This will enable the AS to send messages to application servers in a secure fashion. To begin, the user logs on to a workstation and requests access to a particular server. The client process representing the user sends a message to the AS that includes the user's ID and a request for what is known as a ticket-granting ticket (TGT) . The AS checks its database to find the password of this user. Then the AS responds with a TGT and a one-time encryption key, known as a session key, both encrypted using the user's password as the encryption key. When this message arrives back 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 has been supplied, the ticket and session key are successfully recovered.

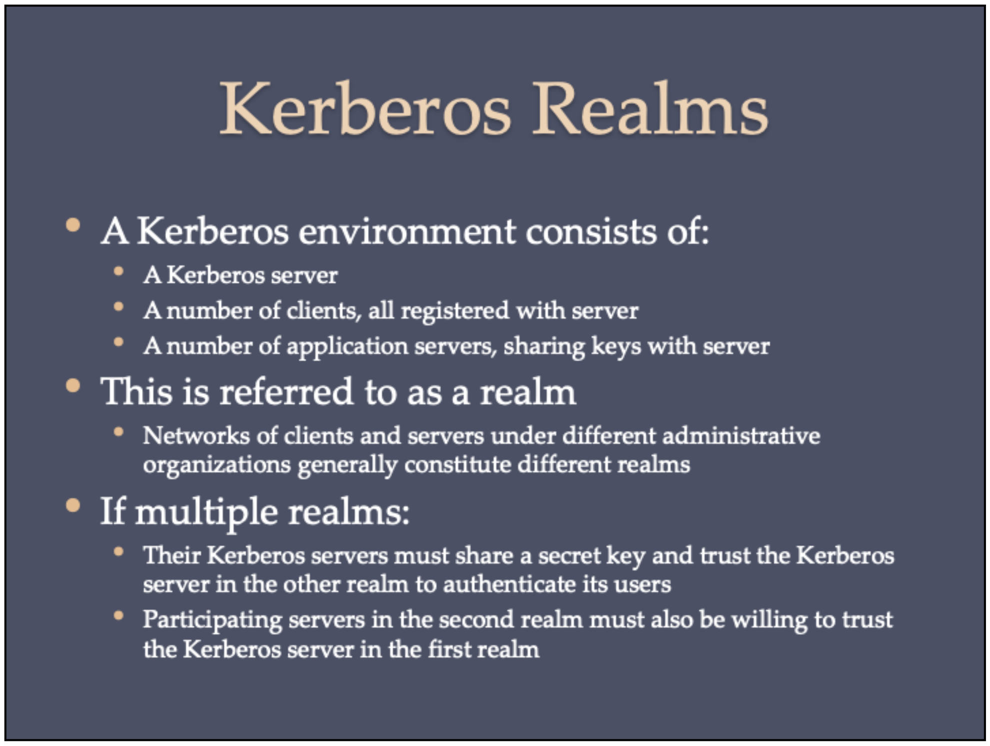
Notice what has happened. The AS has been able to verify the user's identity
since this user knows the correct password, but it has been done in such a way that
the password is never passed over the network. In addition, the AS has passed
information to the client that will be used later on to apply to a server for service,
and that information is secure since it is encrypted with the user's password.

The ticket constitutes a set of credentials that can be used by the client to
apply for service. The ticket indicates that the AS has accepted this client and its
user. The ticket contains the user's ID, the server's ID, a timestamp, a lifetime after
which the ticket is invalid, and a copy of the same session key sent in the outer
message to the client. The entire ticket is encrypted using a secret DES key shared
by the AS and the server. Thus, no one can tamper with the ticket.

Now, Kerberos could have been set up so the AS would send back a ticket granting
access to a particular application server. This would require the client to request
a new ticket from the AS for each service the user wants to use during a logon session,
which would in turn require the AS query the user for his or her password for
each service request, or else to store the password in memory for the duration of
logon session. The first course is inconvenient for the user and the second course is
a security risk. Therefore, the AS supplies a ticket good not for a specific application
service, but for a special ticket-granting server (TGS). The AS gives the client a ticket
that can be used to get more tickets!

The idea is that this ticket can be used by the client to request multiple servicegranting
tickets. So the ticket-granting ticket is to be reusable. However, we do not
wish an opponent to be able to capture the ticket and use it. Consider the following
scenario: An opponent captures the ticket and waits until the user has logged off
the workstation. Then the opponent either gains access to that workstation or configures
his workstation with the same network address as that of the victim. Then
the opponent would be able to reuse the ticket to spoof the TGS. To counter this,

the ticket includes a timestamp, indicating the date and time at which the ticket was issued, and a lifetime, indicating the length of time for which the ticket is valid (e.g., 8 hours). Thus, the client now has a reusable ticket and need not bother the user for a password for each new service request. Finally, note the ticket-granting ticket is encrypted with a secret key known only to the AS and the TGS. This prevents alteration of the ticket. The ticket is reencrypted with a key based on the user's password. This assures that the ticket can be recovered only by the correct user, providing the authentication.



Kerberos Realms

- A Kerberos environment consists of:
  - A Kerberos server
  - A number of clients, all registered with server
  - A number of application servers, sharing keys with server
- This is referred to as a realm
  - Networks of clients and servers under different administrative organizations generally constitute different realms
- If multiple realms:
  - Their Kerberos servers must share a secret key and trust the Kerberos server in the other realm to authenticate its users
  - Participating servers in the second realm must also be willing to trust the Kerberos server in the first realm

A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers, requires the following:

1. The Kerberos server must have the user ID and password of all participating users in its database. All users are registered with the Kerberos server.

