Unit: Network Security

B.Tech VIII

NETWORK AND SYSTEM SECURITY (CORE ELECTIVE - 5) (CS424)

Book :

Computer Security: Principles and Practice
By William Stallings

Chapter 22: Internet Security Protocols and Standards
Chapter 23: Internet Authentication Applications
Chapter 24: Wireless Network Security

Contents

- **Chapter 23: Internet Authentication Applications**
  - Kerberos
  - X.509
  - Public-Key Infrastructure

Kerberos

Kerberos

- An organization can use several approaches to secure networked servers and hosts(clients)
  - **One-time passwords –**
    - require special equipment such as smart cards or synchronized password generators.
  - **Biometric systems –**
    - automated methods of verifying or recognizing identity on the basis of some physiological characteristic.
    - require specialized equipment.
  - **Authentication software**
    - approach taken by Kerberos

Kerberos

- Kerberos is
  - a **protocol for authenticating service requests** between trusted hosts across an untrusted network, such as the internet.
  - developed at MIT(Massachusetts Institute of Technology).
  - a software utility available both in the public domain and in commercially supported versions.
  - built in to all major computer operating systems, including Microsoft Windows, Apple macOS, FreeBSD and Linux.
  - also used by Broadband service providers to authenticate cable modems and set-top boxes accessing their networks.

Kerberos

- Kerberos (Cerberus) was a three-headed dog who guarded the gates of Hades.
- The three heads of the Kerberos protocol represent the following:
  1. the **client** or principal.
  2. the network resource, which is the **application server** that provides access to the network resource.
  3. a **key distribution center (KDC),** which acts as Kerberos' trusted third-party authentication service.

Why Kerberos ?

- In an unprotected network environment, any client can apply to any server for service.
- **Security risk - impersonation** (i.e. 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.
- In open environment, multiple servers and multiple clients.
- The activity of checking identity of each client, puts substantial burden on each server.
- Kerberos offers solution for this.

Kerberos

- **Key objectives**
  - Passwords must never be transmitted over the network.
  - Passwords must never be stored on client systems and must always be discarded immediately after they are used.
  - Passwords are never stored in plaintext even on the authentication servers. The hash of password is stored.
  - A password is entered only once each session. This is an early form of **single sign-on (SSO) authentication**, and it means that users can authenticate themselves just once but still access any systems for which they are authorized.
  - All authentication information is maintained in a centralized **authentication server**.

Kerberos

- **Key objectives (cont..)**
  - The **application servers** themselves do not store any authentication information. This enables the following features:
    - An administrator can disable authorization for a user to use any application server from the centralized authentication server.
    - Access to individual servers is not necessary to revoke authorization.
    - A single user password is enough to access all Kerberos-authenticated services. A user can reset their password just once, no matter how many services they are authenticated to use.
    - Protecting user information is simplified since all user authentication information is stored on one centralized authentication server rather than on all the individual servers the user is authorized to use.

Kerberos

- Three Key components (as discussed earlier)
  - **Kerberos Client** – who ask for the service of some server
  - **Kerberos Application Server** – who offer service on request from client.
  - **Kerberos KDC (Key Distribution Centre)** – Includes 3 components
    - **Kerberos database** - centralized repository that stores authentication information for each client.
    - **Kerberos Authentication Server (AS)** – verifies the authenticity of client and issues (ticket granting ticket + session key) to client.
    - **Kerberos Ticket Granting Server (TGT)** – issues service granting ticket for the specific application server to client.

Kerberos

- Overview

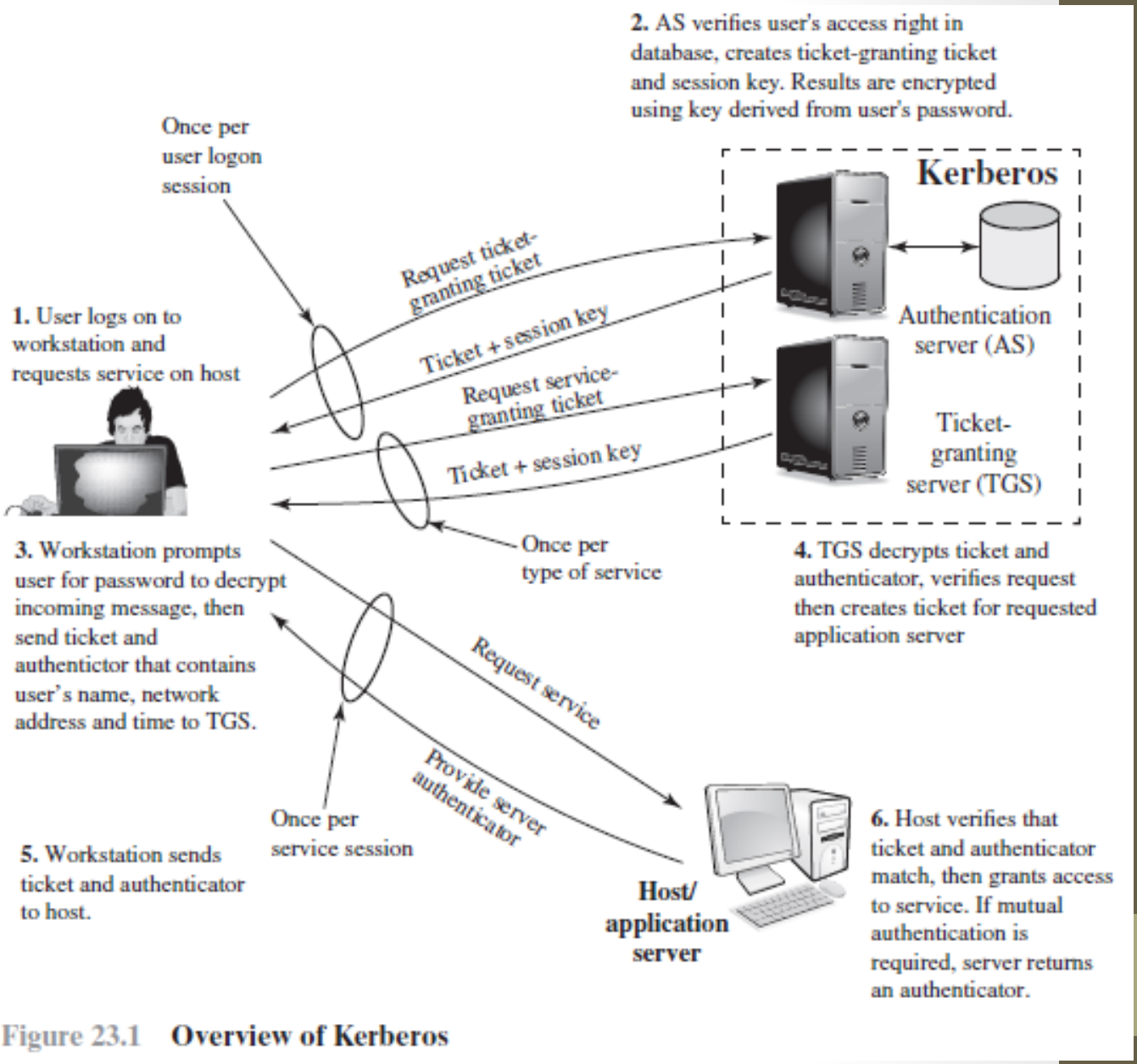


Figure 23.1 **Overview of Kerberos**

Kerberos

- Communication between Kerberos Authentication Server and Application Server
  - Authentication Server(AS) knows the passwords(in the hash form) of all clients and stores these in a centralized database.
  - 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.
  - **All the communication between user(client), AS and application server is in encrypted form.**
  - For encryption, the **Data Encryption Standard (DES)** is used where secret keys have been distributed physically or in some other secure manner.

Kerberos

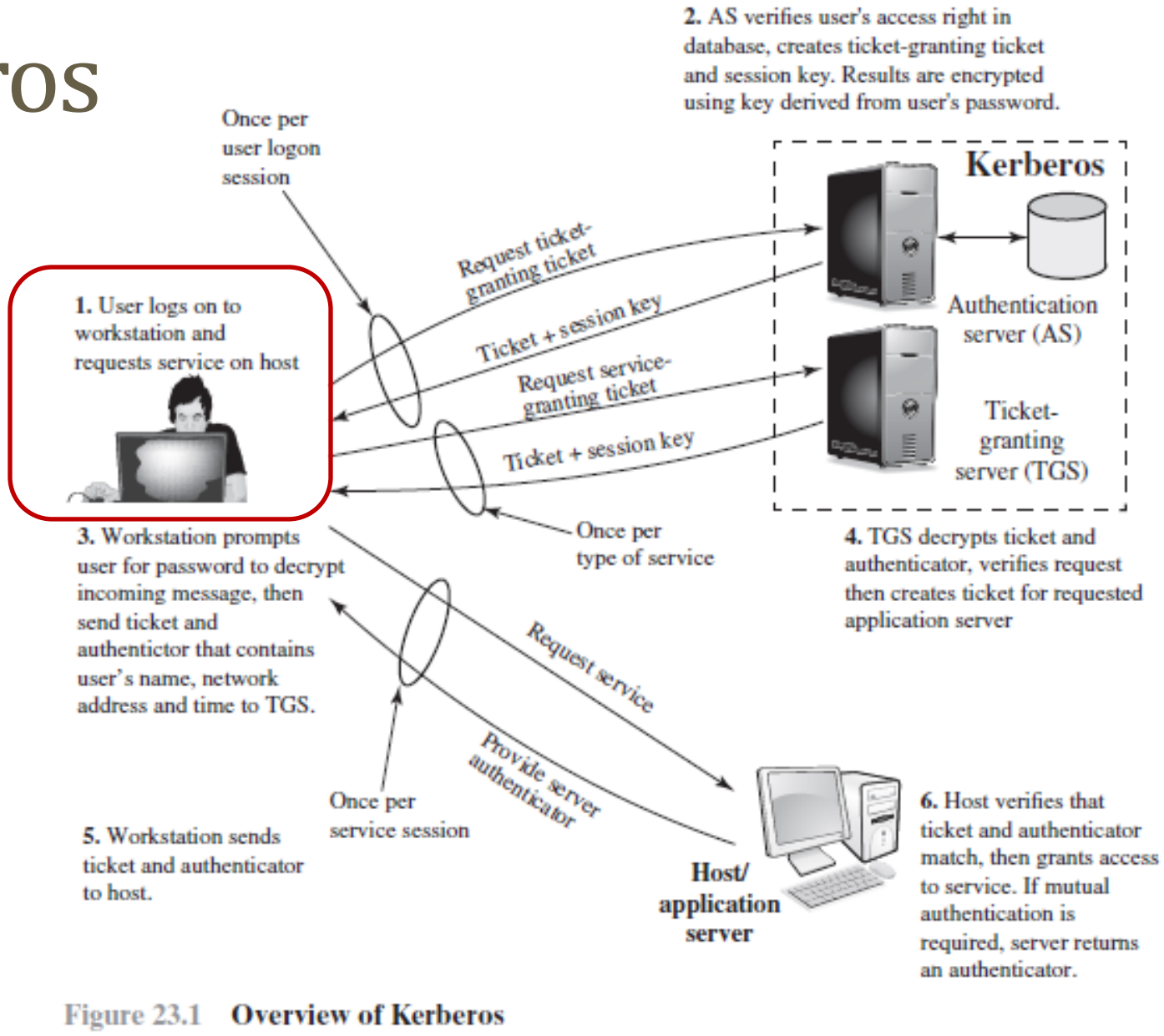


Figure 23.1 **Overview of Kerberos**

- A user logs on to a workstation and requests access to a particular server.

