Network Security and Management

Network Security

Confidentiality

Sender encrypts message
Receiver decrypts message
Only sender + receiver understand contents

Authentication

Sender confirms identity to receiver

Message integrity

Message not changed between sender and receiver

Service availability

Prevent Denial of Service (DoS) attacks

Friends and Enemies

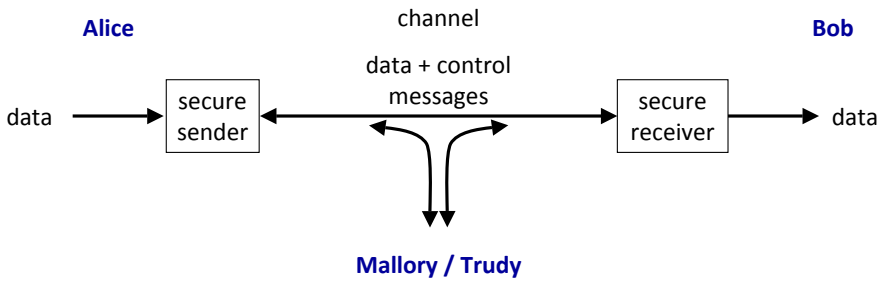
Bob + Alice

Legitimate system users
Communicate securely

Mallory + Trudy

Intruders
Intercept / delete / add / change messages

Philology of cryptology
Alice — party **A**
Bob — party **B**
Mallory — **mal**icious
Trudy — in**trud**er



Malicious Network Activity

Eavesdrop

Intercept legitimate messages for unauthorized reading

Message injection

Insertion of unauthorized messages into legitimate channel

Impersonation

Fake (spoof) source address in packet

Hijacking

Replacing legitimate sender or receiver in existing connection

Denial of Service (DoS)

Prevent access to service
Overloading server resources
Replacing legitimate server software / content

Cryptography

Plaintext

Original message m

Ciphertext

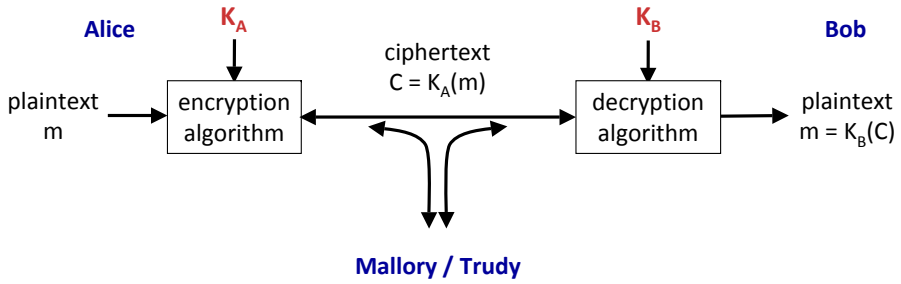
Message m encrypted with key K_A

Ciphertext C = $K_A(m)$

Decipher

Recover plaintext by decryption with key K_B

Plaintext m = $K_B(C) = K_B(K_A(m))$



Symmetric Key Cryptography

One secret key K

Encryption C = K(m)

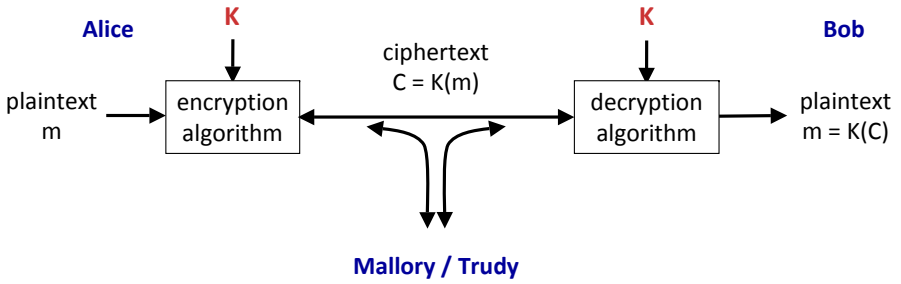
Decryption m = K(C) = K(K(m))

