Unit 4

NETWORK SECURITY
Users wish to access services on servers.

Three threats exist:

– User pretend to be another user.
– User alter the network address of a workstation.
– User eavesdrop on exchanges and use a replay attack.
Provides a centralized authentication server to authenticate users to servers and servers to users.

Relies on conventional encryption, making no use of public-key encryption.

Two versions: version 4 and 5

Version 4 makes use of DES
Kerberos Version 4

Terms:
- C = Client
- AS = authentication server
- V = server
- IDc = identifier of user on C
- IDv = identifier of V
- Pc = password of user on C
- ADc = network address of C
- Kv = secret encryption key shared by AS and V
- TS = timestamp
- || = concatenation
A Simple Authentication Dialogue

(1) C → AS: IDc || Pc || IDv
(2) AS → C: Ticket
(3) C → V: IDc || Ticket

Ticket = E_{K_v}[IDc || Pc || IDv]
Version 4 Authentication Dialogue

Problems:

- Lifetime associated with the ticket-granting ticket
- If too short \(\Rightarrow\) repeatedly asked for password
- If too long \(\Rightarrow\) greater opportunity to replay

The threat is that an opponent will steal the ticket and use it before it expires
Version 4 Authentication Dialogue

**Authentication Service Exchange: To obtain Ticket-Granting Ticket**

1. $C \rightarrow AS$: $IDc || IDtgs || TS1$
2. $AS \rightarrow C$: $E_{Kc} [K_{c,tgs} || IDtgs || TS2 || Lifetime2 || Tickettgs]$

**Ticket-Granting Service Exchange: To obtain Service-Granting Ticket**

3. $C \rightarrow TGS$: $IDv || Tickettgs || Authenticatorc$
4. $TGS \rightarrow C$: $E_{Kc} [K_{c,v} || IDv || TS4 || Ticketv]$

**Client/Server Authentication Exchange: To Obtain Service**

5. $C \rightarrow V$: $Ticketv || Authenticatorc$
6. $V \rightarrow C$: $E_{Kc,v}[TS5 +1]$
1. User logs on to workstation and requests service on host.

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

5. Workstation sends ticket and authenticator to server.

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.

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.

6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.
Request for Service in Another Realm

Figure 4.2  Request for Service in Another Realm
Difference Between Version 4 and 5

- Encryption system dependence (V.4 DES)
- Internet protocol dependence
- Message byte ordering
- Ticket lifetime
- Authentication forwarding
- Interrealm authentication
X.509 Authentication Service

- Distributed set of servers that maintains a database about users.
- Each certificate contains the public key of a user and is signed with the private key of a CA.
- Is used in S/MIME, IP Security, SSL/TLS and SET.
- RSA is recommended to use.
X.509 Formats
Typical Digital Signature Approach

(b) Using public-key encryption
Obtaining a User’s Certificate

Characteristics of certificates generated by CA:

– Any user with access to the public key of the CA can recover the user public key that was certified.

– No part other than the CA can modify the certificate without this being detected.
SSL (Secure Socket Layer)

transport layer security service
originally developed by Netscape
version 3 designed with public input
subsequently became Internet standard known as TLS (Transport Layer Security)
uses TCP to provide a reliable end-to-end service
SSL has two layers of protocols
SSL Architecture

- SSL Handshake Protocol
- SSL Change Cipher Spec Protocol
- SSL Alert Protocol
- HTTP

SSL Record Protocol

TCP

IP
SSL Architecture

SSL session
- an association between client & server
- created by the Handshake Protocol
- define a set of cryptographic parameters
- may be shared by multiple SSL connections

SSL connection
- a transient, peer-to-peer, communications link associated with 1 SSL session
SSL Record Protocol

**confidentiality**
- using symmetric encryption with a shared secret key defined by Handshake Protocol
  - IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
- message is compressed before encryption

**message integrity**
- using a MAC with shared secret key
- similar to HMAC but with different padding
SSL Change Cipher Spec Protocol

- one of 3 SSL specific protocols which use the SSL Record protocol
- a single message
- causes pending state to become current
- hence updating the cipher suite in use
SSL Alert Protocol

- conveys SSL-related alerts to peer entity
- severity
  - warning or fatal
- specific alert
  - unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
  - close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- compressed & encrypted like all SSL data
SSL Handshake Protocol

allows server & client to:
- authenticate each other
- to negotiate encryption & MAC algorithms
- to negotiate cryptographic keys to be used

comprises a series of messages in phases
- Establish Security Capabilities
- Server Authentication and Key Exchange
- Client Authentication and Key Exchange
SSL Handshake Protocol

Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

Phase 4
Change cipher suite and finish handshake protocol.

Note: Shaded transfers are optional or situation-dependent messages that are not always sent.
Secure Electronic Transactions (SET)

- open encryption & security specification
- to protect Internet credit card transactions
- developed in 1996 by Mastercard, Visa etc
- not a payment system
- rather a set of security protocols & formats
  - secure communications amongst parties
SET Components

Cardholder

Certificate Authority

Issuer

Payment Network

Acquirer

Merchant

Payment Gateway
SET Transaction

1. customer opens account
2. customer receives a certificate
3. merchants have their own certificates
4. customer places an order
5. merchant is verified
6. order and payment are sent
7. merchant requests payment authorization
8. merchant confirms order
9. merchant provides goods or service
10. merchant requests payment
Dual Signature

customer creates dual messages
  – order information (OI) for merchant
  – payment information (PI) for bank

neither party needs details of other

but **must** know they are linked

use a dual signature for this
  – signed concatenated hashes of OI & PI
Purchase Request – Customer

- **PI** = Payment Information
- **OI** = Order Information
- **PIMD** = PI message digest
- **OIMD** = OI message digest
- **E** = Encryption (RSA for asymmetric; DES for symmetric)
- **K_S** = Temporary symmetric key
- **KU_b** = Bank's public key-exchange key
Purchase Request – Merchant

Request Message

- Order Information (OI)
- OI message digest (OIMD)
- Payment Order message digest (POMD)
- Decryption (RSA) (D)
- Hash function (SHA-1) (H)
- Customer's public signature key ($KU_c$)

Passed on by merchant to payment gateway:

- Digital Envelope
- PIMD
- OI
- Dual Signature
- Cardholder Certificate

Diagram:

- $H \rightarrow OIMD$
- $D \rightarrow POMD$
- $KU_c$

Compare
Purchase Request – Merchant

1. verifies cardholder certificates using CA sigs
2. verifies dual signature using customer's public signature key to ensure order has not been tampered with in transit & that it was signed using cardholder's private signature key
3. processes order and forwards the payment information to the payment gateway for authorization (described later)
4. sends a purchase response to cardholder
Payment Gateway Authorization

1. verifies all certificates
2. decrypts digital envelope of authorization block to obtain symmetric key & then decrypts authorization block
3. verifies merchant's signature on authorization block
4. decrypts digital envelope of payment block to obtain symmetric key & then decrypts payment block
5. verifies dual signature on payment block
6. verifies that transaction ID received from merchant matches that in PI received (indirectly) from customer
7. requests & receives an authorization from issuer
8. sends authorization to merchant for order fulfillment
Payment Capture

- merchant sends payment gateway a payment capture request
- gateway checks request
- then causes funds to be transferred to merchants account
- notifies merchant using capture response