Kerberos

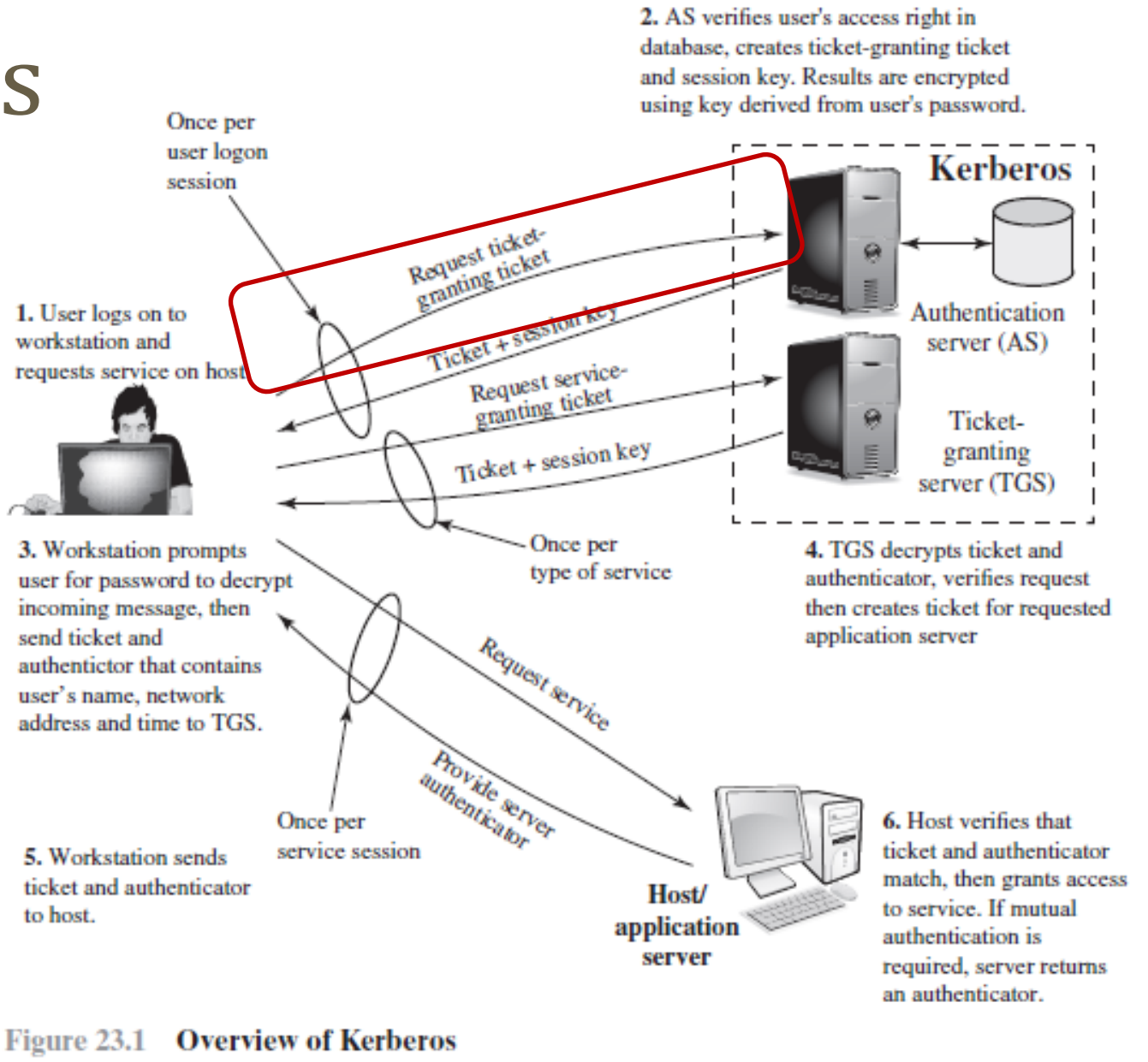


Figure 23.1 **Overview of Kerberos**

- The client process representing the user sends a message to the AS that includes the user's ID and a request for ticket-granting ticket (TGT).

Kerberos

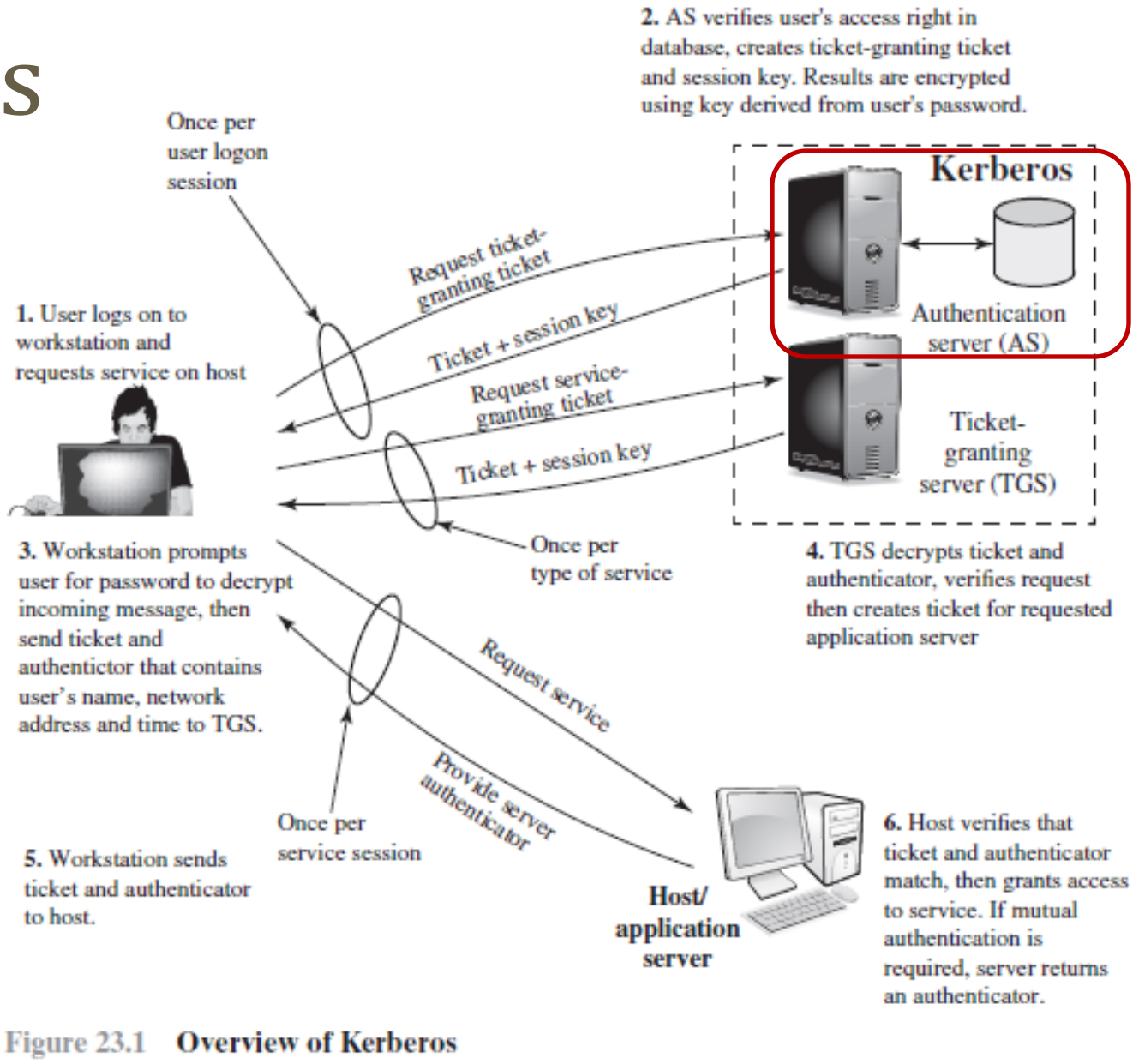


Figure 23.1 **Overview of Kerberos**

- The AS checks its database to find the password of this user.

Kerberos

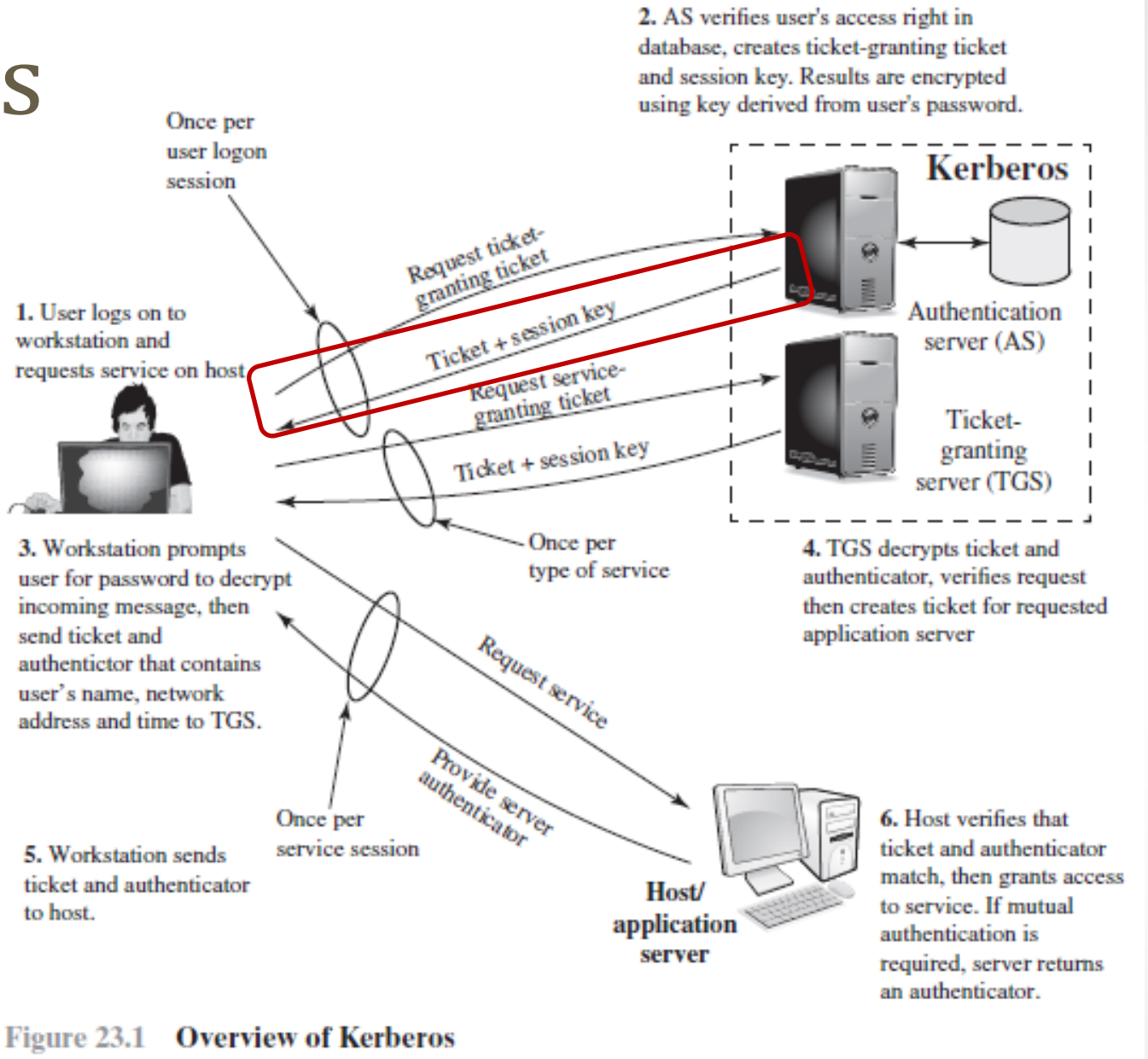


Figure 23.1 **Overview of Kerberos**

- AS responds with a **TGT (Ticket Granting Ticket)** and a one-time encryption key(**session key**), both encrypted using the user's password as the encryption key.

