Application-Layer Security Comprehensive Exam Notes

Table of Contents

Email Security Overview

Basic Email Vulnerabilities

Email systems suffer from four fundamental security weaknesses:

1. **Lack of Confidentiality**
   - Messages sent in clear text over open networks
   - Stored on potentially insecure clients and servers
2. **Lack of Integrity**
   - Both message headers and content can be modified during transmission
   - No protection against tampering
3. **Lack of Authentication**
   - Sender identity is easily forgeable
   - No verification of message origin
4. **Lack of Non-Repudiation**
   - Senders can deny having sent messages
   - Recipients can deny having received messages

Email Structure (RFC 822)

- **Envelope**: Contains transmission and delivery information
- **Content**: Divided into header (Date, From, To, Subject) and body (actual message)
- Uses ASCII character format

- Limited to text-based content in basic form

PGP (Pretty Good Privacy)

Overview

- Developed by Philip Zimmermann
- Provides confidentiality and authentication for email
- Specified in RFCs 2015, 3156, and 4880
- Uses hybrid cryptography approach

PGP Services Summary

Function	Algorithms Used
Digital Signature	DSS/SHA or RSA/SHA
Message Encryption	CAST, IDEA, 3DES, AES, RSA, ElGamal
Compression	ZIP
Email Compatibility	Radix-64 conversion

PGP Operations

Authentication Only (Digital Signature)

1. Sender creates message and computes SHA-1 160-bit hash
2. Sender signs hash with private key (RSA/DSS)
3. Signature is prepended to message
4. Receiver verifies signature using sender's public key

Confidentiality Only (Encryption)

1. Sender generates random 128-bit session key
2. Message encrypted with session key using symmetric cipher
3. Session key encrypted with recipient's public key (RSA/ElGamal)
4. Both encrypted message and encrypted session key sent to recipient
5. Recipient decrypts session key with private key, then decrypts message

Combined Authentication and Confidentiality

- Create digital signature first

- Encrypt both message and signature
- Attach public-key encrypted session key
- **Order of operations**: Sign → Compress → Encrypt

Key Management

Key Generation Options

- RSA & RSA (sign and encrypt)
- DSA & ElGamal (sign and encrypt)
- RSA (sign only)
- DSA (sign only)

Key Identification

- **Key ID**: Least significant 64 bits of public key (PUa mod 2^64)
- Allows recipient to identify which key pair was used
- Avoids need to try all possible keys

Key Rings

Each user maintains two key rings:

1. **Private Key Ring**
   - Stores user's own private/public key pairs
   - Protected by user password/passphrase
   - Security depends on passphrase strength
2. **Public Key Ring**
   - Stores public keys of other users
   - Contains trust information and signatures

PGP Web of Trust

Unlike X.509 hierarchical trust model, PGP uses decentralized web of trust:

- **Owner Trust Field**: User-assigned trust level for other users
- **Key Legitimacy Field**: Automatically computed trust level for each public key
- **Signature Collection**: Public keys certified by multiple users
- **Trust Propagation**:

- Fully trusted user signatures validate keys
- Two partially trusted signatures can validate a key

Email Compatibility Features

Radix-64 Conversion

- Converts 8-bit binary data to printable ASCII characters
- Expands message size by 33%
- Necessary for email system compatibility

Segmentation and Reassembly

- Email systems often limit message size to 50,000 octets
- PGP automatically segments large messages
- Receiver reassembles segments before processing

S/MIME (Secure/Multipurpose Internet Mail Extensions)

Overview

- Security enhancement to MIME email
- Based on RSA Data Security technology
- Specified in RFCs 3369, 3370, 3850, and 3851
- Widely supported (Outlook, Mozilla, Mac Mail, Lotus Notes)

MIME Background

MIME extends RFC 822 capabilities:

- Supports non-textual content and non-ASCII character sets
- Enables long message transfer
- Introduces new header fields for format specification
- **Base64 encoding**: 24 data bits (3 bytes) → 4 ASCII characters (4 bytes)

S/MIME Functions

1. **Enveloped Data**: Encryption only
2. **Signed Data**: Digital signature only
3. **Signed and Enveloped**: Combined signature and encryption

S/MIME Algorithms

- **Digital Signatures**: DSS & RSA
- **Session Key Encryption**: ElGamal & RSA
- **Message Encryption**: AES, Triple-DES, others
- **MAC**: HMAC with SHA

S/MIME Processing

Uses PKCS (Public Key Cryptography Standard) objects containing:

- Original content
- All information needed for security processing
- Base64 encoded for email compatibility

EnvelopedData Processing

1. Generate random session key
2. Encrypt MIME entity with session key
3. Encrypt session key with recipient's public key
4. Create PKCS object with encrypted content and recipient info
5. Apply Base64 encoding

SignedData Processing

1. Hash MIME entity
2. Sign hash with sender's private key
3. Create PKCS object with original content and signature info
4. Apply Base64 encoding

Certificate Management

Uses X.509 v3 certificates with three trust levels:

Class	Identity Checks	Usage
1	Name/email check	Web browsing/email
2	+ enrollment/address check	Email, subscriptions, software validation
3	+ ID documents	E-banking/service access

Centralized Authentication

Distributed System Challenges

- Users access services on multiple servers across network
- Servers must authenticate users before providing services
- Need for scalable authentication solution

Centralized Authentication Server (AS)

- Manages all long-term user credentials
- Assists servers in client authentication
- Establishes session keys for secure communication
- Used in Windows environments (NTLM, Kerberos)

NTLM Authentication

NTLM Process

1. **Challenge**: Server sends random nonce (C) to user
2. **Response**: User encrypts challenge with hashed password: R = E_Hash(pwd)(C)
3. **Verification**: Server forwards encrypted response to Authentication Server
4. **Authentication**: AS verifies response and returns yes/no decision

NTLM Characteristics

- Used in Windows NT systems
- Relatively simple challenge-response mechanism
- Authentication Server maintains hashed passwords
- Shared keys between servers and AS for secure communication

---

Kerberos Authentication

Overview

- Named after three-headed dog guarding Hades in Greek mythology
- Developed at MIT as part of Project Athena
- Provides Authentication and Authorization Infrastructure (AAI)
- Used in Windows systems since Windows 2000

Kerberos Architecture

Three types of servers:

1. **Authentication Server (AS)**: Issues long-lifetime tickets
2. **Ticket Granting Server (TGS)**: Issues short-lifetime service tickets
3. **Service Servers**: Provide actual services

Kerberos V4 Protocol

Phase 1: Authentication Server Exchange

Step 1: Client → AS: ID_C, ID_tgs, TS_1 **Step 2**: AS → Client: E_Kc[K_c,tgs, ID_tgs, TS_2, Lifetime_2, Ticket_tgs]

- Ticket_tgs = E_Ktgs[K_c,tgs, ID_C, AD_C, ID_tgs, TS_2, Lifetime_2]

Phase 2: Ticket Granting Server Exchange

Step 3: Client → TGS: ID_V, Ticket_tgs, Authenticator_C

- Authenticator_C = E_Kc,tgs[ID_C, AD_C, TS_3]

Step 4: TGS → Client: E_Kc,tgs[K_C,V, ID_V, TS_4, Lifetime_4, Ticket_V]

- Ticket_V = E_Kv[K_c,v, ID_C, AD_C, ID_V, TS_4, Lifetime_4]

Phase 3: Service Server Exchange

Step 5: Client → Server: Ticket_V, Authenticator_C

- Authenticator_C = E_Kc,v[ID_C, AD_C, TS_5]

Step 6: Server → Client: E_Kc,v[TS_5 + 1]

Kerberos V4 Limitations

1. **Encryption**: Limited to DES only
2. **Ticket Lifetime**: 8-bit field limits to ~21 hours maximum
3. **Authentication Forwarding**: Cannot forward credentials to other hosts
4. **Double Encryption**: Unnecessary encryption in steps 2 and 4
5. **Dictionary Attacks**: Step 2 message vulnerable to offline password attacks

Kerberos V5 Improvements

- **Flexible Encryption**: Supports multiple encryption algorithms

- **Extended Lifetimes**: Uses actual start/end times instead of 8-bit field
- **Credential Forwarding**: FORWARDABLE flag enables credential delegation
- **Nonce Protection**: Prevents replay attacks
- **Realm Support**: Better inter-realm authentication

Key V5 Features

- **Options Field**: Requests specific ticket properties (PRE-AUTHENT, HW-AUTHENT, RENEWABLE, FORWARDABLE)
- **Times**: Flexible time specification (from, till, rtime)
- **Subkeys**: Optional sub-encryption keys for application sessions
- **Sequence Numbers**: Protection against replay attacks

Inter-Realm Authentication

- **Realm**: Kerberos server + clients + application servers
- **Requirements**:
  - Kerberos server knows all user credentials
  - Shared secret keys between Kerberos server and application servers
  - Shared secret keys between Kerberos servers in different realms

