Application of Cryptography: IPsec & IKE

Guevara Noubir

http://www.cs.ru.nl/home/nobi/Courses/CSC252/V04

Textbook: "Network Security",
Kaufman - Perlman - Speciner

Reading: Chapters 17-18

Outline

- Introduction to secure networking
- IPsec:
  - Authentication Header and Encapsulated Secure Payload
- Internet Key Exchange (IKE)

Approach to Secure Networking

- Key Distribution Centers: Trusted Third Parties
  - or
- Public Key Systems + Public Key Infrastructure
Key Distribution Center

- Solve the scalability problem of a set of $n$ nodes using secret key
  - $n^2/(n+1)/2$ keys
- New nodes are configured with a key to the KDC
  - e.g. $K_n$ for node $A$
- If node $A$ wants to communicate with node $B$
  - $A$ sends a request to the KDC
  - The KDC securely sends to $A$: $E_K(R_{AB})$ and $E_K(R_{BA})$
- Advantage:
  - Single location for updates, single key to be remembered
- Drawbacks:
  - If the KDC is compromised!
  - Single point of failure/performance bottleneck => multiple KDC?
- Example of systems: Kerberos

Public Keys and Certification Authorities

- How do you know the public key of a node?
- Typical solution:
  - Use a trusted node as a certification authority (CA)
  - The CA generates certificates: Signed(A, public-key, validity information)
  - Everybody needs to know the CA public key
  - Certificates can be stored in a directory service or exchanged during the authentication process
- Advantages:
  - The CA doesn’t have to be online => more physical protection
  - Not a performance bottleneck, not a single point of failure
  - Certificates are not security sensitive: only threat is DoS
  - A compromised CA cannot decrypt conversation but can lead to impersonation
  - A certification hierarchy can be used: e.g. X.509

Securing Network Stacks

- Where to put the security in a protocol stack?
- Practical considerations:
  - End to end security
  - No modification to OS/network stack
SSL vs. IPsec

- **SSL:**
  - Avoids modifying "TCP stack" and requires minimum changes to the application
  - Mostly used to authenticate servers
- **IPsec:**
  - Transparent to the application and requires modification of the network stack
  - Authenticates network nodes and establishes a secure channel between nodes
  - Application still needs to authenticate the users

---

Some Issues with Real-time Communication

- Session key establishment
- Perfect Forward Secrecy
  - Diffie-Hellman based IPS
  - Eavesdropping:
    - If keys are exchanged Diffie-Hellman protects against passive attacks
    - Signature keys are usually not encrypted
- Preventing Denial of Service
  - SYN attack on TCP: use stateless cookies = hash(UDP addr, secret)
  - Puzzles: e.g., what 27-bit number has an MD = x? (These techniques do not fully protect against DDOS launched through viruses)
- Hiding endpoint identity:
  - DH + authentication allows anonymous connection or detects man-in-the-middle
- Live partner re-assurance:
  - Modify DH to include a nonce in the computation of the session key
  - Optimization using parallel computation, session resumption, denialability

---

IPsec Protocol Suite (IETF Standard)

- Provides inter-operable cryptographically based security services:
  - Services: confidentiality, authentication, integrity, and key management
- Protocols:
  - Authentication Header (AH): RFC2402
  - Encapsulating Security Payload (ESP): 2406
  - Internet Key Exchange (IKE)
- Environments: IPv4 and IPv6
- Modes:
  - Transport (between two hosts)
  - Tunnel (between hosts/firewalls)
**IPsec**

- **Assumption:**
  - End nodes already established a shared session key (manually or IKE)

- **Security Association:**
  - Each secure connection is called a security association (SA)
  - For each SA: key, end-node, sequence number, services, algorithms
  - SA is unidirectional and identified by (destination-address, SPI = Security Parameter Index)

- **Protocols:**
  - Authentication Header: integrity protection
  - Encapsulated Security Payload: encryption and/or integrity

---

**IP Packets**

- Diagram showing IP packet structure:
  - IP header
  - TCP data

---

**AH Formatting**

- AH Protocol Number = 51

- Diagram showing AH packet format:
  - Transport mode
  - Tunnel mode

- SN: for replay detection
**ESP Formatting**

ESP Protocol Number = 50

1. **Transport mode**
   - IP header
   - ESP
   - TCP
   - Data

2. **Tunnel mode**
   - Encapsulated IP header
   - ESP
   - TCP
   - Data

**ESP Header**

- Security Parameters Index (SPI)
- Sequence Number (32)
- Padding Length (8)
- Payload Data (variable)
- Padding (0-255 bytes)
- Authentication Data (variable)
- Authenticator (variable)
- Authorization Data