Kerberos

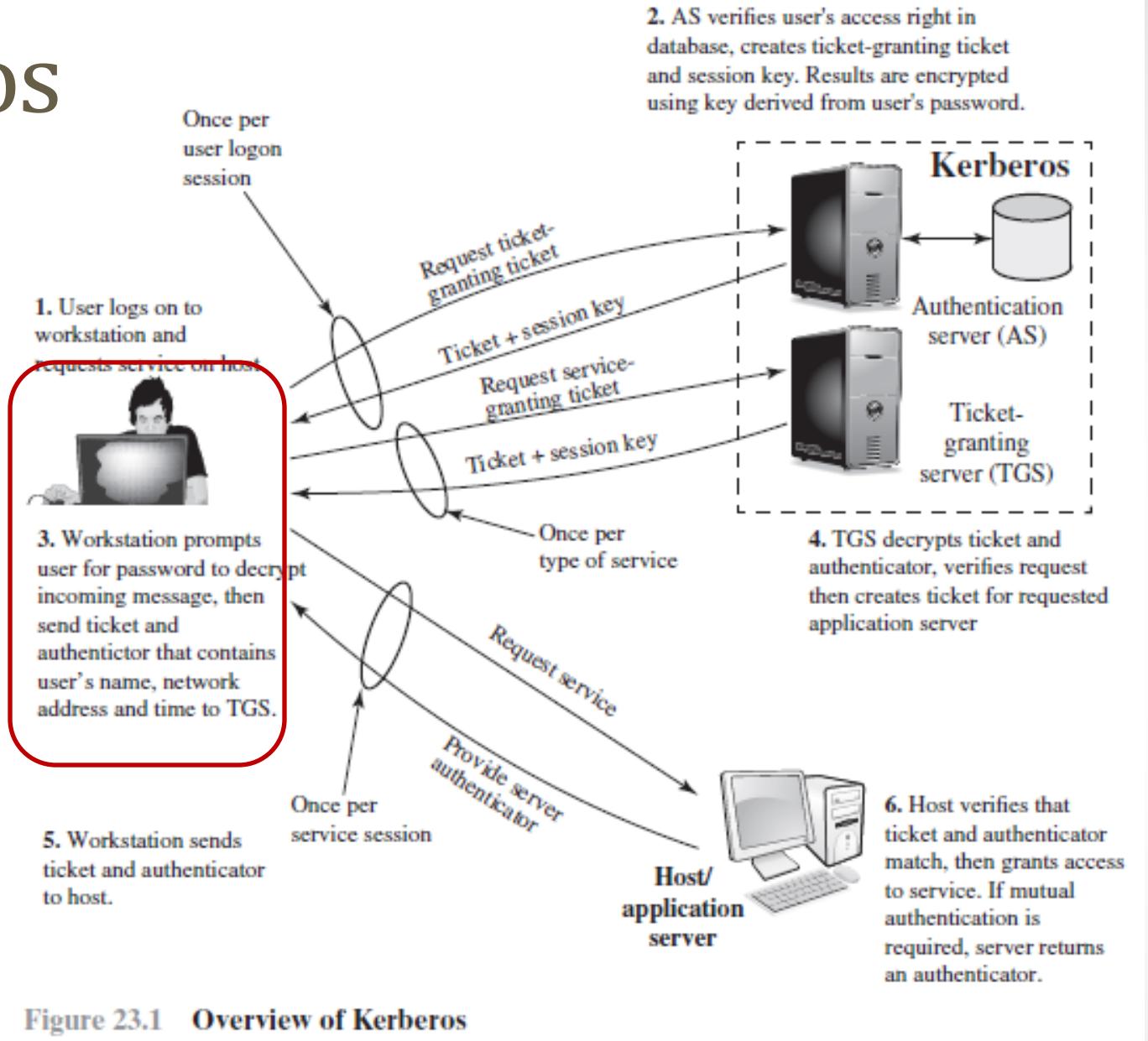


Figure 23.1 **Overview of Kerberos**

- The client prompts the user for his password, generates the key, and decrypt the incoming message to get ticket and session key.

Kerberos

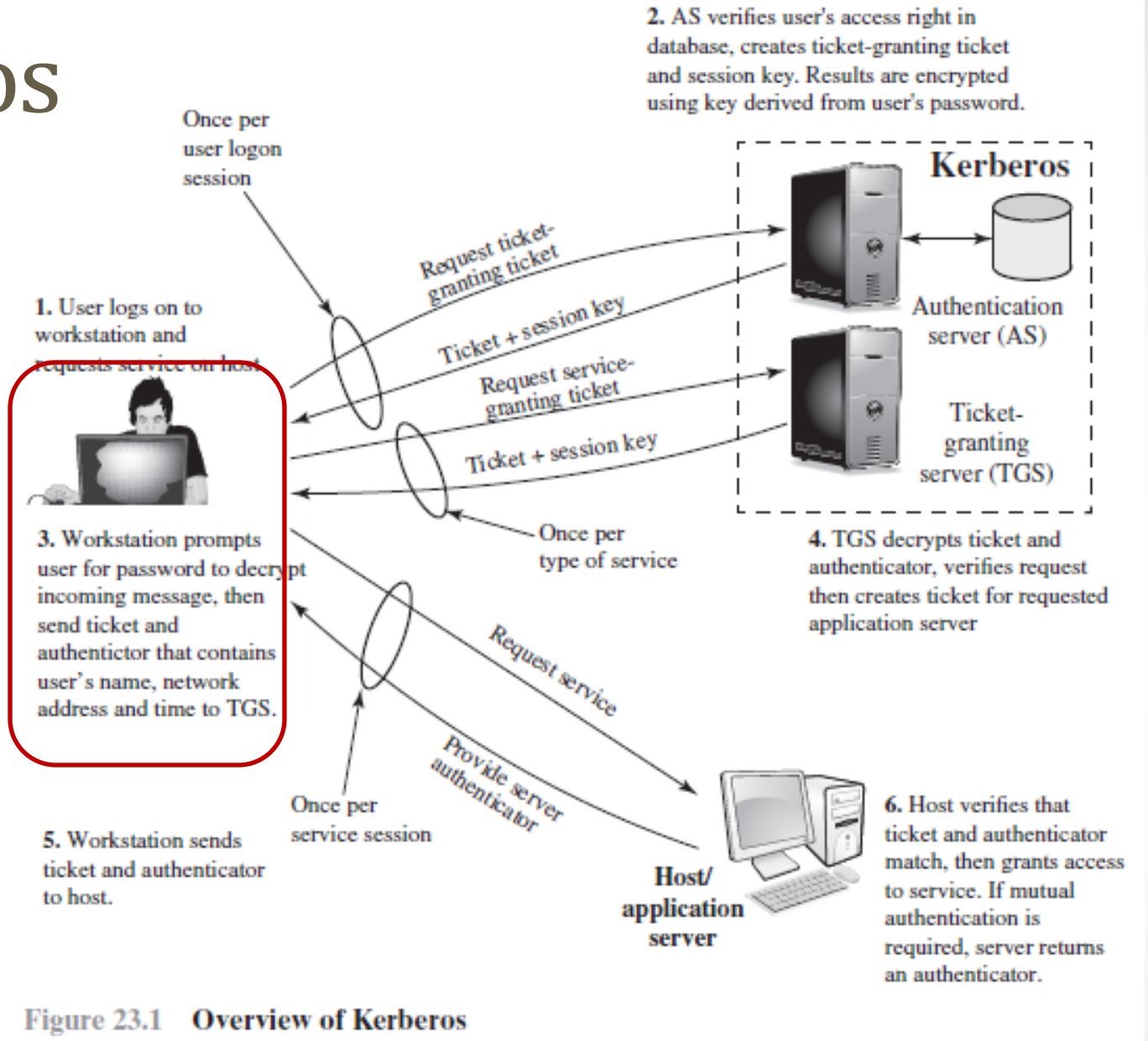


Figure 23.1 **Overview of Kerberos**

- **Note: (i) No password has passed over network, (ii) the communication between AS and client is secure by means of encryption with client's password.**

Kerberos

- **Another Notes:**
  - The TGT indicates that the AS has accepted the client.
  - The TGT
    - Constitutes a set of credentials that can be used by the client to apply for service.
    - 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 TGT is encrypted using a secret DES key shared by the AS and TGS. Thus, no one can tamper with the ticket.
  - **TGT is reusable and client use the same TGT to request for multiple service granting ticket to TGS (Ticket Granting Server).**

