SSL
Real Life²: Secure Communication

- **Secure Socket Layer (SSL)**
  - Transport layer protocol
- **IP Security (IPSec)**
  - Network layer protocol
SSL

• **Transport layer security**
  – Provides confidentiality, integrity, authentication of endpoints
  – Developed by Netscape for WWW browsers and servers

• **Internet protocol version: TLS**
  – Almost identical to SSL
  – RFC 4346 (ver. 1.1)
SSL Session

• Association between two peers
  – May have many associated connections
  – Information for each association:
    • Unique session identifier
    • Peer’s X.509v3 certificate, if needed
    • Compression method
    • Cipher spec for cipher and MAC
    • “Master secret” shared with peer (384 bits)
SSL Connection

- Describes how data exchanged with peer
- Information for each connection
  - Random data
  - Write keys (used to encipher data)
  - Write MAC key (used to compute MAC)
  - Initialization vectors for ciphers, if needed
  - Sequence numbers
Structure of SSL

SSL Alert Protocol
SSL Handshake Protocol
SSL Record Protocol
SSL Change Cipher Spec Protocol
SSL Application Data Protocol
Supporting Crypto

• Initial phase: PK system exchanges keys
  – Messages enciphered using classical ciphers, check-summed using cryptographic checksums
  – Only certain combinations allowed
    • Depends on algorithm for interchange cipher
  – Interchange algorithms: e.g., RSA, DH, Fortezza (D.O.D)
## RSA: Cipher, MAC Algorithms

<table>
<thead>
<tr>
<th>Interchange cipher</th>
<th>Classical cipher</th>
<th>MAC Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA, key ≤ 512 bits</td>
<td>none</td>
<td>MD5, SHA</td>
</tr>
<tr>
<td></td>
<td>RC4, 40-bit key</td>
<td>MD5</td>
</tr>
<tr>
<td></td>
<td>RC2, 40-bit key, CBC mode</td>
<td>MD5</td>
</tr>
<tr>
<td></td>
<td>DES, 40-bit key, CBC mode</td>
<td>SHA</td>
</tr>
<tr>
<td>RSA</td>
<td>None</td>
<td>MD5, SHA</td>
</tr>
<tr>
<td></td>
<td>RC4, 128-bit key</td>
<td>MD5, SHA</td>
</tr>
<tr>
<td></td>
<td>IDEA, CBC mode</td>
<td>SHA</td>
</tr>
<tr>
<td></td>
<td>DES, CBC mode</td>
<td>SHA</td>
</tr>
<tr>
<td></td>
<td>DES, EDE mode, CBC mode</td>
<td>SHA</td>
</tr>
</tbody>
</table>
## Fortezza: Cipher, MAC Algorithms

<table>
<thead>
<tr>
<th>Interchange cipher</th>
<th>Classical cipher</th>
<th>MAC Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortezza key exchange</td>
<td>none</td>
<td>SHA</td>
</tr>
<tr>
<td></td>
<td>RC4, 128-bit key</td>
<td>MD5</td>
</tr>
<tr>
<td></td>
<td>Fortezza, CBC mode</td>
<td>SHA</td>
</tr>
</tbody>
</table>
Digital Signatures

• RSA
  – Concatenate MD5 and SHA hashes
• Fortezza
  – Compute SHA hash
SSL Record Layer

Message

Compressed blocks

Compressed blocks, enciphered, with MAC

MAC
Record Protocol Overview

• Lowest layer, taking messages from higher
  – Max block size 16,384 bytes
  – Bigger messages split into multiple blocks

• Construction
  – Block $b$ compressed; call it $b_c$
  – MAC computed for $b_c$
    • If MAC key not selected, no MAC computed
  – $b_c$, MAC enciphered
    • If enciphering key not selected, no enciphering done
  – SSL record header pre-pended
SSL MAC Computation

• Symbols
  – $h$ hash function (MD5 or SHA)
  – $k_w$ write MAC key of entity
  – $ipad = 0x36$, $opad = 0x5C$
    • Repeated to block length (from HMAC)
  – $seq$ sequence number
  – $SSL_{\text{comp}}$ message type
  – $SSL_{\text{len}}$ block length

• MAC: $h(k_w || opad || h(k_w || ipad || seq || SSL_{\text{comp}} || SSL_{\text{len}} || block))$
SSL Handshake Protocol

- Used to initiate connection
  - Sets up parameters for record protocol
  - 4 rounds
- Upper layer protocol
  - Invokes Record Protocol
- Note: what follows assumes client, server use RSA as interchange cryptosystem
Overview of Rounds