2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.

Such an environment is referred to as a realm. Networks of clients and servers under different administrative organizations generally constitute different realms (Figure 23.2). That is, it generally is not practical, or does not conform to administrative
policy, to have users and servers in one administrative domain registered
with a Kerberos server elsewhere. However, users in one realm may need access to servers in other realms, and some servers may be willing to provide service to users from other realms, provided that those users are authenticated.

Kerberos provides a mechanism for supporting such interrealm authentication.
For two realms to support interrealm authentication, the Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.

The scheme requires that the Kerberos server in one realm trust the Kerberos server in the other realm to authenticate its users. Furthermore, the participating servers in the second realm must also be willing to trust the Kerberos server in the first realm.

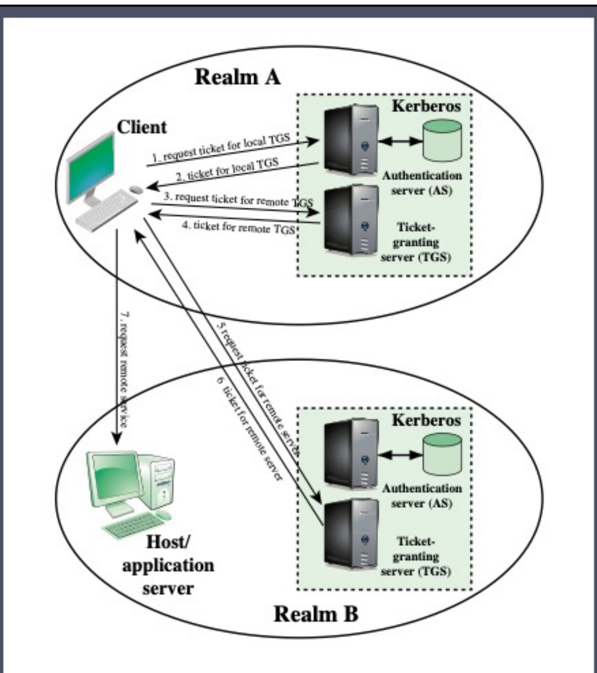
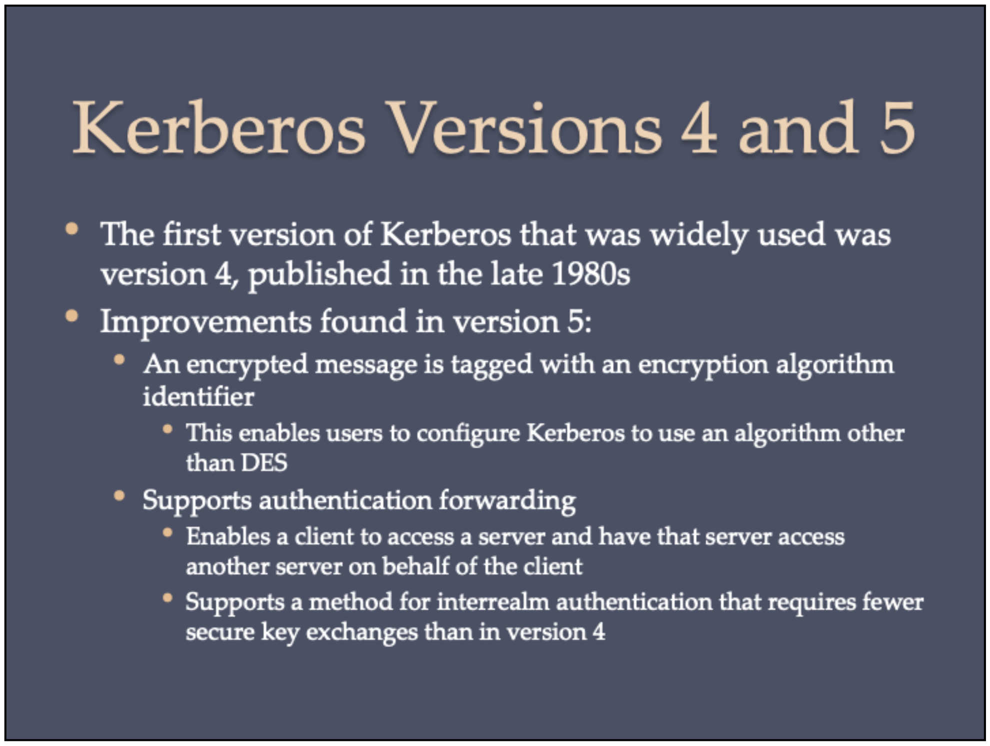

Realm A
Client
Kerberos
1. request ticket for local TGS
2. ticket for local TGS
3. request ticket for remote TGS
4. ticket for remote TGS
Authentication server (AS)
Ticket-granting server (TGS)
7. request remote service
5. request ticket for remote server
6. ticket for remote server
Kerberos
Authentication server (AS)
Ticket-granting server (TGS)
Host/ application server
Realm B

Figure 23.2 Request for Service in Another Realm

With these ground rules in place, we can describe the mechanism as follows (Figure 23.2): A user wishing service on a server in another realm needs a ticket for that server. The user's client follows the usual procedures to gain access to the local TGS and then requests a ticket-granting ticket for a remote TGS (TGS in another realm). The client can then apply to the remote TGS for a service-granting ticket for the desired server in the realm of the remote TGS.

