CS 494/594
Computer and Network Security

Dr. Jinyuan (Stella) Sun
Dept. of Electrical Engineering and Computer Science
University of Tennessee
Fall 2010
Real-Time Communication Security

- Network layers
- Session key establishment
- Perfect forward secrecy (PFS)
- Escrow-foilage
- Clogging protection
- Identifier hiding
- Live partner reassurance
Network Basics - Headers
An example

Figure 16.13 Configuration for TCP/IP Example
What Layer?

- Application layer security
  - Client/Server (Kerberos)
  - E-mail (PEM, PGP)
  - Web access (SSL)
- Transport Layer
  - SSL/TLS
- IP layer
  - IPSec

<table>
<thead>
<tr>
<th>OS</th>
<th>application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>insert SSL (TLS or SSH)</td>
</tr>
<tr>
<td>TCP</td>
<td>replace IP with IPsec</td>
</tr>
<tr>
<td>lower layers</td>
<td></td>
</tr>
</tbody>
</table>
Real-time protocol

- Parties negotiate interactively
  - (Mutual) Authentication
  - Session key establishment
- Security association: the conversation protected by the session key
  - Perfect forward secrecy
  - Clogging protection
  - Escrow-foilage
  - Endpoint identity hiding
- IPsec, SSL/TLS, SSH
Session Key Establishment Issues

- Message authentication with a session key establishment is needed against connection hijacking.
- Sequence numbers needed against packet replays (different from TCP seq.no.).
- Session key reset before SN wrap around.
- For freshness guarantee, both parties should contribute to the session key:
  - Less likely to attacks when someone impersonate one party to the other.
  - Good key even if only one party has access to random key generator.
Perfect Forward Secrecy

- **PFS**
  - An eavesdropper cannot decrypt a session after the session concludes, even if the eavesdropper records the entire encrypted session and subsequently obtains the two parties’ long-term secrets.

- **How to achieve PFS**
  - Generate a temporary session key, not derivable from information stored at the node after the session concludes, and then forget it after the session.

- **Check the following**
  - Kerberos
  - Alice chooses the session key, and sends it to Bob, encrypted with Bob’s public key.
PFS protocol

1. Alice sends $["Alice", g^a \mod p]_{Alice}$ to Bob.
2. Bob responds with $["Bob", g^b \mod p]_{Bob}$.
3. Alice hashes the message $g^{ab} \mod p$ and sends the result to Bob.
4. Bob verifies the hash and sends a confirmation message $\text{hash}(1, g^{ab} \mod p)$ back to Alice.

Messages:
- Message 1: $["Alice", g^a \mod p]_{Alice}$
- Message 2: $["Bob", g^b \mod p]_{Bob}$
- Message 3: $\text{hash}(g^{ab} \mod p)$
- Message 4: $\text{hash}(1, g^{ab} \mod p)$
Escrow-Foilage Protection

- *key escrow* – communicating parties have to store their long-term keys with a third-party (authorities, etc.)
- *escrow-foilage* – key stored at the third party is used maliciously;
Escrow-Foilage Protection (Cont’d)

- Escrow-foilage protection:
  - A third party (e.g., a trustworthy organization) may know Alice and Bob’s long-term keys. However, the conversation between Alice and Bob can still be made secret against a passive eavesdropper with prior knowledge of Alice and Bob’s long-term keys.

- Anything with PFS will also have escrow-foilage against a passive attacker.

- Active attackers – with the long-term keys, they can impersonate Alice or Bob.
Denial-of-Service Protection

- DoS attacks: the imposter launches DoS attacks with forged IP addresses. The purpose is to use up Bob’s resources so he cannot serve the legitimate users.
  - TCP SYN attack
Denial of Service/Clogging Protection

- Cookies – server responds to a session request with a random number (cookie), initiator has to reply back with that cookie to continue
  - attacker have to either reveal its address or, abort the attack
  - stateless cookies: cookie is $H$(IP address, server’s secret); server doesn’t have to remember it

- Stateless Cookies
  - Bob does not need to keep state
  - The cookie is a function of the IP address and a secret known to Bob
  - It is easy to forge a source IP address but it is difficult to receive the packet sent back to the forged address.
Stateless Cookie Protocol

\[ c = \text{hash(IP address, secret)} \]

