CS 3251 - Computer Networks I: Authentication

Professor Patrick Traynor
4/14/11
Lecture 25
Announcements

- Homework 3 is due next class.
  - Submit via T-Square or in person.

- Project 3 has been graded.
  - Scores have been posted!

- Project 4 is due in 1 week...
  - ...‘nuff said...
Last Time

• What are the four (general) properties security tries to provide?

• The Caesar Cipher is an example of what kind of cryptographic cipher?

• What are the differences between symmetric and asymmetric (public key) cryptography?
Diffie-Hellman - Class Exercise

- Select a partner.
- Setup: Pick a prime number $p$ and a base $g$ ($<p$)
  - $p=13$, $g=4$
- Each partner chose a private value $x$ ($<p-1$)
- Generate the following value and exchange it.
  $$y = g^x \mod p$$
- Now generate the shared secret $z$:
  $$z = y^x \mod p$$
- You should have both calculated the same value for $z$. This is your key!
Chapter 8 roadmap

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message Integrity
8.4 End point Authentication
8.5 Securing e-mail
8.6 Securing TCP connections: SSL
8.7 Network layer security: IPsec
8.8 Securing wireless LANs
8.9 Operational security: firewalls and IDS
Message Integrity

- Bob receives msg from Alice, wants to ensure:
  - message originally came from Alice
  - message not changed since sent by Alice

- Cryptographic Hash:
  - takes input m, produces fixed length value, H(m)
    - e.g., as in Internet checksum... but a bit different...
  - computationally infeasible to find two different messages, x, y such that H(x) = H(y)
    - equivalently: given m = H(x), (x unknown), can not determine x.
    - note: Internet checksum fails this requirement!
Internet Checksum: Poor Crypto Hash Function

- Internet checksum has some properties of hash function:
  - produces fixed length digest (16-bit sum) of message
  - is many-to-one

- But given a message with given hash value, it is easy to find another message with same hash value:

<table>
<thead>
<tr>
<th>message</th>
<th>ASCII format</th>
<th>message</th>
<th>ASCII format</th>
</tr>
</thead>
<tbody>
<tr>
<td>I O U 1</td>
<td>49 4F 55 31</td>
<td>I O U 9</td>
<td>49 4F 55 39</td>
</tr>
<tr>
<td>0 0 . 9</td>
<td>30 30 2E 39</td>
<td>0 0 . 1</td>
<td>30 30 2E 31</td>
</tr>
<tr>
<td>9 B O B</td>
<td>39 42 4F 42</td>
<td>9 B O B</td>
<td>39 42 4F 42</td>
</tr>
</tbody>
</table>

B2 C1 D2 AC  different messages  B2 C1 D2 AC  but identical checksums!
Message Authentication Code (MAC)

\[ H(m+s) \]

Public Internet

\[ \text{compare} \]

\( m \)

\( s \)

\( \text{(shared secret)} \)

\( \text{(message)} \)

\( m \)

\( \text{append} \)

\( H() \)

\( H(m+s) \)

\( \text{shared secret} \)
MACs in Practice

• MD5 hash function widely used (RFC 1321)
  ‣ computes 128-bit MAC in 4-step process.
  ‣ arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x
    • recent (2005) attacks on MD5

• SHA-1 is also used
  ‣ US standard [NIST, FIPS PUB 180-1]
  ‣ 160-bit MAC
    • Brute-force attacks on SHA now require $2^{63}$ operations to find a collision.
Digital Signatures

- Cryptographic technique analogous to hand-written signatures.
  - sender (Bob) digitally signs document, establishing he is document owner/creator.
  - verifiable, nonforgeable: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document.
Digital Signatures

- simple digital signature for message m:
  - Bob “signs” m by encrypting with his private key KB, creating “signed” message, KB(m)

Bob’s message, m

Dear Alice
Oh, how I have missed you. I think of you all the time! ...(blah blah blah)
Bob

Bob’s private key

public key encryption algorithm

KB (m)

Bob’s message, m, signed (encrypted) with his private key
Digital Signatures (more)

• Suppose Alice receives msg m, digital signature $K_B(m)$

• Alice verifies m signed by Bob by applying Bob’s public key $K_B^+$ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.

• If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob’s private key.

• Alice thus verifies that:
  ‣ Bob signed m.
  ‣ No one else signed m.
  ‣ Bob signed m and not m’.

• non-repudiation:
  ‣ Alice can take m, and signature $K_B(m)$ to court and prove that Bob signed m.
Digital Signature = signed MAC

Bob sends digitally signed message:

The diagram shows:
- A large message, denoted as $m$.
- A hash function, $H$, which takes the message as input and produces a hash, $H(m)$.
- A digital signature that is encrypted with Bob's private key, $K_B^-$.
- The encrypted message digest, $K_B(H(m))$.