The ticket presented to the remote server indicates the realm in which the user was originally authenticated. The server chooses whether to honor the remote request.



Kerberos Versions 4 and 5

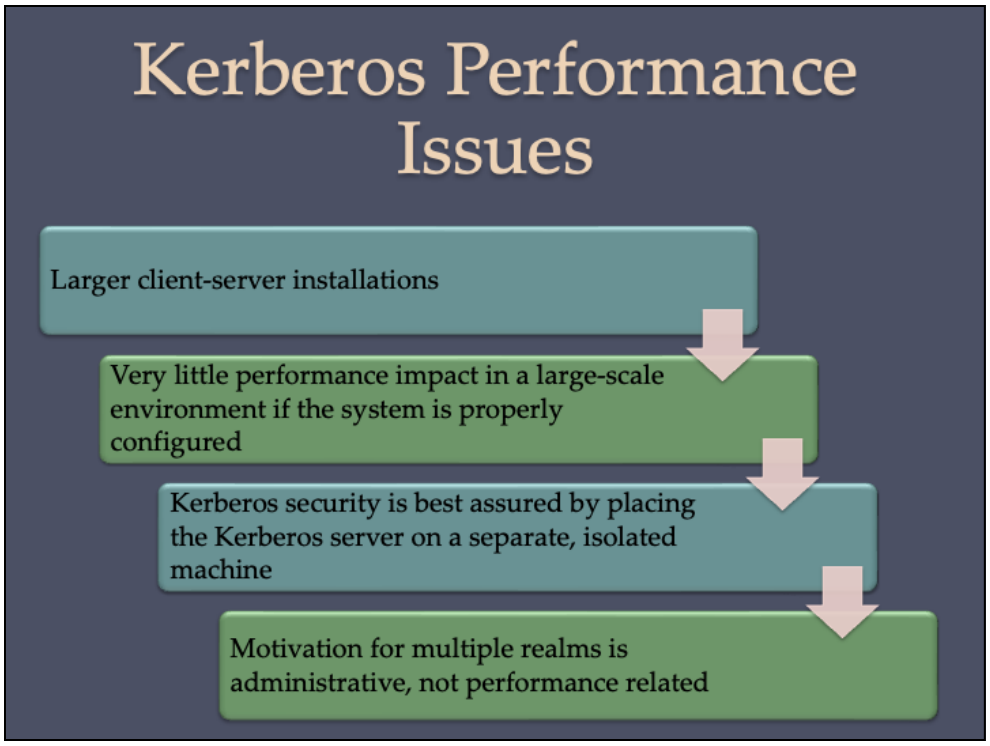
- The first version of Kerberos that was widely used was version 4, published in the late 1980s
- Improvements found in version 5:
  - An encrypted message is tagged with an encryption algorithm identifier
    - This enables users to configure Kerberos to use an algorithm other than DES
  - Supports authentication forwarding
    - Enables a client to access a server and have that server access another server on behalf of the client
    - Supports a method for interrealm authentication that requires fewer secure key exchanges than in version 4

The first version of Kerberos that was widely used was version 4, published in the late
1980s. An improved and extended version 5 was introduced in 1993, and updated in 2005. Kerberos version 5 is now widely implemented, including as part of Microsoft's Active Directory service, in most current UNIX and Linux systems, and in Apple's Mac OS X. It includes a number of improvements over version 4. First, in version 5, an encrypted message is tagged with an encryption algorithm identifier. This enables users to configure Kerberos to use an algorithm other than DES, with the Advanced Encryption Standard (AES) now the default choice.

Version 5 also supports a technique known as 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. Authentication forwarding enables 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.

Finally, version 5 supports a method for interrealm authentication that requires fewer secure key exchanges than in version 4.



Kerberos Performance Issues

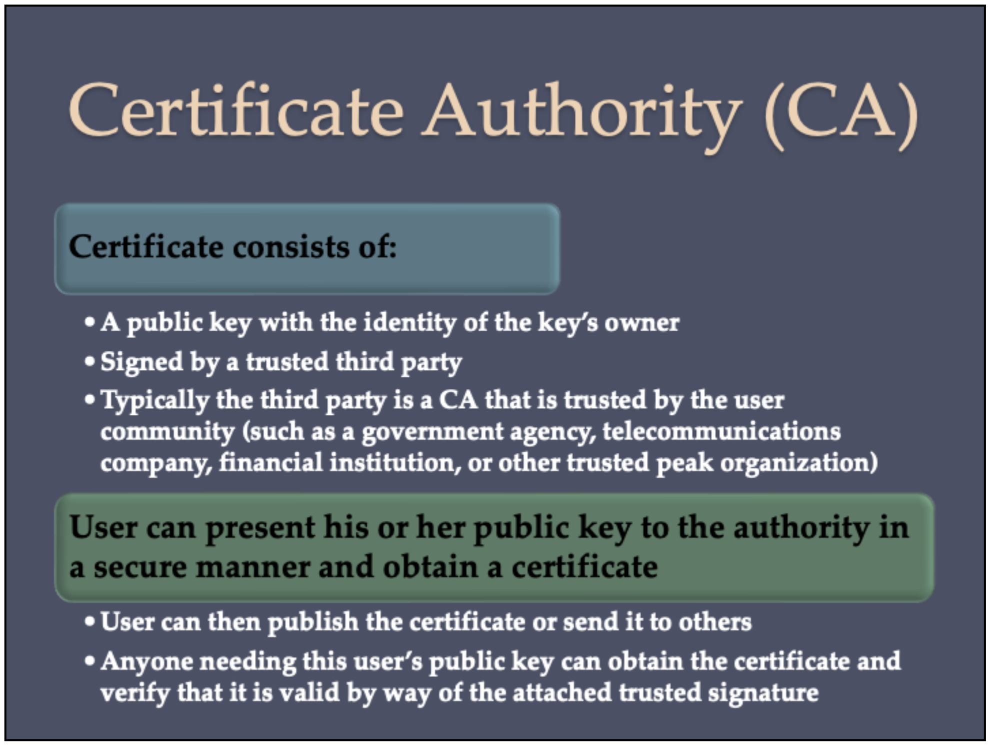
- Larger client-server installations
- Very little performance impact in a large-scale environment if the system is properly configured
- Kerberos security is best assured by placing the Kerberos server on a separate, isolated machine
- Motivation for multiple realms is administrative, not performance related

