COMP347 Computer Networks

Transport Layer Security
2006

Secure Sockets Layer (SSL)

- A protocol widely used on the Web
  - Operates between the application and transport layers
- Operations of SSL
  - Negotiation for PKI
    - Server and browser negotiate to select cryptographic algorithm and create a session secret key.
  - Communications
    - Encrypted by using the key that was negotiated.

SSL & TLS

- Secure Sockets Layer (SSL)
  - V2 1994 netscape
  - V3 1996 netscape
- Transport Layer Security (TLS)
  - V1.0 1999 RFC2246 IETF minor update from SSL v3.0
  - V1.1 April 2006 RFC4346 updates to prevent specific security attacks

IP Security Protocol (IPSec)

- Another widely used encryption protocol
  - Use with any application layer (not just web)
- Operations of IPSec between A and B
  - A and B negotiate algorithm and key
  - A and B then begin transmitting data using either:
    - Transport mode: only the IP payload is encrypted
    - Tunnel mode: entire IP packet is encrypted (needs a new header for routing in Internet) – used for VPN
**Security goals**

- Secrecy
- Authentication
- Non-repudiation
- Integrity

**The Cryptographic Model**

Plaintext \( P \) → Ciphertext \( C = E_k(P) \) → Plaintext \( P = D_k(E_k(P)) \)

**Approaches**

- **Secret key**
  - Alice and Bob share a secret \( k \)
  - Public algorithms \( E \) (encrypt), \( D \) (decrypt)
  - \( P \rightarrow E_k(P) \rightarrow D_k(E_k(P)) \)

- **Public key**
  - Bob creates a pair of keys \( E_b, D_b \)
    - Different but mathematically related
  - Public algorithms \( E, D \) require key pair
  - \( P \rightarrow E_{E_b}(P) \rightarrow D_{D_b}(E_{E_b}(P)) \)
Public Key Cryptography

\[ C = E_B(P) \]
\[ P = D_B(E_B(P)) \]

Digital signature

\[ P = E_B(D_B(P)) \]
\[ C = D_B(P) \]

Analogy

Pay Alice $5

Analogy
Key certificate

- Issued by a trusted authority
  - $D_{\text{Bob}}(\text{"Bob's public key is Eb"})$
- All users know public key of authority
  - Public keys of certifying authorities built into Internet Explorer
- Key certificate can be distributed by owner of key

Analogy

SSL/TLS

- Encrypts TCP connection
- Use public key from certificate to establish session secret key
- Uses challenge-response for authentication
- Supports mutual authentication if required

TLS sublayers

<table>
<thead>
<tr>
<th>Application protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS protocol</td>
</tr>
<tr>
<td>handshake messages</td>
</tr>
<tr>
<td>change cipher spec message</td>
</tr>
<tr>
<td>alert messages</td>
</tr>
<tr>
<td>data messages</td>
</tr>
<tr>
<td>TLS Record Protocol</td>
</tr>
<tr>
<td>TCP</td>
</tr>
</tbody>
</table>
**TLS Record protocol**

- Multiple messages of the same type may be sent in one record.
- A message may be broken across multiple records.
- Record (message) types may be interleaved
  - Each record type is a separate subchannel

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol version</th>
<th>Fragment length</th>
<th>Fragment</th>
</tr>
</thead>
</table>

**TLS Record Protection**

- Record contents are compressed and encrypted according to current settings
- Compress fragment (null only supported)
- Encrypt fragment:
  - Append MAC (message authentication code)
    - Includes record sequence number
  - For block ciphers, add needed padding
    - May use extra padding, to frustrate length analysis
  - Encrypt entire fragment

**TLS Record Protocol Design**

- Dialog: Asynchronous
- Representation: Bytes, 16-bit length MSB first, compression, encryption of data payload
- Security: Nil
- Transport-layer: TCP
- Framing: Length
- Error reporting: TLS alert protocol (see below)
- Syntax: Typed records with length
- State: Nil