Alice verifies signature and integrity of digitally signed message:

The verification process includes:
- Bob's public key, $K_B^+$, used to decrypt the message.
- Alice receives the encrypted message digest, $K_B(H(m))$.
- The decrypted digest is compared to the hash of the message, $H(m)$.
- If the hashes are equal, the integrity is verified.

The diagram visually represents these steps, with arrows indicating the flow of data and keys.
Public Key Certification

• Public Key Problem:
  ‣ When Alice obtains Bob’s public key (from web site, e-mail, diskette), how does she know it is Bob’s public key, not Trudy’s?

• Solution:
  ‣ Trusted certification authority (CA)
Certificate Authorities

- **Certificate Authority** (CA): binds public key to particular entity, E.

- E registers its public key with CA.
  - E provides “proof of identity” to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E’s public key digitally signed by CA: CA says “This is E’s public key.”
Certificate Authority

- When Alice wants Bob’s public key:
  - gets Bob’s certificate (Bob or elsewhere).
  - apply CA’s public key to Bob’s certificate, get Bob’s public key
A Certificate Contains:

- Serial number (unique to issuer)
- Info about certificate owner, including algorithm and key value itself (not shown)
- Info about certificate issuer
- Valid dates
- Digital signature by issuer
Problems with PKI

- Why exactly do you trust a CA?
  - Anyone have any idea how many you actually trust?

- If two CAs present you with a certificate for Microsoft, which one is right?

- What prevents a CA from making up a key for you?

- What happens when keys are compromised?
Chapter 8 Roadmap

8.1 What is network security
8.2 Principles of cryptography
8.3 Message Integrity
8.4 End point Authentication
8.5 Securing e-mail
8.6 Securing TCP connections: SSL
8.7 Network layer security: IPsec
8.8 Securing wireless LANs
8.9 Operational security: firewalls and IDS
Authentication

Goal: Bob wants Alice to “prove” her identity to him

Protocol ap1.0: Alice says “I am Alice”

Failure scenario??
**Goal:** Bob wants Alice to “prove” her identity to him

**Protocol ap1.0:** Alice says “I am Alice”

in a network, Bob can not “see” Alice, so Trudy simply declares herself to be Alice
Authentication: another try

**Protocol ap2.0**: Alice says “I am Alice” in an IP packet containing her source IP address

Failure scenario??
Authentication: another try

Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address

Trudy can create a packet “spoofing” Alice’s address
Authentication: another try

**Protocol ap3.0:** Alice says “I am Alice” and sends her secret password to “prove” it.

Failure scenario??
Authentication: another try

**Protocol ap3.0**: Alice says “I am Alice” and sends her secret password to “prove” it.

**replay attack**: Trudy records Alice’s packet and later plays it back to Bob.
Authentication: yet another try

Protocol ap3.1: Alice says “I am Alice” and sends her encrypted secret password to “prove” it.

Failure scenario??
**Protocol ap3.1:** Alice says “I am Alice” and sends her encrypted secret password to “prove” it.

Record and playback still works!
Authentication: yet another try

**Goal:** avoid playback attack

**Nonce:** number (R) used only once –in-a-lifetime

**ap4.0:** to prove Alice “live”, Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key

Failures, drawbacks?

“I am Alice”

R

\( K_{A-B} (R) \)

Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!
Authentication: ap5.0

ap4.0 requires shared symmetric key

- can we authenticate using public key techniques?

**ap5.0**: use nonce, public key cryptography

"I am Alice"

Bob computes

\[ K^+_A (K^-_A (R)) = R \]

and knows only Alice could have the private key, that encrypted R such that

\[ K^+_A (K^-_A (R)) = R \]
Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)

I am Alice

Send me your public key

R

K⁻(R)

Kₐ

Send me your public key

Trudy gets

m = K⁺(K⁻(m))

sent m to Alice

encrypted with Alice's public key

m = K⁻(K⁺(m))
Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)

Difficult to detect:
• Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation)
• problem is that Trudy receives all messages as well!
Remember Diffie-Hellman?

- How does Alice know Bob sent $T_A$?

Alice picks her $x$: $S_A$

Bob picks his $x$: $S_B$

$T_A = g^{S_A} \mod p$

$T_B = g^{S_B} \mod p$

Alice computes: $K = T_B^{S_A} \mod p$

Alice computes:

$T_B^{S_A} = (g^{S_B})^{S_A} = g^{S_B \cdot S_A} = g^{S_A} \cdot S_B = (g^{S_A})^{S_B} = T_A^{S_B} \mod p$

Encrypted communication with $K$

Alice computes: $K = T_A^{S_B} \mod p$

- There is nothing to prevent a man-in-the-middle attack against this protocol.
Next Time

- Read Sections 8.5-8.6
- Homework 3 is due at the beginning of next class.
  - Show up late and it will be marked as late!