As client/server applications become more popular, larger and larger client/server installations are appearing. A case can be made that the larger the scale of the networking environment, the more important it is to have logon authentication. But the question arises: What impact does Kerberos have on performance in a large-scale environment?

Fortunately, the answer is that there is very little performance impact if the system is properly configured. Keep in mind that tickets are reusable. Therefore, the amount of traffic needed for the granting ticket requests is modest. With respect to the transfer of a ticket for logon authentication, the logon exchange must take place anyway, so again the extra overhead is modest.

A related issue is whether the Kerberos server application requires a dedicated platform or can share a computer with other applications. It probably is not wise to run the Kerberos server on the same machine as a resource-intensive application such as a database server. Moreover, the security of Kerberos is best assured by placing the Kerberos server on a separate, isolated machine.

Finally, in a large system, is it necessary to go to multiple realms in order to maintain performance? Probably not. Rather, the motivation for multiple realms is administrative. If you have geographically separate clusters of machines, each with its own network administrator, then one realm per administrator may be convenient. However, this is not always the case.



Certificate Authority (CA)

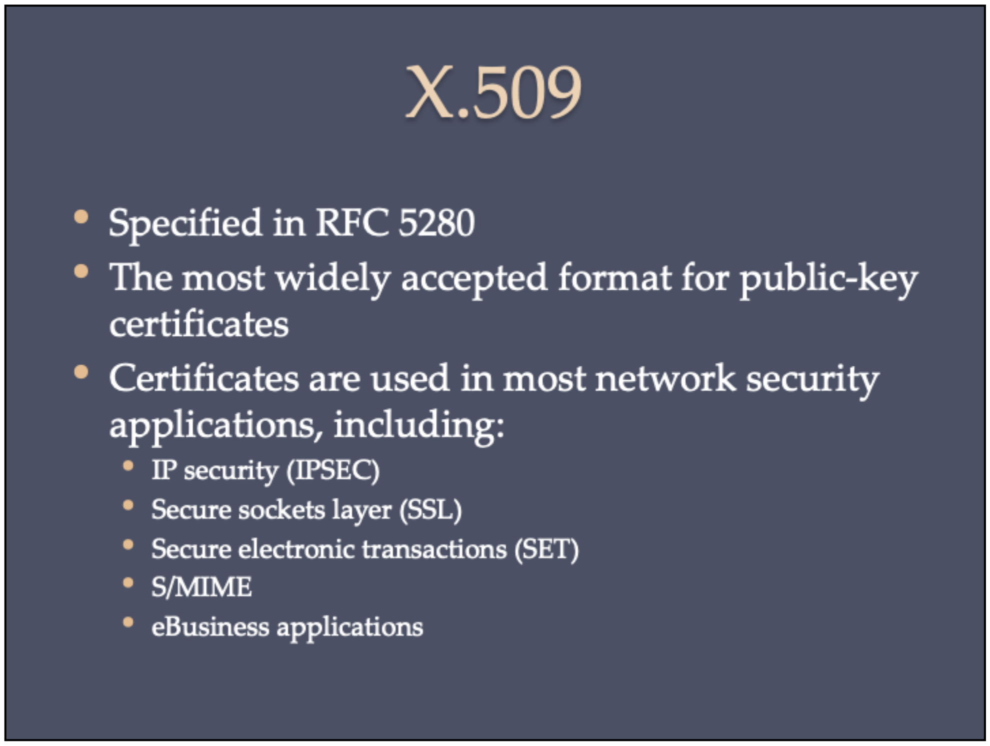
Certificate consists of:

- A public key with the identity of the key's owner
- Signed by a trusted third party
- Typically the third party is a CA that is trusted by the user community (such as a government agency, telecommunications company, financial institution, or other trusted peak organization)

User can present his or her public key to the authority in a secure manner and obtain a certificate

- User can then publish the certificate or send it to others
- Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature

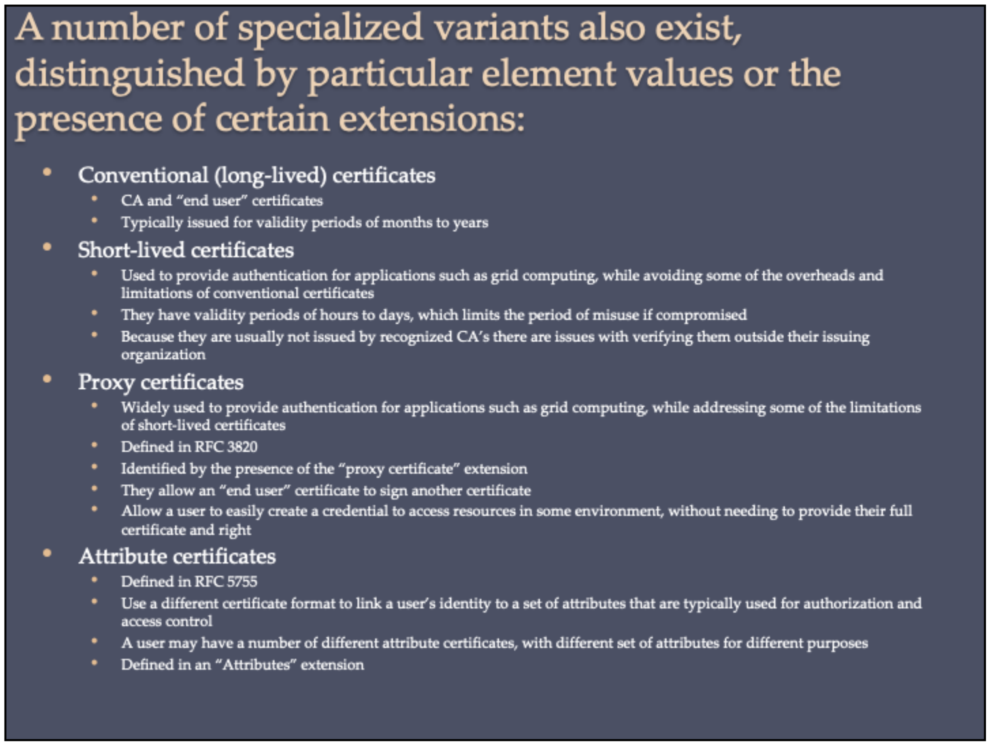
Public-key certificates are mentioned briefly in Section 2.4. Recall that a certificate links a public key with the identity of the key's owner, with the whole block signed by a trusted third party. Typically, the third party is a **certificate authority (CA)** that is trusted by the user community, such as a government agency, financial institution, telecommunications company, or other trusted peak organization. A user can present his or her public key to the authority in a secure manner and obtain a certificate. The user can then publish the certificate, or send it to others. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature, provided they can verify the CA's public key. Figure 2.8 illustrates this process.



X.509

- Specified in RFC 5280
- The most widely accepted format for public-key certificates
- Certificates are used in most network security applications, including:
  - IP security (IPSEC)
  - Secure sockets layer (SSL)
  - Secure electronic transactions (SET)
  - S/MIME
  - eBusiness applications

The X.509 ITU-T standard, also specified in RFC 5280 **(***Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile***,** 2008**),** is the most widely accepted format for public-key certificates. X.509 certificates are used in most network security applications, including IP security (IPSEC), secure sockets layer (SSL), secure electronic transactions (SET), and S/MIME, as well as in eBusiness applications.



A number of specialized variants also exist, distinguished by particular element values or the presence of certain extensions:

- Conventional (long-lived) certificates
  - CA and "end user" certificates
  - Typically issued for validity periods of months to years
- Short-lived certificates
  - Used to provide authentication for applications such as grid computing, while avoiding some of the overheads and limitations of conventional certificates
  - They have validity periods of hours to days, which limits the period of misuse if compromised
  - Because they are usually not issued by recognized CA's there are issues with verifying them outside their issuing organization
- Proxy certificates
  - Widely used to provide authentication for applications such as grid computing, while addressing some of the limitations of short-lived certificates
  - Defined in RFC 3820
  - Identified by the presence of the "proxy certificate" extension
  - They allow an "end user" certificate to sign another certificate
  - Allow a user to easily create a credential to access resources in some environment, without needing to provide their full certificate and right
- Attribute certificates
  - Defined in RFC 5755
  - Use a different certificate format to link a user's identity to a set of attributes that are typically used for authorization and access control
  - A user may have a number of different attribute certificates, with different set of attributes for different purposes
  - Defined in an "Attributes" extension

The CA and "end user" certificates discussed above are the most common form of X.509 certificates. However, a number of specialized variants also exist, distinguished by particular element values or the presence of certain extensions. Variants include:

• **Conventional (long-lived) certificates:** are the CA and "end user" certificates discussed above. They are typically issued for validity periods of months to years.

• **Short-lived certificates:** are used to provide authentication for applications such as grid computing, while avoiding some of the overheads and limitations of conventional certificates [HSU98]. They have validity periods of hours to days, which limits the period of misuse if compromised. Because they are usually not issued by recognized CA's, there are issues with verifying them outside their issuing organization.

• **Proxy certificates:** are now widely used to provide authentication for applications

such as grid computing, while addressing some of the limitations of shortlived certificates. RFC 3820 (*Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*, 2004) defines proxy certificates, which are identified by the presence of the "proxy certificate" extension. They allow an "end user" certificate to sign another certificate, which must be an extension of the existing certificate with a sub-set of their identity, validity period, and authorizations. They allow a user to easily create a credential to access resources in some environment,
without needing to provide their full certificate and rights. There are other proposals to use proxy certificates as network access capability tickets, which authorize a user to access specific services with specific rights.

