SSL/TLS as Real-Time Protocols

- A real-time protocol is one where parties negotiate interactively to authentication each other and establish a session key
- Examples: IPsec, SSL/TLS, SSH
  - Public key based
- SSL: Secure Socket Layer
- TLS: Transport Layer Security
SSL/TLS is on top of TCP

- SSL/TLS is in a user-level process
  - No requirement on OS changes
- Relies on TCP to ensure reliable delivery
  - Timing out issues or lost data will be retransmitted
- But relying on TCP introduces the rogue packet problem . . .
The Rogue Packet Problem

• A rogue packet with malicious data can be inserted into TCP stream
• TCP won’t notice and forwards that to SSL
  – And will expect next packet in sequence
• SSL discard it
• Now the genuine packet comes
• TCP now discards the packet because the packet appears to be a duplicate :(

How about SSL/TLS Atop UDP?

• Well, it can solve the rogue packet problem
  – UDP does not care about the sequence numbers
• But SSL/TLS then needs to handle reliability issues
A Compromised Decision

• SSL/TLS is on top of TCP, not UDP
  – No worry about reliability issues
• But has to live with the rogue packet problem

Quick History

• SSLv1: never deployed
• SSLv2: deployed in Netscape Navigator 1.1 in 1995
• Microsoft introduced PCT (Private Communication Tech) by improving SSLv2
• Netscape overhauled the protocol as SSLv3
• IETF introduced TLS to unify all of them
  – Seems just another incompatible protocol
SSL/TLS Processing Unit

- TCP stream is partitioned into records
- Each record has a header and crypto protection
- Four types of records:
  - User data
  - Handshake messages (we focus on this one)
  - Alerts
  - Change cipher spec
    - should be regarded as handshake

SSLv3/TLS Basic Protocol

Choose secret $S$, compute $K=f(S, R_{Alice}, R_{Bob})$

I want to talk, ciphers I support, $R_{Alice}$

$\text{certificate}$, cipher I choose, $R_{Bob}$

$\{S\}_{Bob}$: {keyed hash of handshake msgs}

{keyed hash of handshake msgs}

Data protected w/ keys derived from $K$

{e.g. Alice, password, credit card info}

compute $K=f(S, R_{Alice}, R_{Bob})$
Several Important Terms

- $R_{Alice}$: a random number from Alice
- $S$: pre-master secret
- $K$: master secret
- $\{\}$$_{Bob}$ stands for message encrypted with Bob’s public key
- $\{}$ stands for **protected** message using encryption and/or integrity protection through secret key algorithm

If a Keyed Hash Result in *Plaintext*

Choose secret $S$, compute $K=f(S, R_{Alice}, R_{bob})$

I want to talk, ciphers I support, $R_{Alice}$

- certificate, cipher I choose, $R_{Bob}$

$\{S\}_{Bob}$, keyed hash of handshake msgs

keyed hash of handshake msgs

Data protected w/ keys derived from $K$

{e.g. Alice, password, credit card info}

compute $K=f(S, R_{Alice}, R_{bob})$
How Bob Verifies the Key Hash

- Decrypt \( \{S\}_{Bob} \) using his private key
- Compute \( K=f(S, R_{Alice}, R_{Bob}) \)
- Calculate \( hash(K, (m1, m2, "CLNT")) \)
  - HMAC algorithm
- Compares the result with the received one
- Verified if equal

Q: must the keyed hash be protected?

How Alice Verifies the Key Hash

- Calculate \( hash(K, (m1, m2, "SRVR")) \)
  - HMAC algorithm
  - Recall Alice knows \( K \) already
  - The constant string make the hash different from what Bob receives
- Compares the result with the received one
- Verified if equal

Q: must the keyed hash be protected?
Questions

- Can Eve eavesdrop?
- Can Mallory manipulate the data stream?