Alice + Bob share secret key K

Typical algorithm

Ciphertext = m ⊕ K

Ciphertext ⊕ K = (m ⊕ K) ⊕ K = m ⊕ (K ⊕ K) = m ⊕ 0 = m



Asymmetric Key Cryptography

Public key cryptography

Public encryption key K^+

$C = K^+(m)$

Private decryption key $K^- \neq K^+$

$m = K^-(C) = K^-(K^+(m))$

Rivest, Shamir and Adleman (RSA) algorithm

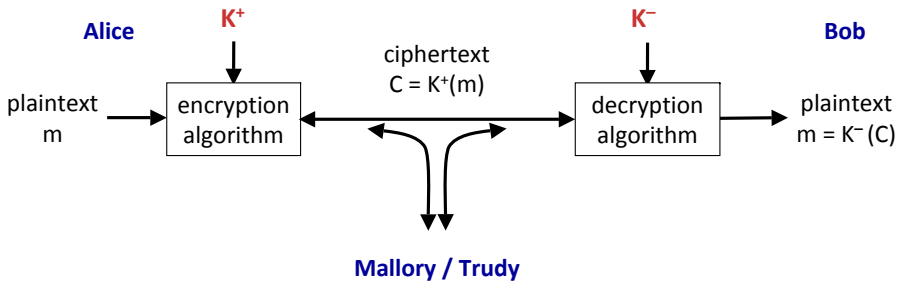
$m = K^-(K^+(m)) = K^+(K^-(m))$

Cannot obtain K^- from K^+ or K^+ from K^-

No shared key

Alice encrypts with Bob's published K^+

Bob decrypts with private K^-



Message Integrity

Message content not altered

Not corrupted / maliciously changed

Message digest

Public hash function h = H(m)

$m \neq m' \Rightarrow$ small probability of H(m) = H(m')

MD5 (RFC 1321) / SHA-1 (US NIST standard)

Alice

Computes h = H(m)

Sends (m,h)

Bob

Computes h = H(m) and compares with received h

H(m) = h ⇒ high probability of message integrity



Authentication

AAA Protocols

- Authentication
  - Check that user identity is authentic (as claimed)
  - Login, password, ...
- Authorization
  - Granting access / service rights
- Accounting
  - Tracking resource consumption by users
  - Logins, disk usage, ...

Common AAA systems

- Terminal Access Controller Access-Control System (TACACS)
- Kerberos
- Remote Authentication Dial In User Service (RADIUS)

Challenge / Response Authentication

Unencrypted

Challenge	Response	System Action
`Login:`	`Alice`	Check user name in database
`Password:`	`********`	Compare password with database

Encrypted

Challenge	Response	System Action
`Login:`	`Alice`	Check user name in database Set key K = password in database Set nonce R (one-time value) Send encrypted K(R)
`Password:`	`********` Set key K = `********` Decrypt R = K(K(R)) Send R + 1 unencrypted	 Correct value ⇒ correct password

Message Authentication Code (MAC)

Authentication key

Alice and Bob share secret key s

Authenticated message

- Alice
  - Computes h = H(m, s)
  - Sends (m, h)
- Bob
  - Computes h = H(m, s) and compares with received h
  - H(m,s) = h ⇒ high probability of message integrity + authenticity



Playback Attack

Authenticated message

- Alice sends Bob legitimate message with MAC
- Bob
  - Receives message
  - Confirms secret shared key
- Mallory
  - Captures Alice's message
  - Resends Alice's message to Bob
- Bob
  - Receives message
  - Confirms secret shared key
- Example
  - Electronic funds transfer (EFT)

Defense

Nonce (one-time use number)