**IPsec: Internet Key Exchange**

- **Goal:**
  - Mutual authentication and establishment of a shared secret session key using:
    - Pre-shared secret key or public signature-only key, or public encryption key
    - Negotiation of features and cryptographic algorithms

- **Specification documents:**
  - ISAKMP (Internet Security Association and Key Management Protocol): RFC 2406
  - IKE: RFC 2409
  - DOI (Domain Of Interpretation): RFC 2407
### Photuris

- **Photuris goal:** signed Diffie-Hellman exchange
  1. \( A \rightarrow B, C_A \)
  2. \( B \rightarrow A, C_B, C_{op} \) crypto offered
  3. \( A \rightarrow B, C_B, C_{op}, g^{\ast} \mod p \) crypto selected
  4. \( B \rightarrow A, C_B, C_{op}, g^{\ast} \mod p \)
  5. \( A \rightarrow B, C_B, C_{op}, g^{\ast} \mod p(A, \text{signature of previous message}) \)
  6. \( B \rightarrow A, C_B, C_{op}, g^{\ast} \mod p(B, \text{signature of previous message}) \)

- Role of \( C_B, C_{op} \) and messages
- Additional features: SPI selection
- Why not sign messages 3 & 4...?

### Simple Key-Management for Internet Protocol (SKIP)

- Uses long term Diffie-Hellman keys
- Parties assumed to know each other public keys (i.e., \( g^a \mod p \)) or exchange certificates
- Session key \( X = g^{\ast} \mod p \) is established in 0 messages
- Each packet is encrypted using data key \( S \) and each packet contains: \( X(S) \)
  - Same \( S \) can be used for several packets
- Later on PFS was added by periodically forgetting the keys and doing a new DH

### ISAKMP (RFC2408)

- Proposed by NSA as a framework and accepted by IETF
- Runs over UDP and allows to exchange fields to create a protocol
- IKE (RFC2409) based on OAKLEY & SKEME using ISAKMP syntax

- **IKE phases:**
  1. Mutual authentication and session key establishment (also called ISAKMP SA or IKE SA)
  2. AH/ESP SAs establishment
     - Each source/destination/port has its own SA/keys otherwise ESP traffic not using integrity could be decrypted...
     - IKE uses default port 500
Phase 1 IKE

- Two modes:
  - Aggressive mode: mutual authentication and session key establishment in three messages
    - $A \rightarrow B: g^r \mod p, A, $crypto$ proposal$
    - $B \rightarrow A: g^s \mod p, crypto$ choice, proof I'm B
    - $A \rightarrow B: proof' I'm А$
  - Main: additional features such as hiding end-points identities and negotiating crypto $DH$ algorithm
    - $A \rightarrow B: crypto$ suite 1 support
    - $B \rightarrow A: crypto$ suite 1 choose
    - $A \rightarrow B: g^r \mod p$
    - $B \rightarrow A: g^s \mod p$
    - $A \rightarrow B: g^w \mod p (A, proof I'm А)$
    - $B \rightarrow A: g^v \mod p (B, proof I'm B)$

Phase 1 IKE

- Key types:
  - Pre-shared secret key
  - Public encryption key: fields are separately encrypted using the public key
  - Optimized public encryption key: used to encrypt a random symmetric key, and then data is encrypted using the symmetric key
  - Public signature key: used only for signature purpose

- 8 variants of IKE phase 1: 2 modes x 4 key types

- Proof of Identity:
  - Required in messages 2-3 aggressive mode and 5-6 main mode
  - Proves the sender knows the key associated with the identity
  - Depends on the key type
  - Hash of identity key, $DH$ values, nonces, crypto choices, cookies
  - Alternative: MAC of previous messages

Phase 1 IKE

- Negotiating cryptographic parameters
  - Specifies rules of acceptable algorithms:
    - (3DES, IDEA, RSA: public key encryption, $DH$)
    - (96 bits, pre-shared key, strong curve, ...)
  - The standard specifies a $DH$-based implementation of algorithms
  - $DHkey =$ shared key
  - The lifetime of the $SA$ can be negotiated

- Session keys:
  - Key seed: $SKSKEY$
    - $SKSKEY = key$ plus $pre-shared, proof_A$
  - Encryption public keys: $pr(SH, (A, (g^s, cookies)), (A, (g^r, cookies)))$
  - Secret to generate other keys: $SKSKEY$, $SKSKEY_0 = pr(SKKEY, (g^w, cookies, 0))$
  - Integrity key: $SKSKEY_a = pr(SKKEY, (SKSKEY_0, (g^w, cookies, 1)))$
  - Encryption key: $SKSKEY_e = pr(SKKEY, (SKSKEY_0, (g^w, cookies, 2)))$