Kerberos

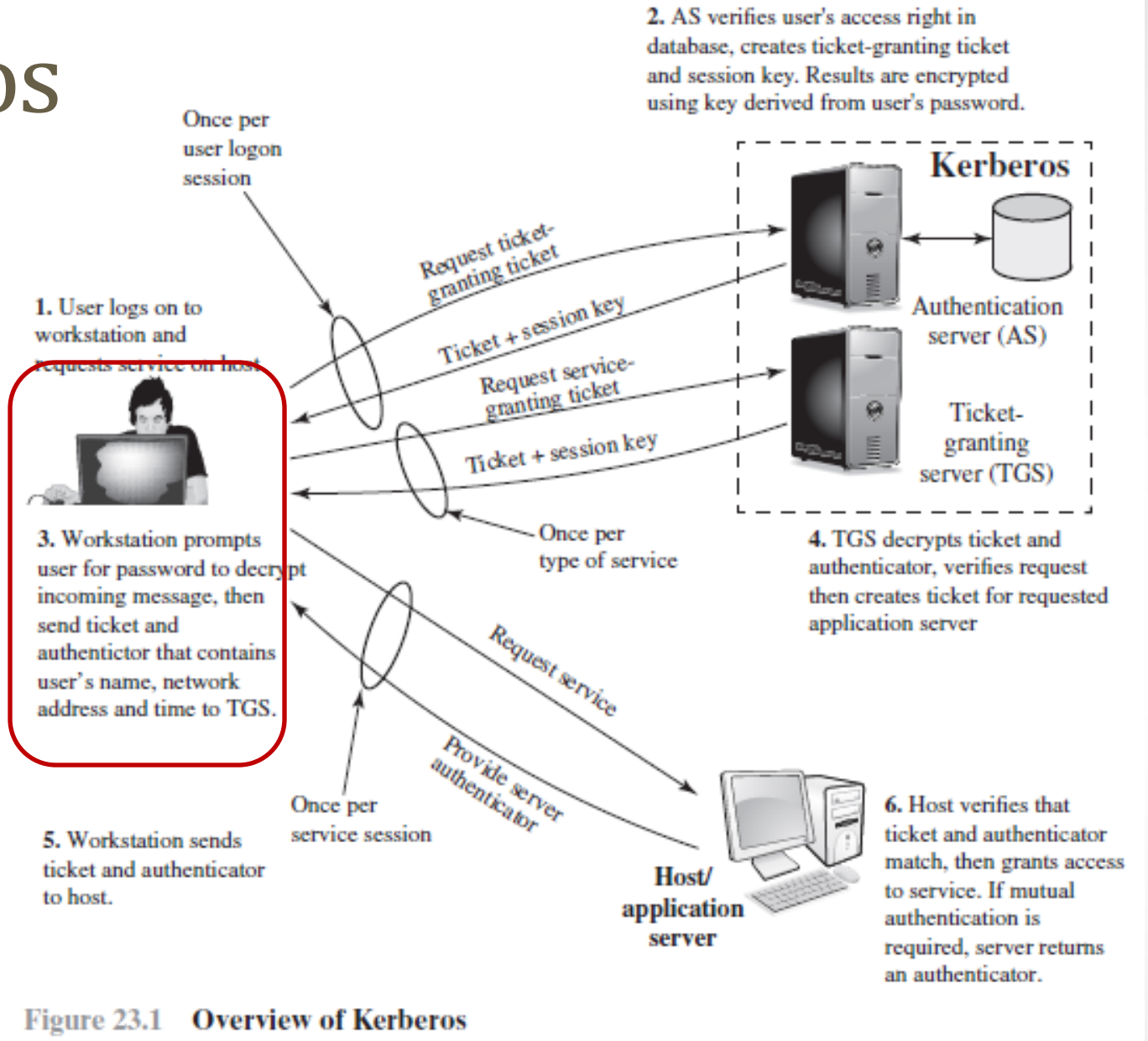


Figure 23.1 **Overview of Kerberos**

- **Assume that the client X has requested access to server V and has obtained a TGT and a temporary session key.**

Kerberos

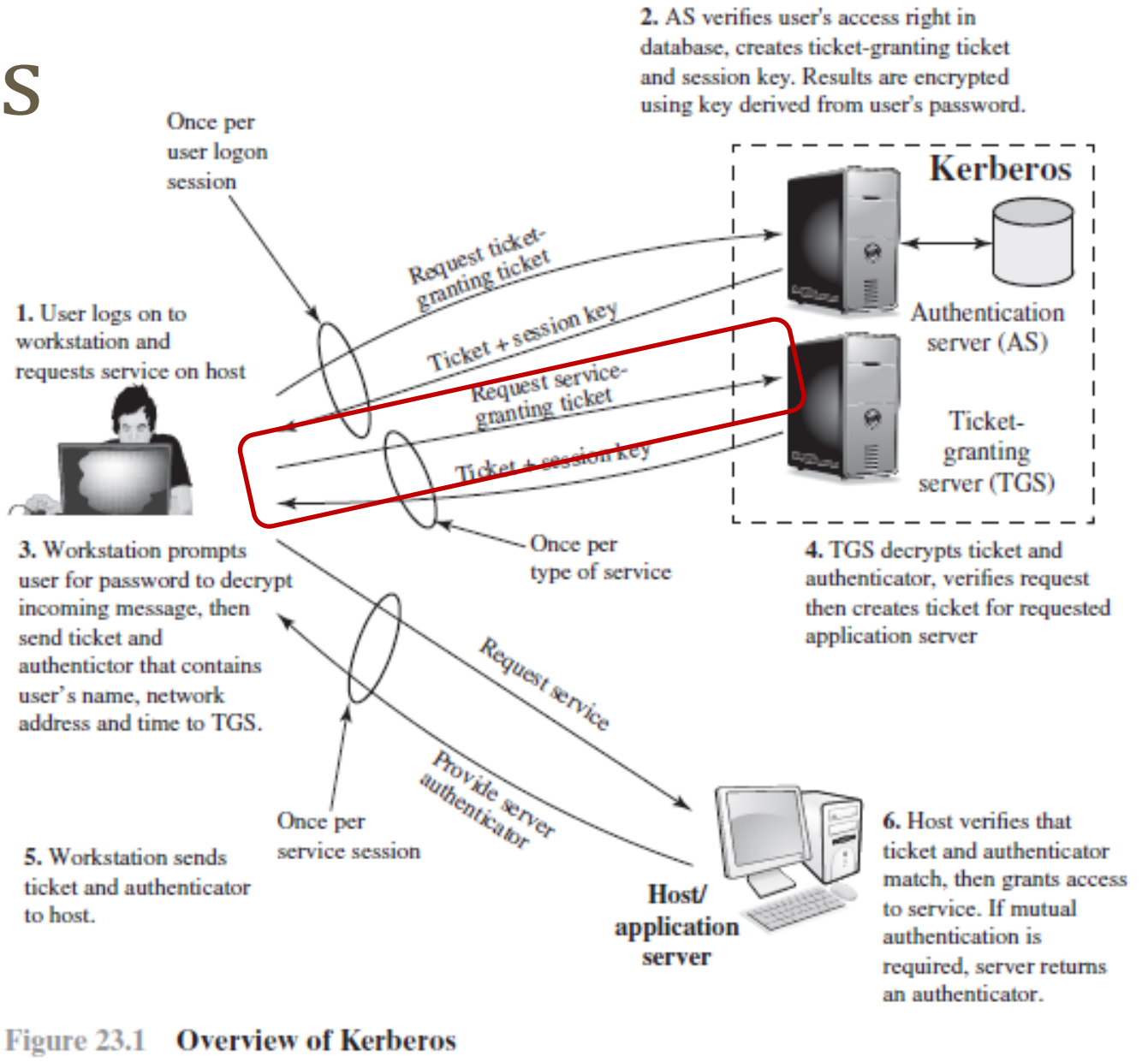


Figure 23.1 **Overview of Kerberos**

- The client then sends a request message for **SGT (Service Granting Ticket)** which includes (ID of server V, TGT, authenticator) to TGS.

Kerberos

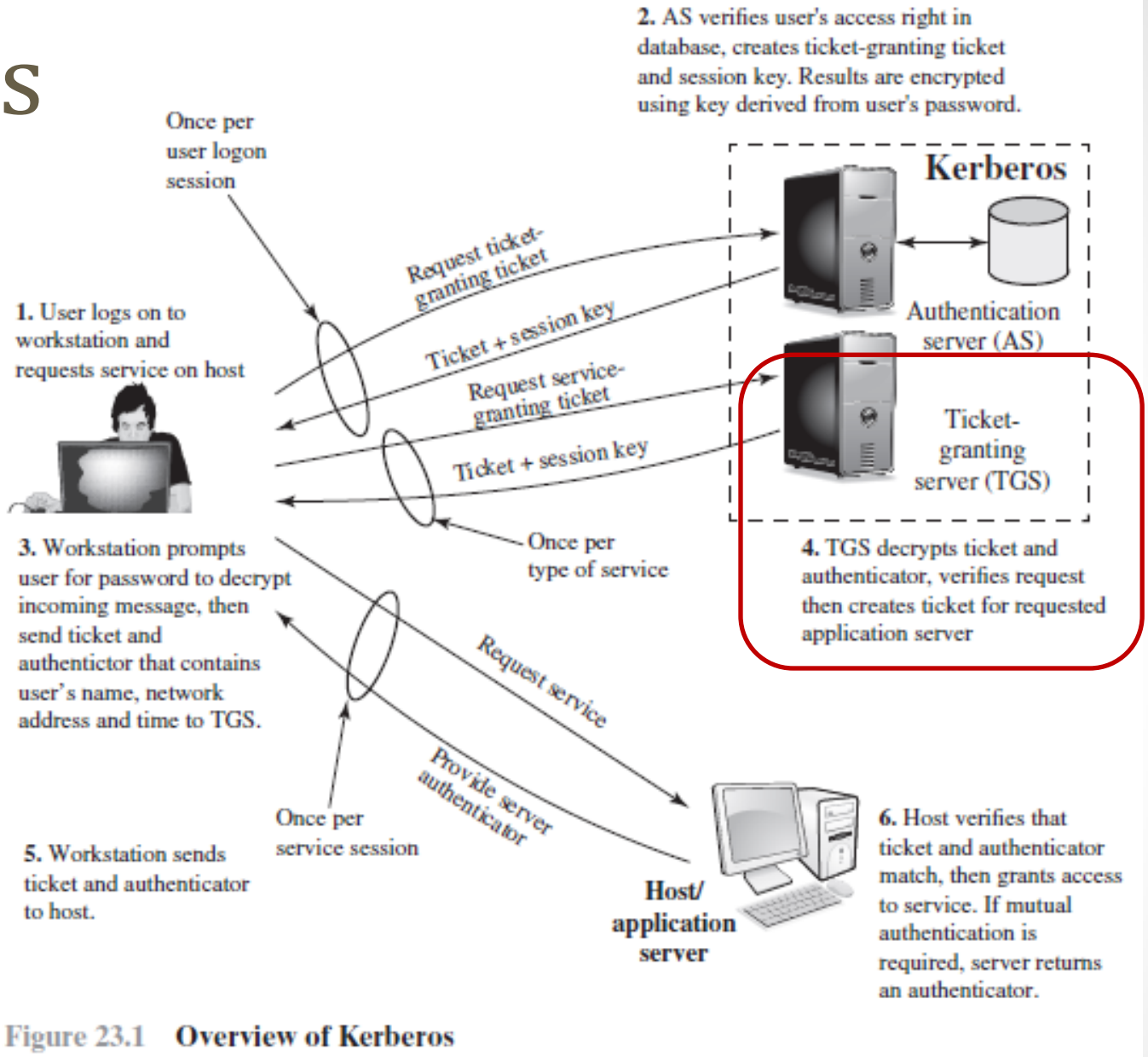


Figure 23.1 **Overview of Kerberos**

- The TGS decrypts the incoming TGT (**remember, the ticket is encrypted by a key known only to the AS and the TGS**) and verifies the success of the decryption by the presence of its own ID.