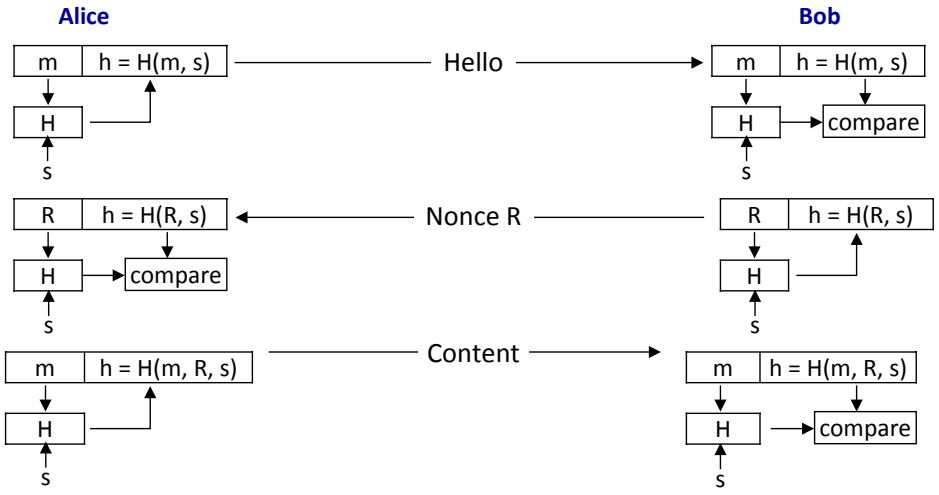
MAC with Nonce

Protected against replay

Nonce ⇒ correct 3-step handshake

Hello → R → Content

No 3-step handshake ⇒ ignore content message



Digital Signature

MAC with public key cryptography

Exploits RSA symmetry

$m = K^- (K^+ (m)) = K^+ (K^- (m))$

Signed message

Bob

Computes hash h = H(m)

Encrypts $ch = K^- (h)$ using private key K^- (only Bob can encrypt)

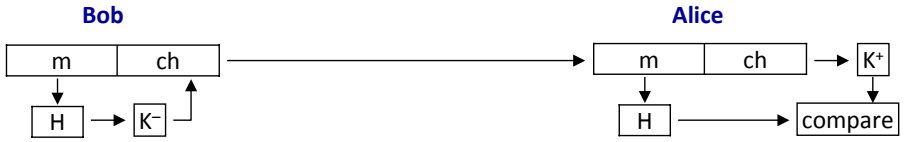
Sends (m, ch)

Alice

Decrypts $h = K^+ (ch)$ using Bob's public key K^+

Computes H(m) and compares with h

h = H(m) ⇒ message hash was signed using Bob's private key



Public Key Certification

Vulnerability of digital signature

Alice

Receives (m, ch) from Trudy calling herself "Bob"

Requests public key from "Bob"

Confirms m sent by "Bob"

Question

Is "Bob" really Bob

Certification Authority (CA)

Trusted organization

Certifies real-world identity of user

Shifts question to

Reliability of CA

Deviousness of intruder

Issues certificate to user

Certificate = real-world user information + user public key K^+

Digitally signed with CA private key K^-_{CA} (only CA can encrypt)

Certified Message

Bob

Calculates hash h = H(m)

Encrypts $ch = K^- (h)$ using private key K^- (only Bob can encrypt)

Sends (m, ch, certificate)

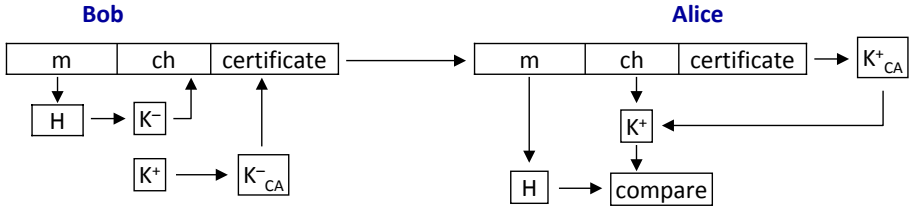
Alice

Decrypts $K^+ = K^+_{CA}$(certificate) using CA's public key K^+_{CA}

Decrypts $h = K^+ (ch)$ using Bob's public key K^+

Computes H(m) and compares with h

h = H(m) ⇒ message hash was encrypted using Bob's private key



Secure Sockets

Security layer between application and transport

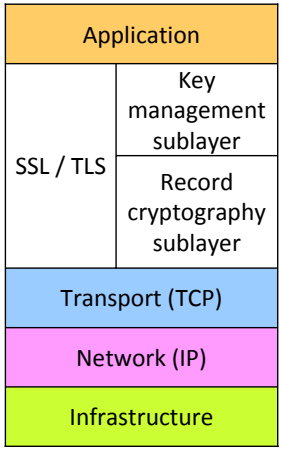
- Secure Sockets Layer (SSL)
  - Netscape browser (1993)
- Transport Layer Security (TLS)
  - RFC 2246 (1999 – 2011)