- Message 10:
  - Random 32-bit serves the purpose of a SRT but is an inefficient manner because they have to be renumbered
IKE Phase 1:
Public Signature Keys, Main Mode

- Description:
  - Both parties have public keys for signatures
  - Hidden endpoint identity (except for $A$)

- Protocol:
  - $A \rightarrow B$: CP
  - $B \rightarrow A$: CPA
  - $A \rightarrow B$: $g^a \mod p$, nonce$_A$ $A$
  - $B \rightarrow A$: $g^b \mod p$, nonce$_B$ $B$
  - $K = f(g^a \mod p, nonce$_A$, nonce$_B$)
  - $A \rightarrow B$: $K$, proof I'm $A$, [certificate]
  - $B \rightarrow A$: $K$, proof I'm $B$, [certificate]

- Questions:
  - What is the purpose of the nonces?
  - Can we make the protocol shorter (2 messages)? At what expense?

IKE Phase 1:
Public Signature Keys, Aggressive Mode

- Protocol:
  - $A \rightarrow B$: CP, $g^a \mod p$, nonce$_A$, $A$
  - $B \rightarrow A$: CPA, $g^b \mod p$, nonce$_B$, $B$, proof I'm $B$, [certificate]
  - $A \rightarrow B$: proof I'm $A$, [certificate]

IKE Phase 1:
Public Encryption Keys, Main Mode, Original

- Protocol:
  - $A \rightarrow B$: CP
  - $B \rightarrow A$: CPA
  - $A \rightarrow B$: $g^a \mod p$, {nonce$_A$}$_B$ {A}$_B$
  - $B \rightarrow A$: $g^b \mod p$, {nonce$_B$}$_A$ {B}$_A$
  - $K = f(g^a \mod p, nonce$_A$, nonce$_B$)
  - $A \rightarrow B$: $K$, proof I'm $A$
  - $B \rightarrow A$: $K$, proof I'm $B$
IKE Phase 1:
Public Encryption Keys, Aggressive Mode, Original

- Protocol:
  - $A \rightarrow B$: $CP, g^r \mod p, \{\text{nonce}\}_{A}^{d} \{A\}_B$
  - $B \rightarrow A$: $CPA, g^s \mod p, \{\text{nonce}\}_{A}^{d} \{B\}_A$ proof I'm B
  - $A \rightarrow B$: proof I'm A

IKE Phase 1:
Public Encryption Keys, Main Mode, Revised

- Protocol:
  - $A \rightarrow B$: $CP$
  - $B \rightarrow A$: $CPA$
  - $K_s = \text{hash}(\text{nonce}_A, \text{cookie}_A)$
  - $A \rightarrow B$: $\{\text{nonce}\}_A, K_s(g^r \mod p), K_s(A), [K_s(A's
    \text{ cert})]$\n  - $K_s = \text{hash}(\text{nonce}_A, \text{cookie}_A)$
  - $B \rightarrow A$: $\{\text{nonce}\}_B, K_s(g^s \mod p), K_s(B)$
  - $K = K(g^e \mod p, \text{nonce}_A, \text{nonce}_B, \text{cookie}_A, \text{cookie}_B)$
  - $A \rightarrow B$: $K(\text{proof I'm A})$
  - $B \rightarrow A$: $K(\text{proof I'm B})$
IKE Phase 1:
Shared Secret Keys, Main Mode

- Assumption A and B share a secret J
- Protocol:
  - A -> B: C
  - B -> A: CPA
  - A -> B: g^r mod p, nonce_a
  - B -> A: g^r mod p, nonce_b
  - K = H(J, g^r mod p, nonce_a nonce_b cookie_a cookie_b)
  - A -> B: K(proof I'm A)
  - B -> A: K(proof I'm B)

IKE Phase 1:
Shared Secret Keys, Aggressive Mode

- Protocol:
  - A -> B: C, g^r mod p, nonce_a A
  - B -> A: CPA, g^r mod p, nonce_b B, proof I'm B
  - A -> B: proof I'm A

IKE: Phase 2

- Also known as "Quick Mode": 3- messages protocol
  - A -> B: X, Y, CP, traffic, SPI_a nonce_a (g^r mod p)_optional
  - B -> A: X, Y, CPA, traffic, SPI_b nonce_b (g^r mod p)_optional
  - A -> B: X, Y, ACK
- All messages are encrypted using SKENID_e and integrity protected using SKENID_e (except X, Y)
- Parameters:
  - X: pair of cookies generated during phase 1
  - Y: 32-bit number unique to this phase 2 session chosen by the initiator
  - CP: Crypto Proposal, CPA: Crypto Proposal Accepted
  - DH is optional and could be used to provide PFS
  - Nonces and cookies get shuffled into SKENID to produce the SA encryption and integrity keys