1. Create SSL client-server connection
2. Server authenticates itself
3. Client validates server, begins key exchange
4. Acknowledgments all around
Handshake Round 1: connection

1. \{ v_C \parallel r_1 \parallel s_1 \parallel ciphers \parallel comps \}  

Client \rightarrow \text{Server}

2. \{ v \parallel r_2 \parallel s_2 \parallel cipher \parallel comp \}  

Client \leftarrow \text{Server}

\( v_C \): Client’s version of SSL  
\( v \): Highest version of SSL that Client, Server both understand  
\( r_1, r_2 \): nonces (timestamp and 28 random bytes)  
\( s_1 \): Current session id (0 if new session)  
\( s_2 \): Current session id (if \( s_1 = 0 \), new session id)  
\( ciphers \): Ciphers that client understands  
\( comps \): Compression algorithms that client understand  
\( cipher \): Cipher to be used  
\( comp \): Compression algorithm to be used

Note: we assume client and server use RSA as interchange cryptosystem
Handshake Round 2: server authentication

Client ← {server_certificate} Server
Client ← {mod || exp || \{ h(r_1 || r_2 || mod || exp) \} k_S} Server
Client ← {ctype || gca} Server
Client ← {er2} Server

Note: if Server not to authenticate itself, only last message sent; third step omitted if Server does not need Client certificate (**mutual auth not default !!!**)

- \( k_S \): Server’s private key
- \( ctype \): Certificate type requested (by cryptosystem)
- \( gca \): Acceptable certification authorities
- \( er2 \): End round 2 message
- \( mod, exp \): For a new temporary key pair (not the one associated with certificate)
Handshake Round 3: key exchange

Client \(ightarrow \text{Server}\)  \(\{\text{client\_certificate}\}\)

Client \(\rightarrow \text{Server}\)  \(\{\text{pre}\}\)

Both Client, Server compute master secret \(\text{master}\):
\[
\text{master} = \text{MD5}(\text{pre} \ || \ \text{SHA}(\text{‘A’} \ || \ \text{pre} \ || \ r_1 \ || \ r_2) \ || \\
\text{MD5}(\text{pre} \ || \ \text{SHA}(\text{‘BB’} \ || \ \text{pre} \ || \ r_1 \ || \ r_2) \ || \\
\text{MD5}(\text{pre} \ || \ \text{SHA}(\text{‘CCC’} \ || \ \text{pre} \ || \ r_1 \ || \ r_2)
\]

Client \(\rightarrow \text{Server}\)  \(\{h(\text{master} || \text{opad} || h(\text{msgs} || \text{master} \ || \text{ipad}))\}\)

- \(\text{msgs}\) Concatenation of previous messages sent/received in this handshake
- \(\text{opad, ipad}\) As above
- \(\{\text{pre}\}\) Encrypted with mod/exp from previous slide
Handshake Round 4: acknowledgements

Client sends “change cipher spec” message using that protocol

\[
\{ h(master \ || \ opad \ || \ h(msgs \ || \ 0x434C4E54 \ || \ master \ || \ ipad )) \} \ k_{cipher}
\]

Server sends “change cipher spec” message using that protocol

\[
\{ h(master \ || \ opad \ || \ h(msgs \ || \ 0x53525652 \ || \ master \ || \ ipad )) \} \ \ k_{cipher}
\]

msgs  Concatenation of messages sent/received this handshake in previous rounds (does not include these messages)

opad, ipad, master  As above
SSL Change Cipher Spec Protocol

• Send single byte
• In handshake, new parameters considered “pending” until this byte received
  – Old parameters in use, so cannot just switch to new ones
SSL Alert Protocol

• Closure alert
  – Sender will send no more messages
  – Pending data delivered; new messages ignored

• Error alerts
  – Warning: connection remains open
  – Fatal error: connection torn down as soon as sent or received
SSL Alert Protocol Errors

- Always fatal errors:
  - unexpected_message, bad_record_mac, decompression_failure, handshake_failure, illegalParameter

- May be warnings or fatal errors:
  - no_certificate, bad_certificate, unsupported_certificate, certificate_revoked, certificate_expired, certificate_unknown
SSL Application Data Protocol

- Passes data from application to SSL Record Protocol layer
Real World

- **Toolkits**
  - http://www.openssl.org

- **Certificate Authorities (300+)**
  - http://www.verisign.com
  - http://www.thawte.com
  - http://www.instantssl.com
  - http://www.entrust.com
Vulnerabilities

- Virtual server issues
- Rogue CAs
- Useless warning messages
- etc.