Kerberos

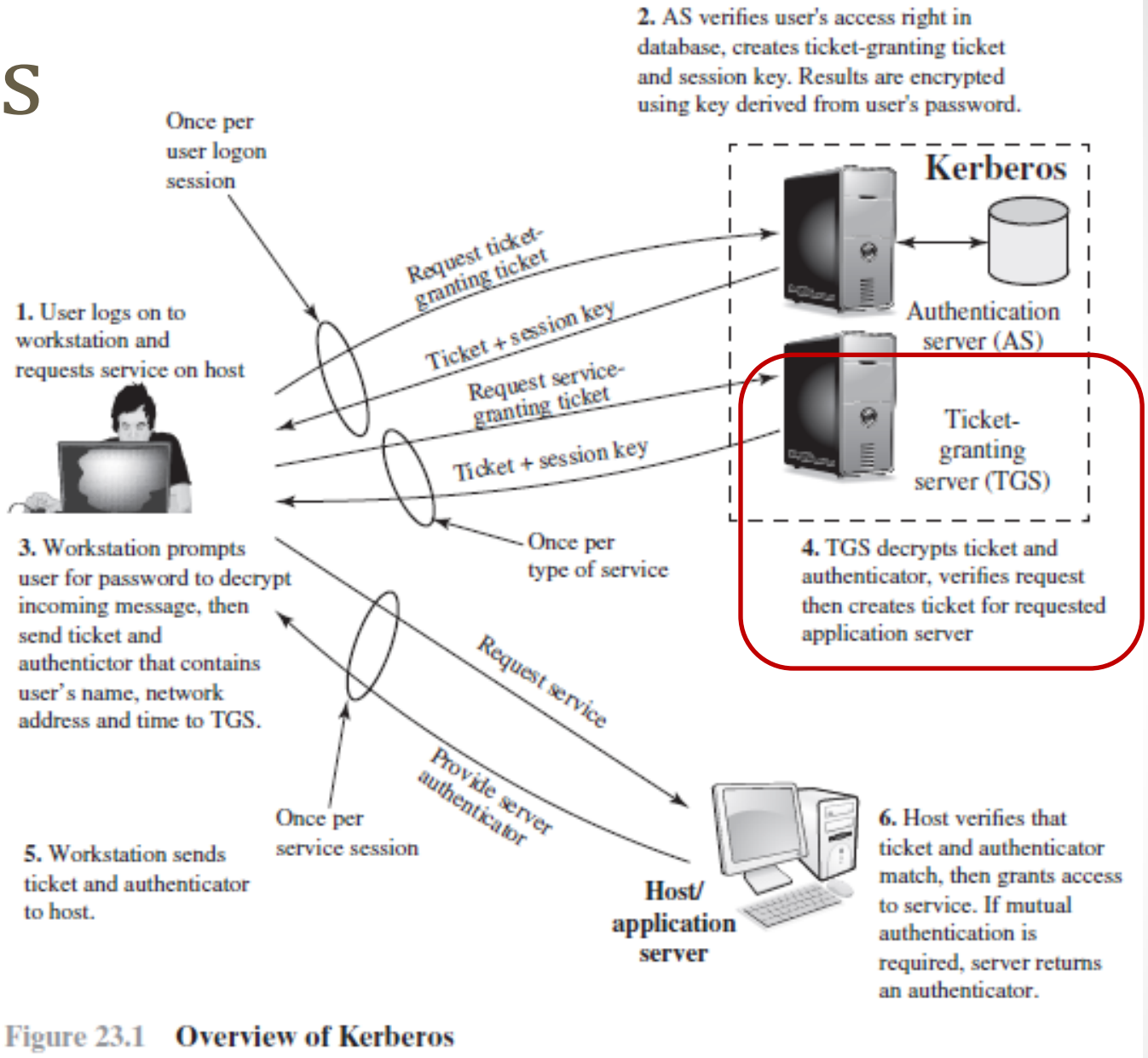


Figure 23.1 **Overview of Kerberos**

- The TGS also checks that the lifetime has not expired.

Kerberos

- **Another Notes:**
  - The lifetime associated with the TGT may cause the problem.
  - If this lifetime is very short (e.g., minutes), then the user will be repeatedly asked for a password.
  - If the lifetime is long (e.g., hours), then an opponent has a greater opportunity for replay.
  - **To get around this problem, the AS has provided a session key to both - the client and the TGS.**
  - **Also the same session key is there in TGT and the TGT was encrypted using the key shared by the AS and TGS.**
  - In the message to the TGS requesting a service-granting ticket, the client includes an **authenticator encrypted with the session key, which contains the ID and address of the user and a timestamp.**

Kerberos

- **Another Notes:**
  - Unlike the ticket, which is reusable, the authenticator is intended for use only once and has a very short lifetime.
  - Now, TGS can decrypt the ticket with the key that it shares with the AS.
  - This ticket indicates that user X has been provided with the session key.
  - TGS uses the session key to decrypt the authenticator.
  - The TGS can then check the name and address from the authenticator with that of the ticket and with the network address of the incoming message.
  - If all match, then the TGS is assured that the sender of the ticket is indeed the ticket's real owner.

Kerberos

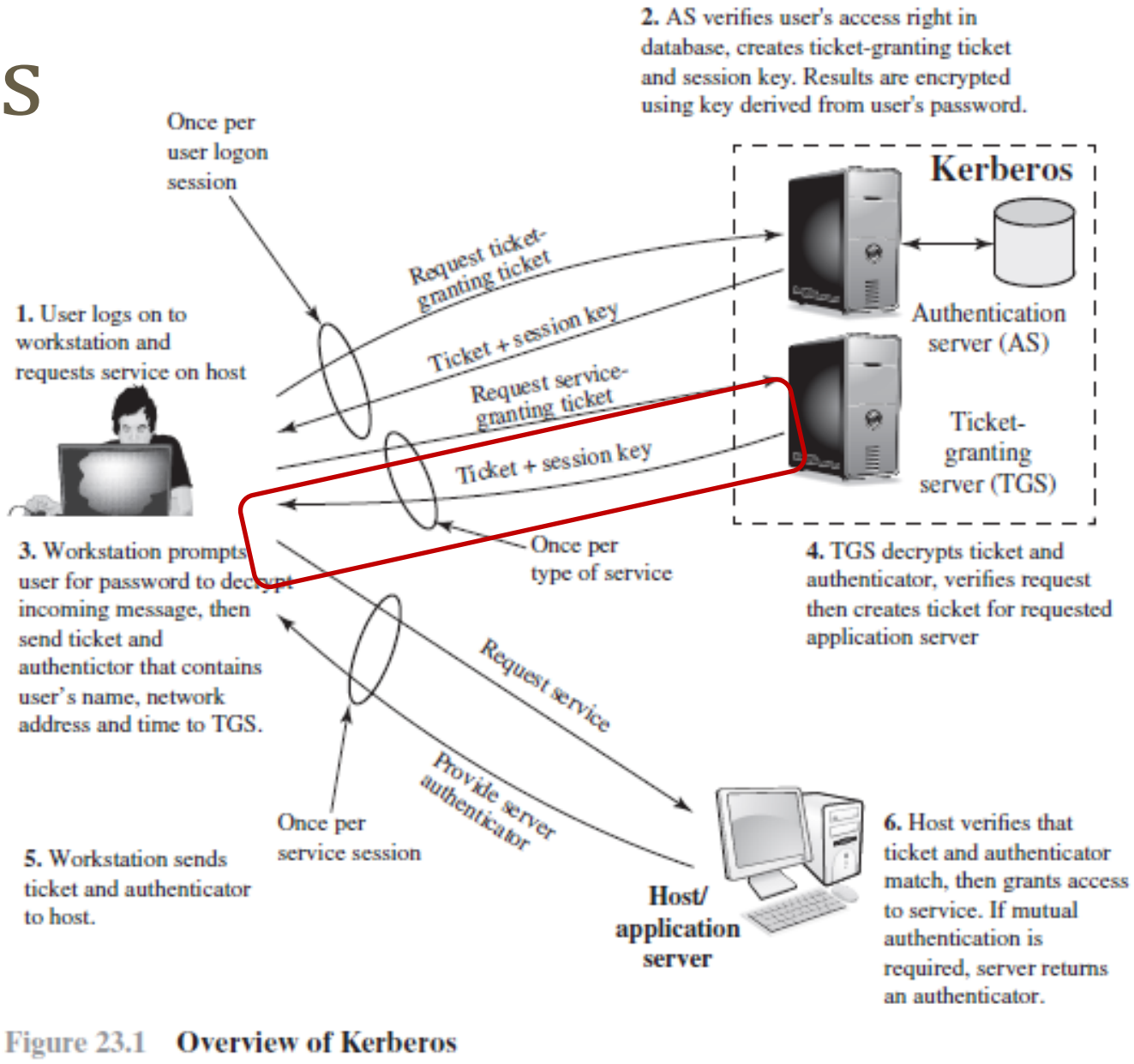


Figure 23.1 **Overview of Kerberos**

- The TGS sends **a Service Granting Ticket(SGT) and a new session key** to the client. The entire message is encrypted with the old session key, so that only the client can recover the message.

Kerberos

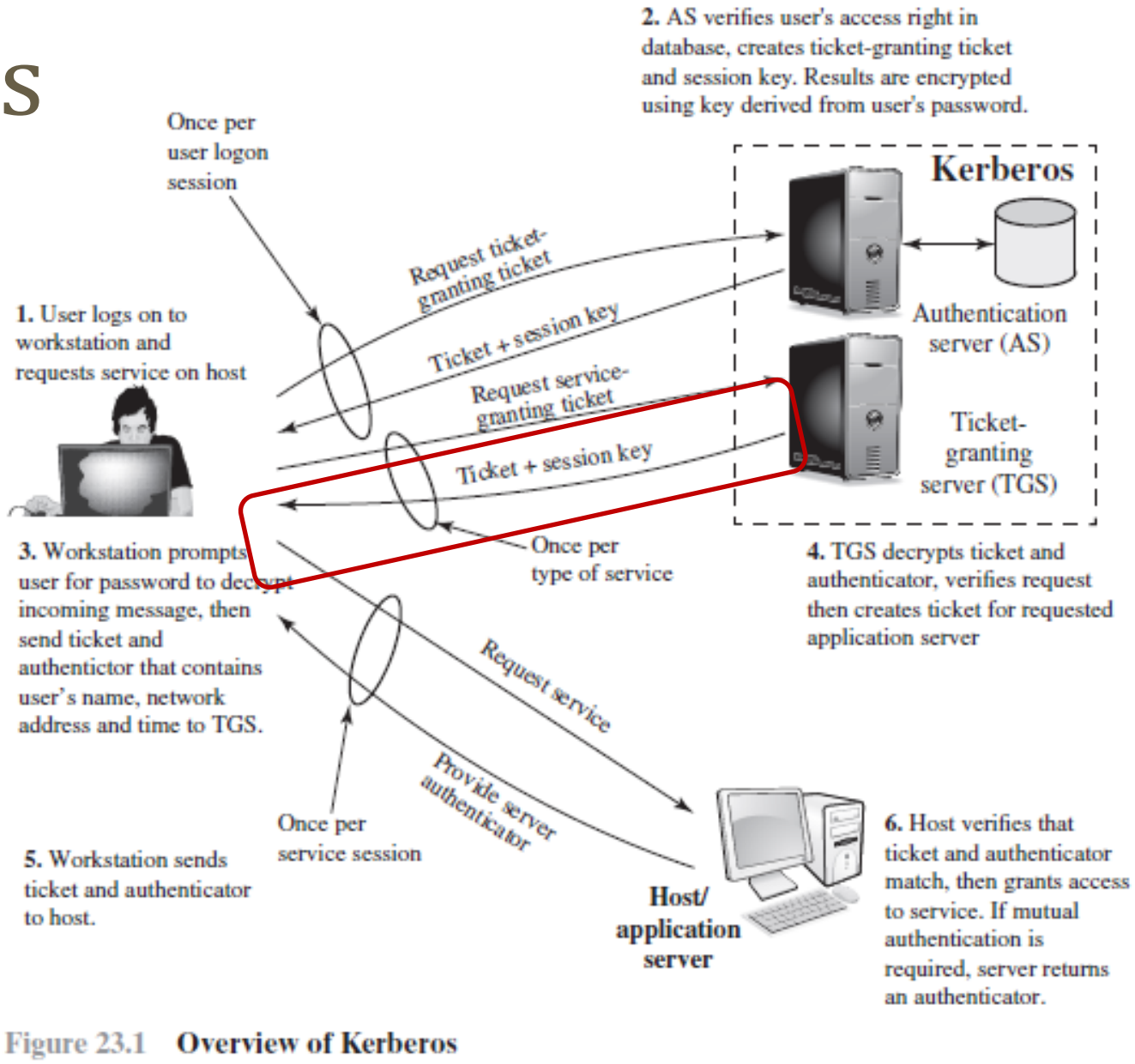


Figure 23.1 **Overview of Kerberos**

- The SGT is encrypted with a secret key shared only by the TGS and server V. The client now has a reusable service-granting ticket for V.