Services to application layer

- Confidentiality (RSA encryption)
- Authentication (certificates)
- Integrity (MAC hash protection)

Data fragmentation

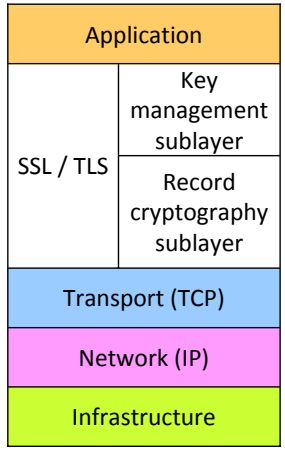
- Record = header + encrypted data fragment



Secure Sockets

Programmer perspective

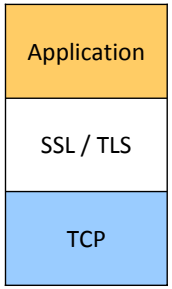
- Socket call invokes SSL / TLS instead of TCP
- SSL / TLS passes encrypted / protected data to TCP
- FTPS — FTP client / server over SSL / TLS
- HTTPS — client / server over SSL / TLS



SSL / TLS Overview

Session initialization

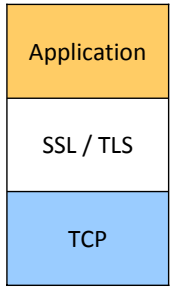
- Application
  - Client opens secure socket for channel to server
- SSL / TLS
  - Opens TCP session between client / server
  - Client / server negotiate certificates + keys + options
  - Establish secure channel over TCP
    - Confidentiality + authentication + integrity
  - Return socket descriptor to application



SSL / TLS Overview

Data exchange

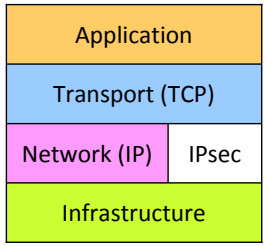
- Application
  - Sends data to open socket
- SSL / TLS
  - Client fragments app data
  - Encrypts data fragments + MAC
  - Adds record header
  - Forwards records to TCP
  - Server receives records from TCP
  - Checks / removes record header
  - Decrypts data fragments + MAC
  - Reassembles fragments to app data



Network Layer Security

Security layer at network layer

IPsec — RFC 4301 (2005)



Service options for IP datagrams

	Authentication Header (AH)	Encapsulating Security Payload (ESP)
Confidentiality		√
Integrity	√	√
Authentication	√	√
Replay protection	√	√

Modes of operation

Transport mode

- IP data encrypted / authenticated
- IP header not modified / encrypted

Tunnel mode

- Encrypted / authenticated datagram = data field of new datagram

Virtual Private Network (VPN)

Private network

Dedicated physically separate infrastructure

- Switches / routers / cables

Most secure at highest cost

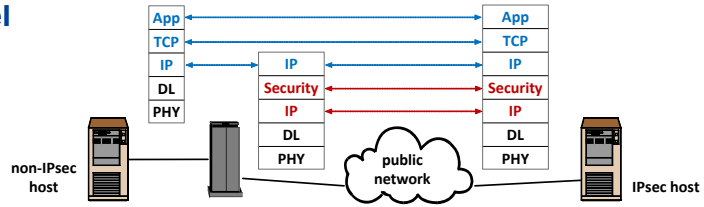
Virtual Private Network

Secure network implemented in shared public infrastructure

Security options

- Proprietary encryption software
- IPsec in tunnel mode
- TLS
- Traffic segregation in shared physical hardware

VPN tunnel



Security Association (SA)

Managing AH / ESP connections

Internet Security Association & Key Management Protocol (ISAKMP)

Security Association (SA)