• **Attribute certificates:** use a different certificate format, defined in RFC 5755 (*An Internet Attribute Certificate Profile for Authorization*, 2010), to link a user's identity to a set of attributes that are typically used for authorization and access control. A user may have a number of different attribute certificates, with different sets of attributes for different purposes, associated with their main conventional certificate. These attributes are defined in an "Attributes" extension.
These extensions could also be included in a conventional certificate, but this is discouraged as being too inflexible. They may also be included in a proxy certificate, further restricting its use, and this is appropriate for some applications.

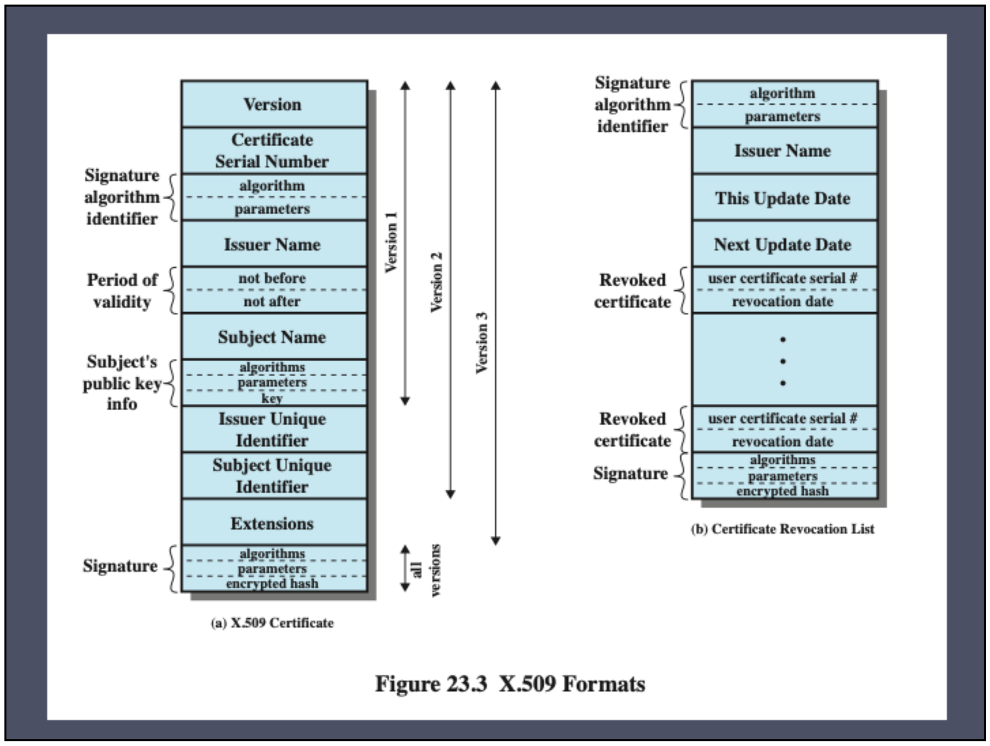


Figure 23.3 X.509 Formats

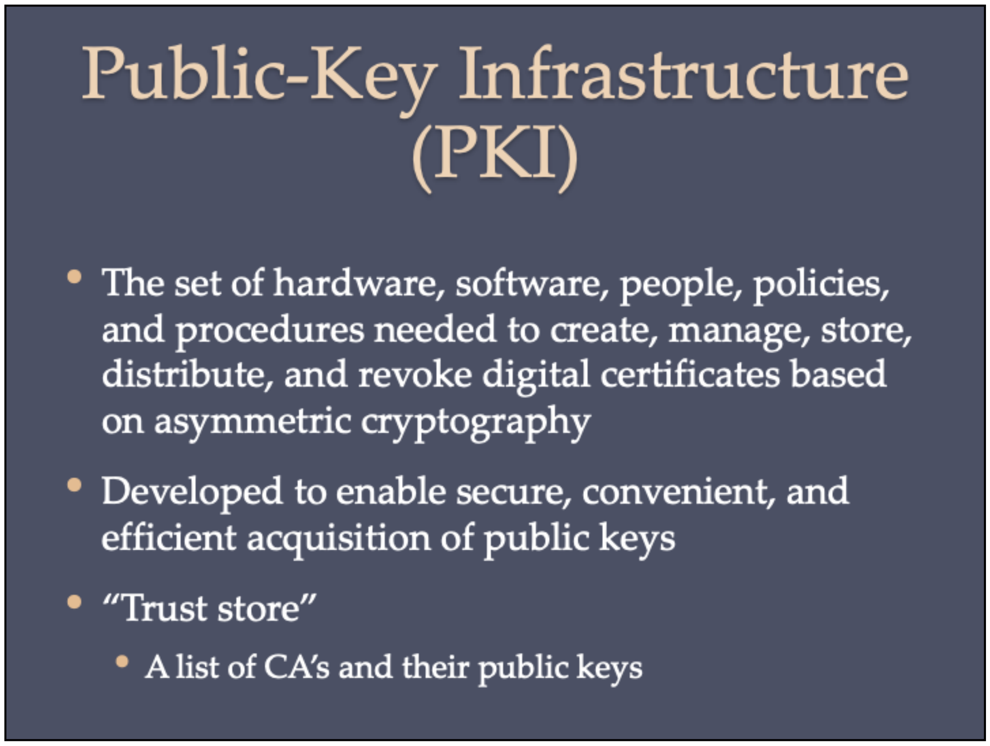
The X.509 standard defines a certificate revocation list (CRL), signed by the issuer, that includes the elements shown in Figure 23.3b. Each revoked certificate entry contains a serial number of a certificate and the revocation date for that certificate.