Kerberos

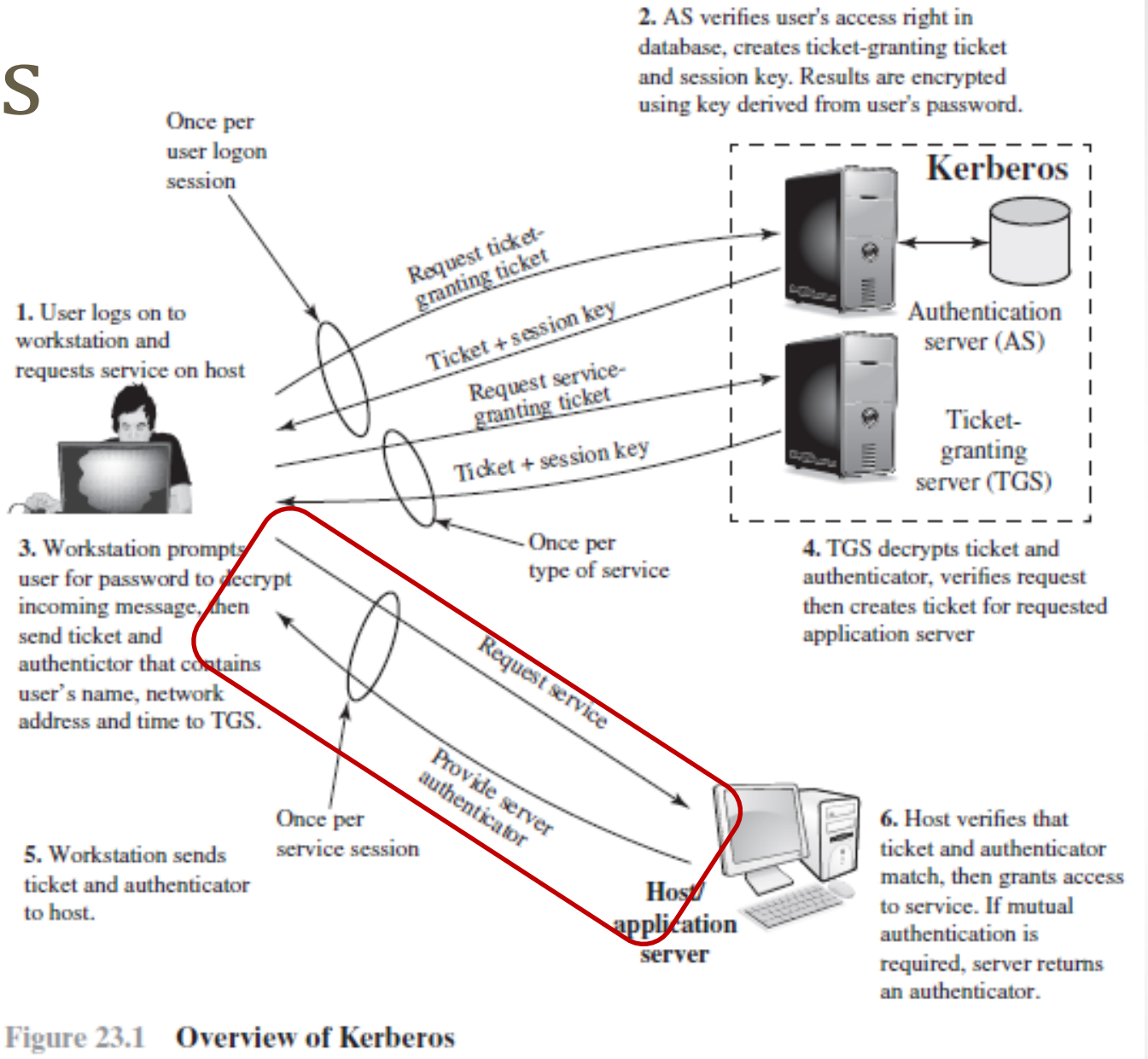


Figure 23.1 **Overview of Kerberos**

- Each time user X wishes to use service V, the client can then send this ticket plus an authenticator to server V. The authenticator is encrypted with the new session key.

Kerberos

- **Another Notes:**
  - If mutual authentication is required, the server can reply with the value of the timestamp from the authenticator, incremented by 1, and encrypted in the session key.
  - The client can decrypt this message to recover the incremented timestamp.
  - Because the message was encrypted by the session key, the client is assured that it could have been created only by V.
  - The contents of the message assures C that this is not a replay of an old reply.

Kerberos Realms and Multiple Kerberi

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

Kerberos Realms and Multiple Kerberi

- Kerberos with Multiple realms

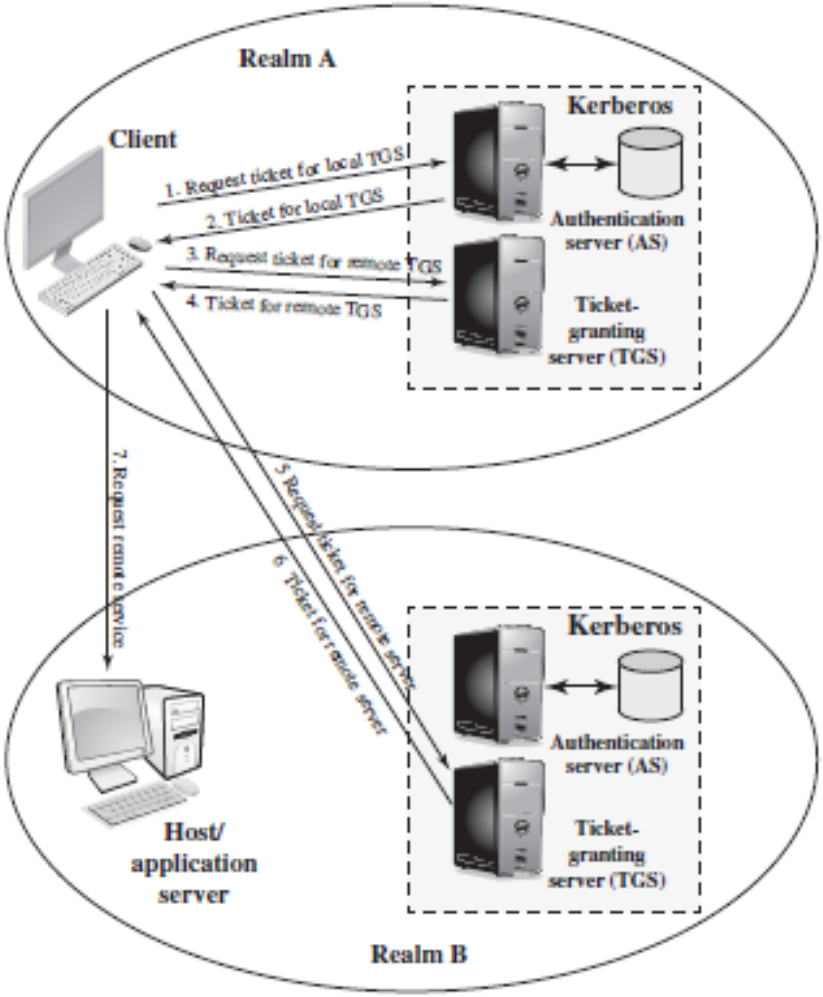


Figure 23.2 **Request for Service in Another Realm**

Kerberos Realms and Multiple Kerberi

- In case of multiple realms,
  - users in one realm may need access to servers in other realms.
  - 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.**
- **Here,** 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
  - the Kerberos server in one realm trust the Kerberos server in the other realm to authenticate its users.
  - the participating servers in the second realm must also be willing to trust the Kerberos server in the first realm.

- Kerberos with Multiple realms

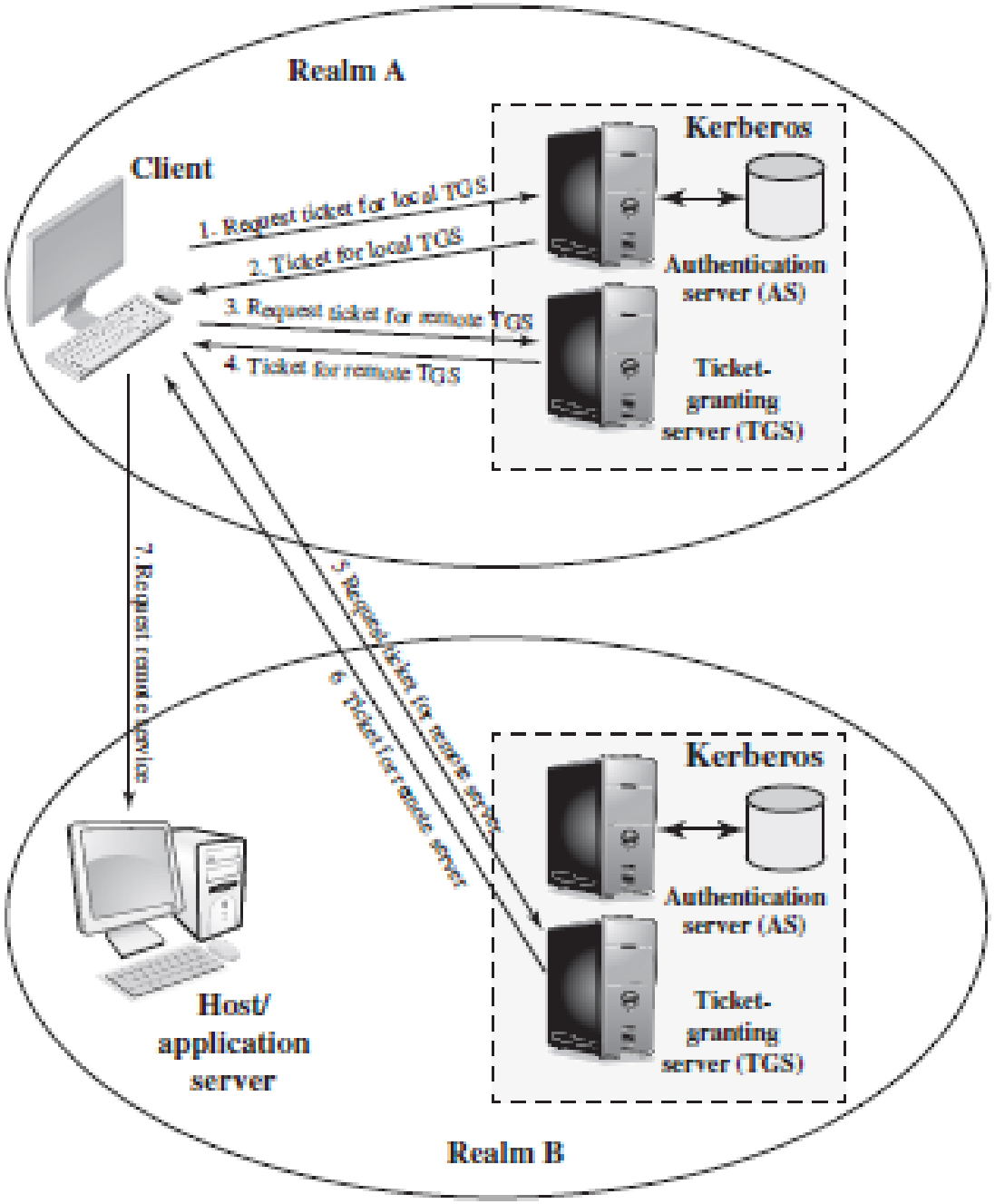


Figure 23.2 **Request for Service in Another Realm**

Kerberos Realms and Multiple Kerberi

- Problem:
  - It does not scale well to many realms.
  - If there are N realms, then there must be $N(N-1)/2$ secure key exchanges so that each realm can interoperate with all other Kerberos realms.