---

SSH (Secure Shell)

Overview

- Originally designed to replace insecure rsh, telnet utilities
- Provides secure remote administration
- General secure channel for network applications
- Applications need modification, but port forwarding helps

SSH-2 Architecture

Three-layer protocol stack:

1. **SSH Transport Layer Protocol**
   - Initial connection establishment
   - Server authentication
   - Secure channel setup via key exchange

2. **SSH Authentication Protocol**
    - Client authentication over secure transport channel
    - Methods: public key (DSS, RSA) or password
3. **SSH Connection Protocol**
    - Multiple logical connections over single transport channel
    - Efficiency through session reuse

SSH Security Goals

- **Server Authentication**: Based on server's host key pairs
- **Fresh Shared Secret**: Established through key exchange
- **Key Derivation**: Encryption keys, MAC keys, IVs derived from shared secret
- **Secure Negotiation**: Encryption, MAC, and compression algorithm selection

SSH Transport Layer Protocol

Key Exchange (Diffie-Hellman)

1. **Client**: Generates random x_c, computes $y_c = g^{x_c} \pmod p$, sends y_c
2. **Server**: Generates random x_s, computes $y_s = g^{x_s} \pmod p$
3. **Shared Secret**: $K = y_c^{x_s} = g^{(x_c \times x_s)} \pmod p$
4. **Exchange Hash**: $H = hash(id_C \| id_S \| init_C \| init_S \| PK_S \| y_c \| y_s \| K)$
5. **Server Authentication**: Server signs H and sends (y_s, PK_S, signature)

Key Derivation

Six keys derived from shared secret K and exchange hash H:

- Initial IV client to server: hash(K||H||"A"||session_id)
- Initial IV server to client: hash(K||H||"B"||session_id)
- Encryption key client to server: hash(K||H||"C"||session_id)
- Encryption key server to client: hash(K||H||"D"||session_id)
- MAC key client to server: hash(K||H||"E"||session_id)
- MAC key server to client: hash(K||H||"F"||session_id)

SSH User Authentication Protocol

- **Public Key**: Digital signature authentication

- **Password**: Simple password authentication
- **Host-based**: Authentication based on host identity

SSH Connection Protocol

Four channel types:

1. **Session**: Remote program execution
2. **X11**: X Window System forwarding
3. **forwarded-tcpip**: Remote port forwarding
4. **direct-tcpip**: Local port forwarding

SSH Port Forwarding

- **Local Port Forwarding**: SSH client listens on local port, forwards to remote application server
- **Remote Port Forwarding**: SSH server listens on remote port, forwards to local application server
- **Process**:
  1. Client establishes SSH connection to SSH server
  2. Client configures local port forwarding
  3. SSH server creates connection to destination server
  4. All traffic encrypted through SSH tunnel

SSH Applications

1. **Anonymous FTP**: Software updates with origin/integrity verification
2. **Secure FTP**: Protected file transfers (e.g., web page uploads)
3. **Secure Remote Administration**: Protected system administration
4. **Virtual Private Network**: Securing other applications via port forwarding

SSH Algorithms

- **Key Exchange**: Ephemeral Diffie-Hellman
- **Server Authentication**: RSA or DSS signatures
- **MAC**: HMAC-SHA1 or HMAC-SHA256
- **Encryption**: 3DES, AES, RC4, others

Key Exam Points

Critical Concepts to Remember

1. **Email Security Fundamentals**: Four main vulnerabilities and how PGP/S/MIME address them
2. **PGP vs S/MIME**: Key differences in trust models (web of trust vs hierarchical)
3. **Kerberos Operation**: Three-phase protocol and purpose of each phase
4. **SSH Architecture**: Three-layer design and security goals
5. **Authentication Methods**: Comparison of NTLM, Kerberos, and SSH approaches

Common Exam Questions

1. Trace through Kerberos authentication protocol steps
2. Explain PGP key management and web of trust
3. Compare and contrast PGP and S/MIME approaches
4. Describe SSH key exchange and key derivation process
5. Analyze security vulnerabilities in different authentication systems

Security Analysis Framework

For each protocol, consider:

- **Confidentiality**: How is data protected?
- **Integrity**: How is tampering detected?
- **Authentication**: How are identities verified?
- **Non-repudiation**: How is message origin proven?
- **Key Management**: How are keys distributed and maintained?
- **Scalability**: How does the system scale with users?
- **Vulnerabilities**: What are the main weaknesses?