Establish SA from sending to receiving entity before sending data

Simplex SA

- Each direction sets up separate SA

Connection-oriented stateful mechanism over IP

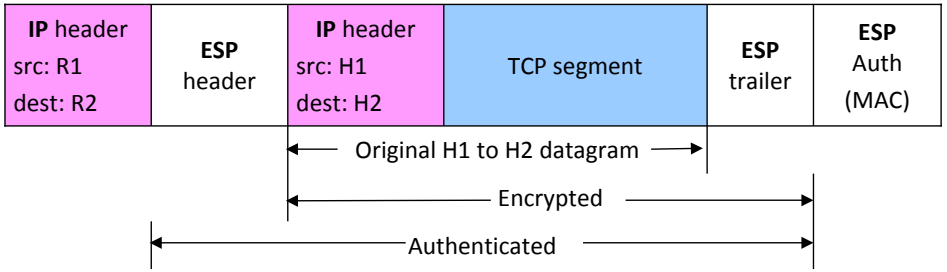
Example — SA for R1 to R2

R1 state table entries

- 32-bit SA identifier: Security Parameter Index (SPI)
- Source / destination interfaces — 200.168.1.105 / 193.68.2.23
- Encryption method / key
- Integrity check method / authentication key



IP Encapsulation



ESP header

Security Parameter Index (SPI)

SEQ

ESP trailer

Padding (for block cypher) + length of padding

Next header — protocol in original datagram



Firewall

Software / hardware based filter

- Blocks packets based on protocol criteria
- Intended to prevent
  - Unauthorized access to internal network from WAN
  - Unauthorized access to WAN by internal software

Stateless packet filtering

- Block packets based on
  - Source / destination IP address
  - TCP/UDP source / destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits (prevent any incoming TCP connections)

Stateful packet filtering

- Track status of TCP connections from SYN to FIN
  - Block packets inconsistent with TCP state machine
  - Block inactive connections

Intrusion Detection Systems (IDS)

Deep packet inspection

- Compare packet contents with database of virus / attack signatures

Correlations among multiple packets

- Port scanning
  - TCP SYN sequence of ports — map open ports
- Network mapping
  - Ping sequence IP addresses — map responding hosts
- Denial of Service (DoS) attack
  - Replayed packet to overload server

Statistical anomaly

- Compare traffic with statistical norms
  - Bandwidth demand
  - Protocols and service ports
  - User access
  - File updates

Data Management Zone (DMZ)

Term of military origin

- DMZ — Demilitarized Zone
- Border area between military powers
- Military activity not permitted

Network DMZ

- Border zone of AS (perimeter network)
- Unrestricted access from WAN

DMZ services

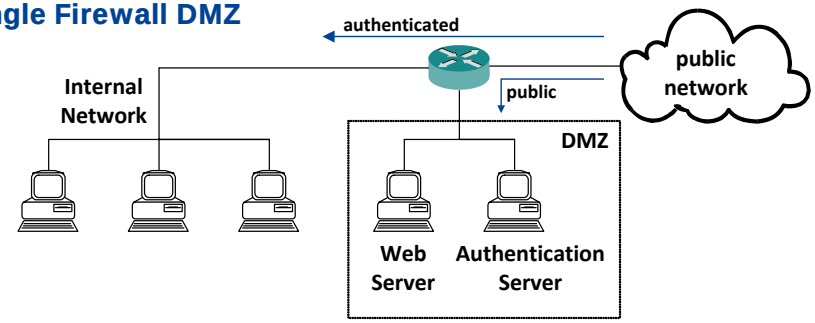
- Public web server
- Other unsecured services
  - Incoming mail server
  - Anonymous FTP server
- Secure authentication server to internal network

Reported intrusions to Pentagon network (1995)

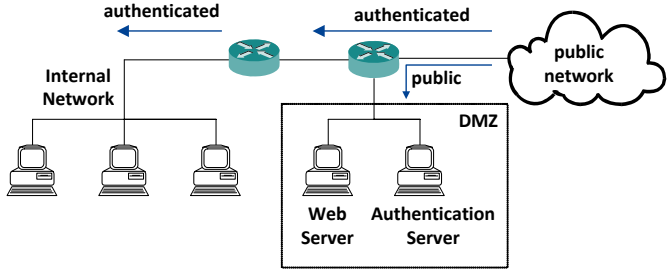
- DMZ — 250,000
- Internal — 0

DMZ Configurations

Single Firewall DMZ



Double Firewall DMZ



Network Management

OAM

- Operation — service availability and statistics monitoring
- Administration — Network resource accounting / assignment
- Maintenance — Repairs, upgrades