When Eve is Eavesdropping

\[ m_1: \text{I want to talk, my ciphers, } R_{\text{Alice}} \]
\[ m_2: \text{certificate, cipher I choose, } R_{\text{Bob}} \]
\[ \{S\}_{\text{Bob}}, h(K, (m_1, m_2, \text{"CLNT"})} \]
\[ h(K, (m_1, m_2, \text{"SRVR"}) \]

Data protected w/ keys derived from K (e.g. Alice, password, credit card info)
When Mallury is Manipulating

Questions

- When hashing, why add “CLNT” or “SRVR”?
- What if not?
If Verified, What does Bob Prove?

- The following can be regarded as the same entity:
  - The one sending, or forwarding, message 1
  - The one computing the pre-master secret that Bob received
  - The one sending message 3
- But not necessarily Alice, even claimed so!
  - Could be Mallury!
  - But Alice won’t be deceived

If Verified, What does Alice Prove?

- The following are the same entity:
  - The one sending message 2
  - The one computing $S$ and $K$ on the other end, and
  - The one sending message 4
- And this entity is Bob!
  - Based on the certificate
- Also, this entity knows $S$ and $K$
  - $S$ and $K$ are decided by Alice
- All handshake messages so far have NOT been tampered
  - Otherwise?
More Issues on SSL/TLS

- Six secrets to protect Alice-Bob communication
- Handling a long *session* with many *connections*
- What if Alice also has a certificate

---

Six Secrets

- In fact, it’s not a single key $K$ for a session
- Definition: write keys and read keys
  - Write keys: keys for transmission
  - Read keys: keys for reception
- Each direction needs three write keys
  - Integrity protection key
  - Encryption key
  - IV, if required by encryption algorithms
- And also three read keys
- Computed using $g_1(K, R_{Alice}, R_{Bob})$
Alice

Bob

Choose secret $S$, compute $K=f(S, R_{Alice}, R_{bob})$

Compute $g_i$

I want to talk, ciphers I support, $R_{Alice}$

Certificate, cipher I choose, $R_{Bob}$

Compute $K=f(S, R_{Alice}, R_{bob})$

Compute $g_i$

{(keyed hash of handshake msgs)}

{keyed hash of handshake msgs}

{keyed hash of handshake msgs}

Data protected w/ keys derived from $K$

{e.g. Alice, password, credit card info}

One Session w/ Multiple Connections

• From a long SSL session, after one connection is set up, many other connections can further be derived
  – Alice (a browser) and Bob (a web site) can have many connections, for instance

• Simplify the SSL for later connections between Alice and Bob
  – They have gone through the pain anyway . . .
**Session Initiation**

Choose secret $S$, compute $K = f(S, R_{Alice}, R_{Bob})$

- **Alice**
  - $session_id$, certificate, cipher, $R_{Bob}$
  - $\{S\}_{Bob}$: {keyed hash of handshake msgs}
  - {keyed hash of handshake msgs}
  - Data protected w/ keys derived from $K$
    - {e.g. Alice, password, credit card info}

- **Bob**
  - Remembers $(session_id, K)$
  - Compute $K = f(S, R_{Alice}, R_{Bob})$

**Session Resumption**

- **Alice**
  - $(session_id, K)$
  - $session_id$, ciphers I support, $R_{Alice}$
  - $session_id$, cipher, $R_{Bob}$: {keyed hash of msgs}
  - {keyed hash of msgs}
  - Data protected w/ keys derived from $K$
    - {e.g. Alice, password, credit card info}

- **Bob**
  - $(session_id, K)$
  - Compute $g_i$
SSL/TLS is Asymmetrical

• Alice authenticated Bob
• But Bob does not authenticate Alice
  – Until Alice login to Bob
    • Kind of late
  – Could be Mallory handshaking with Bob
• SSL/TLS can be enhanced for mutual authentication
  – If the client has a certificate

Quiz 3

• How would the SSL/TLS protocol work if Alice also has a certificate?

Quiz 1: 410avg=4.5 510avg=9.1
Quiz 2: 410avg=3.5  510avg=8.1
Encoding SSL/TLS Protocol

• Read Textbook Page 490 - 497.