Does \( c = \text{hash(IP address, secret)} \)?
If so, continue with protocol.
Puzzle

- puzzles – to continue authentication server requires initiator to solve a puzzle: e.g. $MD5(x) = \ldots$, $x = ?$
  - solving is slow (depends on the size of $x$), verification fast
  - can be made stateless, how?
  - client’s computation power varies, not useful against coordinated distributed DoS attack
Endpoint Identifier Hiding

- some apps require identity protection against eavesdropper
- parties can use Diffie-Hellman anonymously and then use shared key to encrypt the rest of the session (including authentication)
  - passive attacker will not know the identities
  - active attacker may still learn one or both identities, because of man-in-the-middle attack
Endo

tpoint Identifier Hiding (Cont’d)

- Which identity is more valuable to protect? two opinions
  - initiator (Alice) – Bob’s identity is probably already known
  - responder (Bob) – if Bob’s id is harder to impersonate (Alice initiates the conversation)

- In the protocol below, whose id is protected against active attack?
Alice

I want to talk, $g^a \mod p$

Bob

OK, $g^b \mod p$

$g^{ab} \mod p \{"Alice", [g^a \mod p]_{Alice}\}$

$g^{ab} \mod p \{"Bob", [g^b \mod p]_{Bob}\}$
Homework 16.4

Referring to § 16.6 Endpoint Identifier Hiding, modify Protocol 16-4 to hide the initiator's identity rather than the target's identity.
Homework 16.5

- As mentioned in § 16.6 Endpoint Identifier Hiding, it is possible to design a protocol that will hide both identifiers from an active attacker, assuming that Alice (the initiator) already knows Bob's public key. Show such a protocol.
Homework 16.6

Also as mentioned in §16.6 Endpoint Identifier Hiding, it is possible to hide both identities from active attackers if Alice and Bob share a secret key and there is a small set of entities that might initiate a connection to Bob. Show such a protocol.
Endpoint Identifier Hiding (Cont’d)

- Hide the identifiers of the two communicating parties
- Hide both parties’ identifiers from a passive attacker.
- Hide one party’s identifier from an active attacker (man-in-the-middle)
- Hide both parties’ identifiers from both passive and active attackers
  - The two parties need to know who they are talking to.
  - Use some pre-established secret, such as pre-shared secret key, other party’s public key.
Live Partner Reassurance

- Bob is vulnerable to replays
- can use different D-H exponents for different sessions
  - DH exponentiation is expensive: problem for servers, low-end clients
  - solution: same DH exponents, different nonces
    - Incorporate nonces into the session key. E.g., $K = H(g^{ab} \mod p, \text{nonces})$
    - how would these nonces be exchanged?
Live Partner Reassurance (Cont’d)

- Due to computation complexity, it might be nice to reuse some public key values, such as DH values.
Live Partner Reassurance (Cont’d)

\[ K = \text{hash}(N, g^{ab} \mod p) \]

“Alice”, \([g^a \mod p]_{\text{Alice}}\)

\([g^b \mod p]_{\text{Bob}}, N\)

\[ K\{\text{“please give me$100”}\} \]
Parallel Key Computation

- Computing D-H exponents is expensive. May do it in advance.
- In the protocol below, why is Bob sending two messages in sequence rather than combining them?
Other Issues

- **Session resumption**: use previously established session keys to bypass public-key authentication
  - one solution: share a key medium term (derive the session key from it) and request knowledge on resumption

- **Deniability**: leave a proof that Alice talked to Bob:
  - ex: Bob’s name signed by Alice’s key, what does this message prove?
  - solution: don’t use signatures for authentication, use shared secret or public encryption keys
Other Issues (Cont’d)

- **Crypto negotiation**: key exchange protocols negotiate the algorithms to be used as well (ex: key size, compression, prime (p) to use for D-H)
  
  - problem: Trudy may force Alice and Bob to use weak crypto (if it is available as an option for both parties by tampering with messages and removing stronger options)
  
  - solution?
Reading Assignment

- [Kaufman] Chapter 16