X.509 – A standard for Public Key Certificate

Why Public-Key Certificate?

- In public-key cryptography, any participant can send his or her public key to any other participant or broadcast the key to the community.
- **Weakness** :
  - Anyone can forge such a public announcement.
  - That is, an attacker could pretend to be Bob and send a public key to another participant or broadcast such a public key.
  - Until such time as Bob discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for Bob and can use the forged keys for authentication.
- **Public-key certificate offers solution of the issue discussed.**

Public-Key Certificate

- **A Public-Key certificate consists of a public key plus a user ID of the key owner, with the whole block signed by a trusted third party.**
- The third party is a **certificate authority (CA)** that is trusted by the user community, such as a government agency or a financial institution.
- A user can present his or her public key to the authority in a secure manner and obtain a signed certificate.
- The user can then publish the certificate.
- Anyone needing this user's public key can obtain the certificate and verify that it is valid by means of the attached trusted signature.

Public-Key Certificate

- **Steps to create and verify the Public-Key Certificate**
  1. User software (client) creates a pair of keys: one public and one private.
  2. Client prepares an unsigned certificate that includes the user ID and user's public key.
  3. User provides the unsigned certificate to a CA in some secure manner.
  4. CA creates a signature as follows
     a) CA uses a hash function to calculate the hash code of the unsigned certificate.
     b) CA encrypts the hash code with the CA's private key.
  5. CA attaches the signature to the unsigned certificate to create a signed certificate.
  6. CA returns the signed certificate to client.

Public-Key Certificate

- **Steps to create and verify the Public-Key Certificate**
  7. Client may provide the signed certificate to any other user.
  8. Any user may verify the certificate as follows:
     a) User calculates the hash code of certificate (not including signature).
     b) User decrypts the signature using CA's known public key.
     c) User compares the results of (a) and (b). If there is a match, the certificate is valid.

Public-Key Certificate

- Certificate creation and verification process

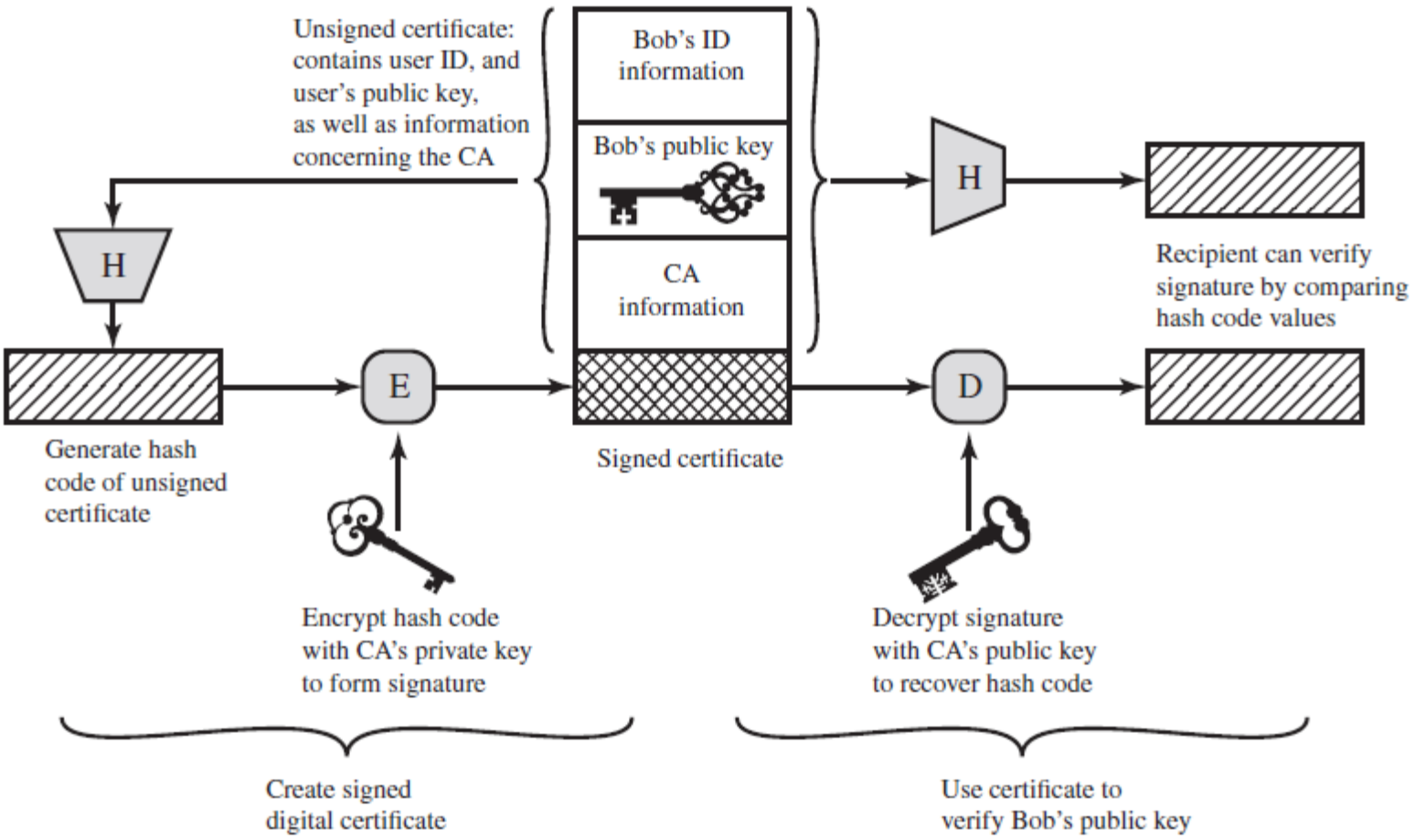


Figure 2.7 **Public-Key Certificate Use**

Public-Key Certificate

- **X.509 certificates**
  - Universally accepted standard for formatting public-key certificates.
  - X.509 certificates are used in most network security applications, including IP Security (IPsec), Transport Layer Security (TLS), Secure Shell (SSH), and Secure/ Multipurpose Internet Mail Extension (S/MIME) as well as in e-Business applications.

Public-Key Certificate

- **Elements of X.509 certificate**

- **Version.** Which X.509 version applies to the certificate, indicating what data the certificate must include.
- **Serial number.** The CA creating the certificate must assign it a serial number that distinguishes the CA certificate from other certificates.
- **Algorithm information.** The signature algorithm the issuer uses to sign the certificate.
- **Issuer distinguished name.** The name of the entity issuing the certificate -- usually, the CA.

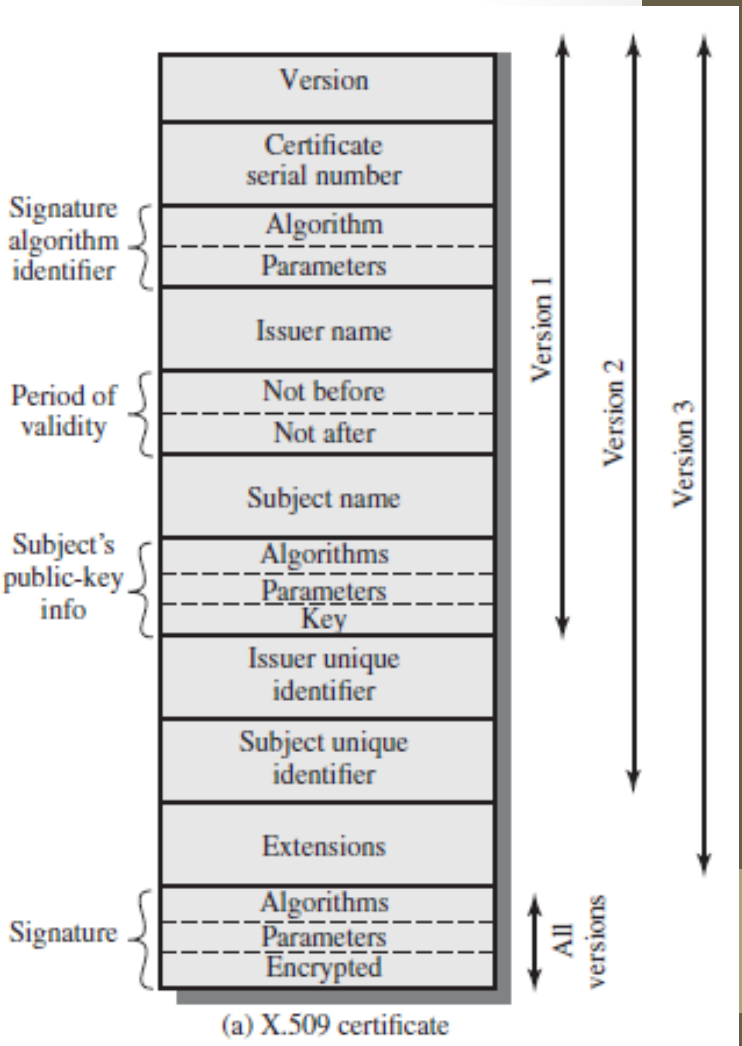


Figure 23.3 **X.509 Formats**

Public-Key Certificate

- **Elements of X.509 certificate**

- **Validity period of the certificate.** The start and end date, as well as the time the certificate is valid and can be trusted.
- **Subject distinguished name.** The name to which the certificate is issued.
- **Subject public key information.** The public key associated with the identity.
- **Extensions (optional).** Extensions have their own unique IDs, expressed as a set of values called an object identifier.