Provisioning

- Resource configuration to support services

Typical management tasks

- Resource
  - Controlling, planning, allocation, deployment, coordination, monitoring
- Traffic routing, load balancing, performance management
- Security management
- Fault management
- Account management

Management tools

- Monitoring agents installed on infrastructure
- Activity logs
- Sniffers

Simple Network Management Protocol (SNMP)

Management agents

- Hardware / software feature of network infrastructure
- Permits remote management of device
  - Monitor device activity
  - Configure device parameters

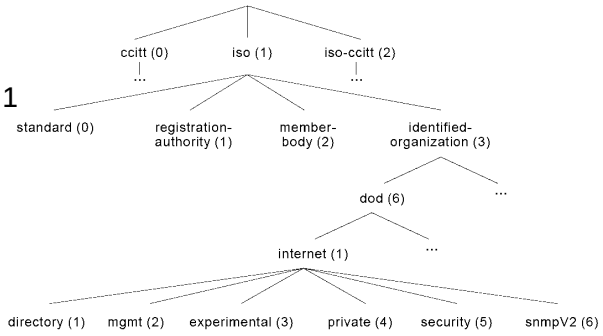
Role of SNMP

- Management Information Base (MIB)
  - Object-orient database record describing network entity
- Structure of Management Information (SMI)
  - Data definition language for MIB objects (subset of ASN.1)
- SNMP protocol
  - Manage entity by read (monitor) / write (control) of MIB information
  - Transaction mode
    - Manager request + agent response
  - Trap mode
    - Agent message on specified condition

Management Information Base (MIB)

MIB classification

- Tree hierarchy
- Internet MIB = 1.3.6.1



MIB

- Set of all MIB objects for managed device

MIB module

- Set of related MIB objects

Managed object

- Specific feature of managed device
- Uniquely identified by object ID
- Leaf level

Example — UDP MIB module

Object ID	Name	Object
1.3.6.1.2.1.7.1	UDPInDatagrams	Datagrams delivered at node
1.3.6.1.2.1.7.2	UDPNoPorts	Undeliverable datagrams — no server at port
1.3.6.1.2.1.7.3	UDInErrors	Undeliverable datagrams — other reasons
1.3.6.1.2.1.7.4	UDPOutDatagrams	Datagrams sent
1.3.6.1.2.1.7.5	udpTable	Entry for each port in use by app
1.3.6.1.2.1.7.5.1	udpEntry	
1.3.6.1.2.1.7.5.1.1	udpLocalAddress	udpTable entry — IP address
1.3.6.1.2.1.7.5.1.2	udpLocalPort	udpTable entry — Port

SNMP Message Types

Manager to agent

- GetRequest
  - Retrieve value of variable or list
- SetRequest
  - Change value of variable or list
- GetNextRequest
  - Discover available variables and their values
- GetBulkRequest
  - Multiple iterations of GetNextRequest

Agent to manager

- Response
  - Returns variable / acknowledgement for
    - GetRequest, SetRequest, GetNextRequest, GetBulkRequest
- Trap
  - Asynchronous notification of trapped event

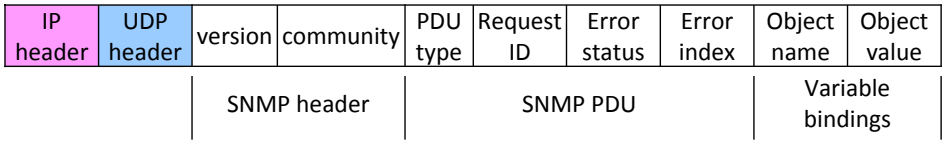
Manager to manager

- InformRequest
  - Asynchronous notification of trapped event

SNMP Packets

SNMP an application layer protocol

Messages send over UDP



Fields

- Version — SNMP version
- Community — environment for group of management systems
- PDU type — message type
- Request ID — associate requests / responses
- Error Status — indicates error type in response operation
- Error Index — associates error object instance
- Object name — name of object to set / monitor
- Object value — value of object