Because serial numbers are unique within a CA, the serial number is sufficient to identify the certificate. When an application receives a certificate, the X.509 standard states it should determine whether it has been revoked, by checking against the current CRL for its issuing CA. However, due to the overheads in retrieving and storing these lists, very few applications actually do this. “The recent Heartbleed Open SSL bug, which has forced the revocation and replacement of very large numbers of server certificates, has dramatically highlighted deficiencies with the use of CRLs.”

A more practical alternative is to use the RFC 6960 (*X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, 2013), to query the CA as to whether a specific certificate is valid. This lightweight protocol is increasingly used,
including in recent versions of most common Web browsers. The “Authority

Information
Access” extension in a certificate can specify the address of the OCSP server to
use, if the signing CA supports this protocol.

Originally, most X.509 certificates signed an MD5 hash of their contents.
Unfortunately,
research advances in creating MD5 collisions has led to the development of
several techniques for forging new certificates for different identities that have the
same hash, and hence can reuse the same signature, as an existing valid certificate
[STEV07]. The Flame malware authors used this approach to forge what appeared
to be a valid Microsoft code-signing certificate. This allowed the malware to remain
undetected for more than 2 years before being identified in 2012. The use of MD5
was depreciated, and the SHA-1 hash algorithm recommended, in the 2000s. However
the creation of SHA-1 collisions in 2017 means that, in turn, this algorithm is no
longer considered secure. As of early 2017, most browsers now reject certificates
using
SHA-1 or MD5. The current requirement is to use one of the SHA-2 hash algorithms
in certificates, with support for SHA-3 as an alternative likely soon.



Public-Key Infrastructure (PKI)

- The set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography
- Developed to enable secure, convenient, and efficient acquisition of public keys
- "Trust store"
  - A list of CA's and their public keys

RFC 4949 (*Internet Security Glossary, Version 2,* 2007) defines public-key infrastructure (PKI) as

the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.

The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys.

In order to verify a certificate, you need to know the public key of the signing CA. This could, in turn, be provided in another certificate, signed by a parent CA, with the CA's organized in a hierarchy. Eventually, however, you must reach the top of the hierarchy, and have a copy of the public key for that root CA. The X.509 standard describes a PKI model that originally assumed there would be a single internationally specified hierarchy of government regulated CAs. This did not happen. Instead, current X.509 PKI implementations came with a large list of CAs and their public keys, known as a "trust store." These CAs usually either directly sign "end-user" certificates or sign a small number of Intermediate-CAs that in turn sign “end-user” certificates. Thus all the hierarchies are very small, and all are equally trusted. Users and servers that want an automatically verified certificate must acquire it from one of these CAs. Alternatively they can use either a “self-signed” certificate or a certificate signed by some other CA. However, in both these cases, such certificates will initially be recognized as “untrusted” and the user presented with stark warnings about accepting such certificates, even if they are actually legitimate.

There are many problems with this model of a PKI, and these have been known for many years [GUTM02], [GRUS13]. Current implementations suffer from a number of critical issues. The first is the reliance on the user to make an informed decision when there is a problem verifying a certificate. Unfortunately, it is clear that most users do not understand what a certificate is and why there might be a problem. Hence they choose to accept a certificate, or not, for reasons that have little to do with their security, which may result in the compromise of their systems.

Another critical problem is the assumption that all of the CAs in the “trust store” are equally trusted, equally well managed, and apply equivalent policies. This was dramatically illustrated by the compromise of the DigiNotar CA in 2011 that resulted in the fraudulent issue of certificates for many well-known organizations. It is widely believed these were used by the Iranian government to mount a “man-the-middle” attack on the secured communications of many of their citizens. As a consequence, the DigiNotar CA keys were removed from the “trust store” in many systems, and the company was declared bankrupt later that year. Another CA, Comodo, was also compromised in 2011, with a small number of fraudulent certificates issued.

A further concern is that different implementations, in the various web browsers and operating systems, use different “trust stores,” and hence present different security views to users.

Given these and other issues, several proposals exist to improve the practical handling of X.509 certificates. Some of these recognize that many applications do not require formal linking of a public key to a verified identity. In many web applications, for example, all users really need is to know that if they visit the same secure site and are supplied with a certificate for it, that it is the same site and same key as when they previously visited. This is analogous to ensuring that if you visit the same physical store, you see the same company name and layout and staff as previously. And further, users want to know that it is the same site and same key as other users in other locations see.

The first of these, confirming continuity in time, can be provided by user's applications keeping a record of certificate details for all sites they visit, and checking against these on subsequent visits. Certificate pinning in applications can provide this feature, as is used in Google Chrome. The Firefox "Certificate Patrol" extension is another example of this approach.