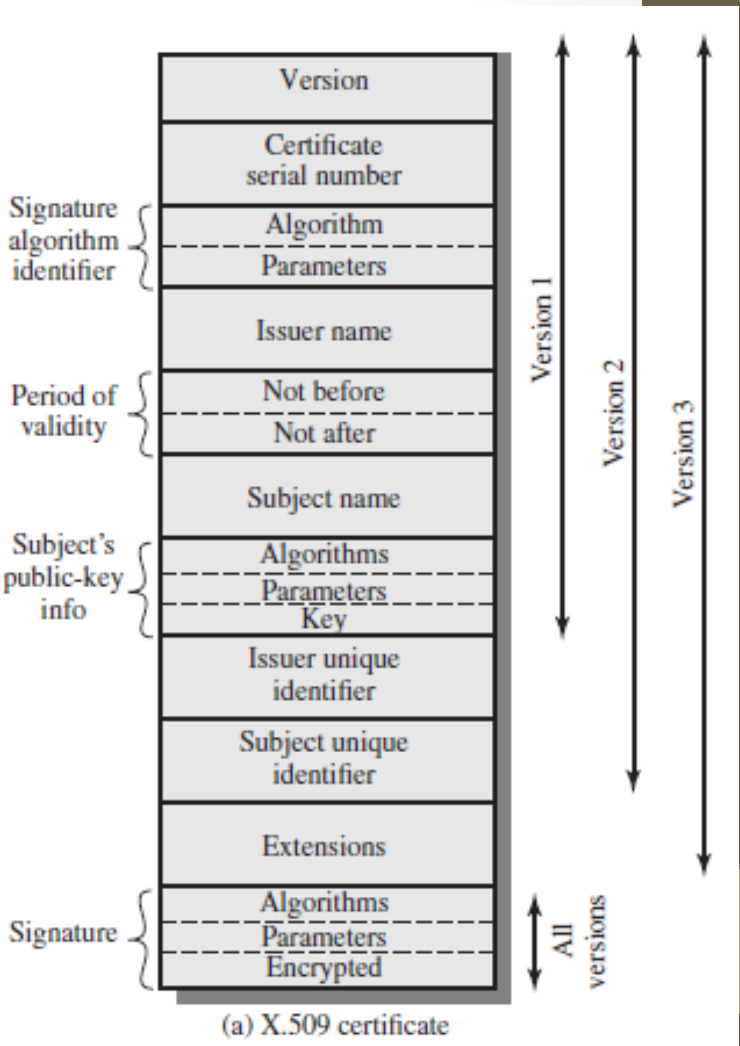


Figure 23.3 **X.509 Formats**

Public-Key Certificate

- **X.509 certificates**
  - One important extension specifies whether the certificate is that of a CA or not.
  - **A CA certificate is used only to sign other certificates**.
  - Otherwise, a normal certificate belongs to an "end-user" (or "end-entity"), and may be used for verifying server or client identities, signing or encrypting email or other content, signing executable code, or other uses in applications.
- The '**CA certificate' and 'end user certificate' are the most common form of X.509 certificates.**
- **Some variants of public-key certificates are also available.**

Public-Key Certificate

- **Variants of X.509 certificates**

1. **Conventional (long-lived) certificates:**
   - These are the CA and "end user" certificates.
   - They are typically issued for validity periods of months to years.
2. **Short-lived certificates:**
   - These are used to provide authentication for applications such as grid computing.(Grid computing is a computing infrastructure that combines computer resources spread over different geographical locations to achieve a common goal.)
   - They have validity periods of hours to days, which limits the period of misuse if compromised.

Public-Key Certificate

- **Variants of X.509 certificates**

3. **Proxy certificates:**
   - Also used for Grid Computing.
   - 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 create a credential to access resources in some environment, without needing to provide their full certificate and rights.

Certificate Authority List

- Few of the certificate authorities in India

Certifying Authorities	Website
e Mudhra	www.e-Mudhra.com
Capricorn	www.certificate.digital
Verasys	www.vsign.in
(n)Code Solutions	www.ncodesolutions.com

Public-Key Certificate

- **Variants of X.509 certificates**

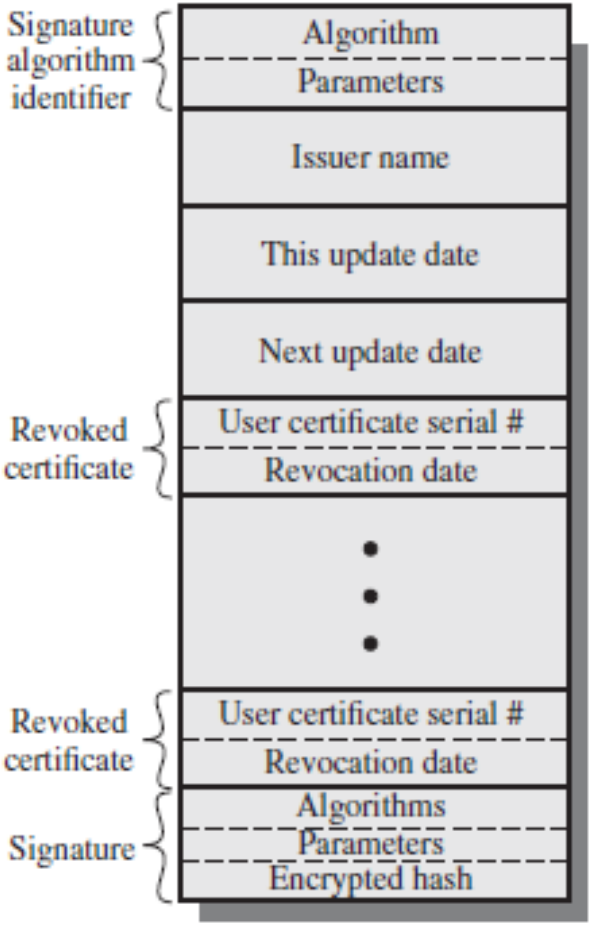
4. **Attribute certificates:**
   - They use a different certificate format, defined in RFC 5755, to link a user's identity to a set of attributes that are typically used for authorization and access control.
   - A user may have a number of different attribute certificates, with different sets of attributes for different purposes, associated with their main conventional certificate.

Certificate Revocation

- **Before using any certificate, an application must check its validity, and ensure that it was not revoked before it expires.**
- **Certificate Revocation** is required in the case when the user's key is compromised, or because an upgrade in the user's software requires the generation of a new key.
  - The X.509 standard defines a **Certificate Revocation List (CRL),** signed by the issuer.
  - Each revoked certificate entry contains a serial number of a certificate and the revocation date for that certificate.
  - As per X.509 standard, on receiving a certificate, an application has to check the certificate against the current CRL for its issuing CA.

Certificate Revocation

- Certificate Revocation List (CRL)



(b) Certificate revocation list

PKI (Public Key Infrastructure)

PKI (Public Key Infrastructure)

- It is the **set of hardware, software, people, policies, and procedures** needed to create, manage, store, distribute, and revoke digital certificates based on **asymmetric cryptography**.
  - PKI uses a pair of keys (**Private/Public**) to achieve the underlying security service.
  - Since the public keys are in open domain, they are likely to be abused.
  - It is, thus, necessary to establish and maintain some kind of trusted infrastructure to manage these keys.
- So X.509 Certificate came into existence where CA issues certificate to user from his public key.

PKI (Public Key Infrastructure)

- A client (once get the certificate)can verify that public key of server(or another party) by verifying the certificate.
- **To verify the certificate, the client use the public key of CA (as certificate is digitally signed).**
  - For authenticity of the CA's public key, another certificate (which is signed by a parent CA) is required.
  - Such process makes hierarchy of CAs.
  - As per the X.509 standard there would be a single internationally specified hierarchy of government regulated CAs.
  - However, the current X.509 PKI implementations came with a large list of CAs and their public keys, known as a "**trust store**".

PKI (Public Key Infrastructure)

- 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.
- There are many problems with this model of a PKI.
  - User doesn’t know what he has to do when there is problem in certificate verification.
  - A common assumption that all of the CAs in the “trust store” are equally trusted, equally well managed, and apply equivalent policies. (Many systems are compromised in past because of this assumption).
  - Various web browsers and operating systems, use different “trust stores,” and hence present different security views to users.
- Several proposals exist to improve the practical handling of X.509 certificates.

PKI (Public Key Infrastructure)

- **Public Key Infrastructure X.509 (PKIX)**
- The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has defined a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet.
- Elements of PKIX model
  - **End entity :** A user or server for whom the certificate is issued.
  - **Certificate authority:** who issues the certificate.
  - **Registration authority (RA):** handles end entity registration
  - **CRL Issuer and Repository :** that manage CRLs.

PKI (Public Key Infrastructure)

- **Public Key Infrastructure X.509 (PKIX)**

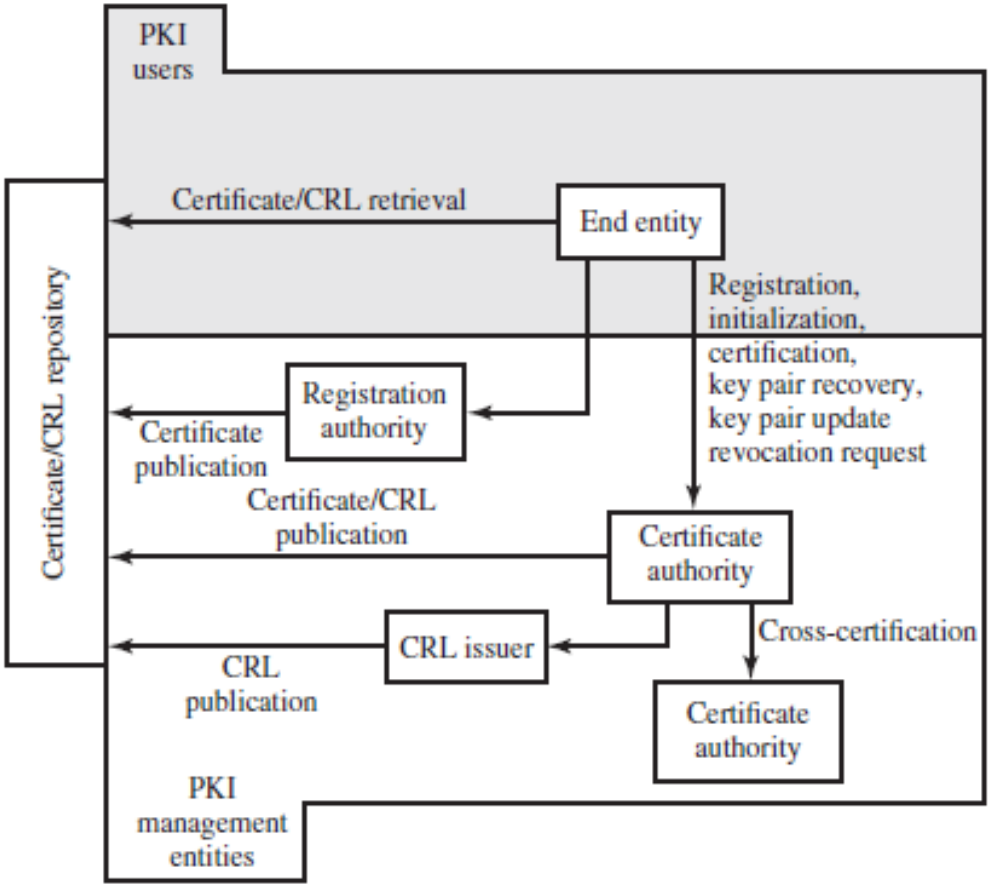


Figure 23.4 **PKIX Architectural Model**

PKI (Public Key Infrastructure)

- **PKIX working**
- The End–entity sends its certificate request to the Registration Authority (RA) for approval.
- If it is actually approved, it is forwarded to the Certification Authority (CA) for signing.
- The CA verifies the certificate request and if it passes the verification, it is signed and the Certificate is produced.
- To public the Certificate, the CA sends it to Certificate Repository for collection from the End–entity.
- Both the CA and RA are shown to deliver Certificates to the repository. Depending on the implementation, one of the two is chosen.
- For the issue of the revocation of the certificates, a similar course with the generation.
- **Note: the diagram shows all possible communications, regardless of the implementation decisions.**

PKI (Public Key Infrastructure)

- **PKIX working**
- For the issue of the revocation of the certificates, the End–entity asks the RA to have its Certificate revoked, the RA decides and possibly forwards it to the CA, the CA updates the revocation list and publishes it on the CRL repository.
- Finally, the End–entities can check the validity of a specific Certificate using an operational protocol.