**TLS Protocol**

- Current settings
  - Compression method
  - Cryptographic methods
- Pending settings
  - Handshake messages
  - Change cipher spec message
**TLS Protocol Messages**

- Handshake

<table>
<thead>
<tr>
<th>HS-Type</th>
<th>Body length</th>
<th>Body</th>
</tr>
</thead>
</table>
- Change cipher spec

<table>
<thead>
<tr>
<th>Level</th>
<th>Descr</th>
</tr>
</thead>
</table>
- Alert – level: fatal or warning; description
- Application data is carried directly

**TLS Handshake**

- Establishes session settings:
  - Session identifier – chosen by server
  - Peer certificate X509v3 certificate of peer (may be null)
  - Compression method
  - Cipher spec: cipher and MAC algorithms.
  - Master secret: 48 bytes shared between client and server
  - IsResumable: flag whether session can be resumed on a new connection.

---

**TLS handshake sequence**

- Optional: Hello request
- Client hello
  - Server hello
  - Optional: server certificate, server key exchange, certificate request
  - Server hello done
- Optional: client certificate
- Client key exchange
- Optional: certificate verify
- Change cipher spec
- Finished

**TLS handshake**

- Client hello
  - Highest protocol version client supports
  - Random data (32B) including timestamp
  - Length (1B); Session ID (0-32B; id to resume; 0 = new)
  - Length (2B); List of cipher suites (2B each) in preference order
  - Length (1B); List of compression methods (1B each)
  - Additional data for future expansion
**TLS handshake**

- **Server hello**
  - Protocol version selected (lower of client version & server version supported)
  - Random data including timestamp; independent of and different from client (32B)
  - Session ID (0 = not cached, not resumable)
  - Ciphersuite selected (2B)
  - Compression method selected (1B)

**TLS handshake**

- **Server certificate** – X.509 certificate chain
  - Commencing with server’s own certificate
  - Chain of certifications to root CA

**TLS handshake**

- **Certificate request**
  - Length (1B); List of acceptable certificate types (1B each)
  - Length (2B); List of acceptable CA’s (opaque byte strings, each with a 2B length)
- **Server hello done**
  - Empty message body

**TLS handshake**

- **Client key exchange**
  - For RSA key exchange, encrypted premaster secret containing:
    - Protocol version – the highest protocol version supported by the client; to prevent rollback attacks.
    - 46 bytes of securely random data
    - Above 48 bytes encrypted with server’s RSA key previously exchanged.
  - Master secret is computed from premaster secret and random values from client and server hello messages
**TLS handshake**

- Certificate verify
  - A digital signature of the handshake messages sent and received up to but excluding this message. (Excludes hello request) (20B for DSA signature; 36B for RSA signature)
  - Used to verify the digital signature algorithm

**TLS alerts**

- Fatal error: transmit alert and close session
- Warning: other party may treat as fatal
- Description: 1-byte code
  - Close_notify
  - Unexpected_message
  - Bad_record_mac
  - Decryption_failed
  - Bad_certificate …

**TLS handshake Design**

- Dialog: Fixed sequence with optional messages
- Representation: Bytes, integers (16, 24), opaque objects
- Security: TLS record encryption
- Transport-layer: TCP
- Framing: Length (handshake messages)
- Error reporting: Alert messages
- Syntax: Typed, structured messages
- State: Connection settings, pending settings; cached sessions (resumable)
Cryptographic techniques

- RSA – public-key method based on exponentiation.
- DES – Data encryption standard (56-bit key)
- 3DES – triple DES EDE – encrypt(k1), decrypt(k2), encrypt(k3)
- IDEA – a new secret key technique
- MD5, SHA-1 message digest/secure hash algorithms
- HMAC – keyed hash for message authentication
- CBC – cipher block chaining – XOR plaintext with prior ciphertext block before encrypting