The second, confirming continuity in space, requires the use of a number of widely separated "network notary servers" that keep records of certificates for all sites they view, that can be compared with a certificate provided to the user in any instance. The "Perspectives Project" is a practical implementation of this approach, which may be accessed using the Firefox "Perspectives" plugin. This also verifies the time history of certificates in use, thus providing both desired features for this approach. The "Google Certificate Catalog" and "Google Certificate Transparency" project are other examples of such notary servers.

In either of the above cases, identification of a different certificate and key to that seen at other times or places may well be an indication of attack or other problems. It may also simply be the result of certificates being updated as they approach expiry, or of organizations incorrectly using multiple certificates and keys for the same, but replicated, server. These latter issues need to be managed by such extensions.

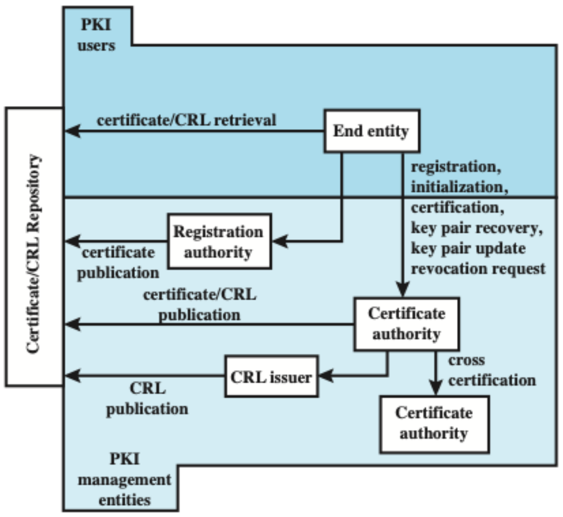
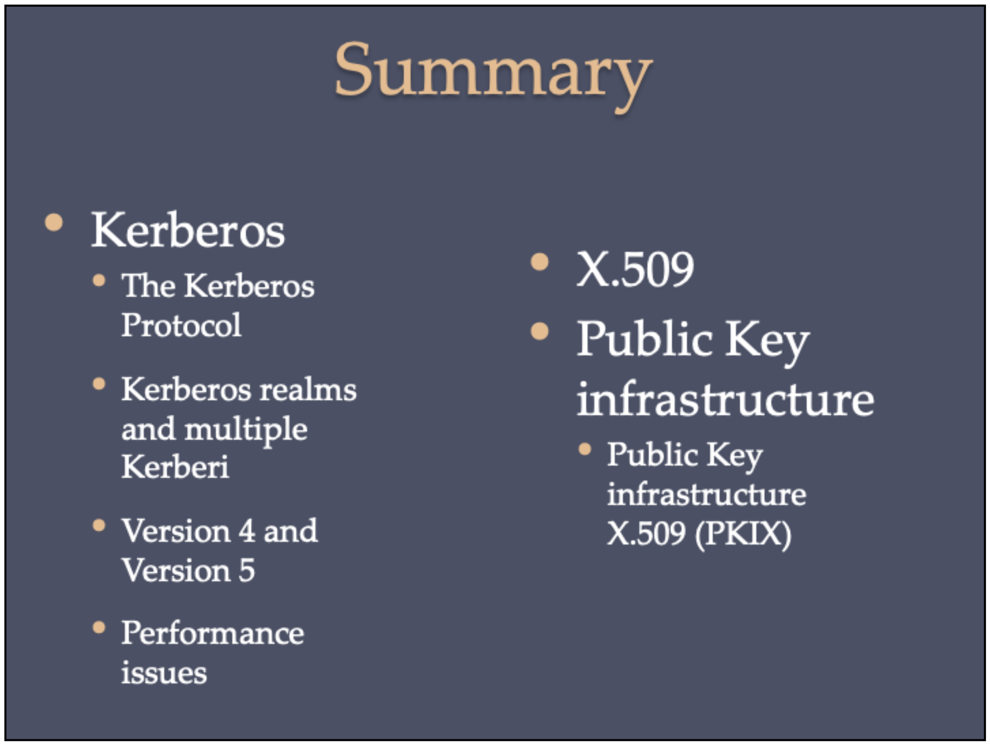

PKI users
certificate/CRL retrieval
End entity
registration, initialization, certification, key pair recovery, key pair update revocation request
Certificate/CRL Repository
Registration authority
certificate publication
certificate/CRL publication
Certificate authority
cross certification
CRL issuer
CRL publication
Certificate authority
PKI management entities

Figure 23.4 PKIX Architectural Model

The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the driving force behind setting up a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet. This section briefly describes the PKIX model. For more detail, see [STAL17].

Figure 23.4 shows the interrelationship among the key elements of the PKIX model. These elements include the **End entity** (e.g., user or server) for which the certificate for and the **Certificate authority** that issues the certificates. The CA's management functions may be further divided to include the **Registration authority (RA)** that handles end entity registration and the CRL issuer and Repository that manage CRLs.

PKIX identifies a number of management functions that potentially need to be supported by management protocols. These are indicated in Figure 23.4 and include user Registration, Initialization of key material, Certification in which a CA issues a certificate, Key pair recovery and update, Revocation request for a certificate, and Cross certification between CAs.



Summary

- Kerberos
  - The Kerberos Protocol
  - Kerberos realms and multiple Kerberi
  - Version 4 and Version 5
  - Performance issues
- X.509
- Public Key infrastructure
  - Public Key infrastructure X.509 (PKIX)

Chapter 23 summary.