Announcements

- HW1 deadline extended to 9/20
- Project 1 Checkpoint 1 this Friday
- Repos: [4:12PM 9/12/11] 21/59 = 35.5%
What will we learn today?

- **Why:** brief history
- **How:** Cryptography and Steganography
  - Codes
  - Ciphers
    - Symmetric, Asymmetric
- **Today:** Kerberos, HTTPS
A continuous arms race

- 1000's of years of guarding secrets
- Spartans – scytale, transposition cipher
- Romans – Caesar Cipher, rotation cipher
- Allied Analysis broke the ADFGVX
  - Led to the Zimmerman Letter decryption
  - Led to US involvement in WWI
- Breaking ENIGMA during WWII
  - Led to Allied tactical advantages
A continuous arms race

Cryptographers

Devise cryptosystems

Cryptanalysts

Find weaknesses
Desired properties [Schneier96]

- **Confidentiality** – Ensure that an eavesdropper can not read a message.
- **Authentication** – It should be possible for the receiver of a message to ascertain its origin; an intruder should not be able to masquerade as someone else.
- **Integrity** – It should be possible for the receiver of a message to verify that it has not been modified in transit; an intruder should not be able to substitute a false message for a legitimate one.
- **Nonrepudiation** – A sender should not be able to falsely deny later that he sent a message.
The history of communication

- Secret Writing
  - Steganography
  - Cryptography
    - Transposition
    - Substitution
      - Code
      - Cipher
Steganography

- The act of *hiding information*
- Often in plain sight...
- Example: slightly modify pixel data...
- See app: **steghide**
  - Operates on both images and audio
  - Graph-theoretic basis
  - `man steghide`
Steganography

- The act of **hiding information**
- Often in plain sight

**When successful, any eavesdropper never knows that a certain message has been transmitted.**

- Operates on both images and audio
- Graph-theoretic basis
- man steghide
Steganography

- The act of hiding information
- Often in plain sight...
- Slightly modify pixel data...
- See app: steghide

When successful, any eavesdropper never knows that a certain message has been transmitted.
Steganography

- The act of hiding information
- Often in plainsight...
- Slightly modify pixel data...

see app: steghide

When successful, any eavesdropper never knows that a certain message has been transmitted.

Plausible Deniability

I just sent a picture of a flower...
Deny that any message was sent!
American Revolution, 1775

- *One if by land, two if by sea.*
- American troops depended on this information about British movements
- “Paul Revere's Ride,” Henry Wadsworth Longfellow
- Military message in plain sight
- Plausible deniability—risk of British arrest
- Steganography at work!
Cryptography

- The act of **disguising information**
- Transforms what is called **plain text** into **cipher text**
- Two forms: transposition, and substitution
  - **Transposition** scrambles the plaintext letters
    - book → kobo
  - **Substitution** replaces words or characters
    - book → cjjl
    - Two forms: codes, and ciphers
    - Codes replace words for other words
      - book → bird
    - Ciphers replace individual characters
      - Title slide ciphertext: Gur Cbjre bs Xabjyrqtr
The unbreakable cipher

- U.S. Patent 1,310,719
- Vernam Cipher – one-time pad (OTP)
- Mauborgne co-invented—thought of randomness
- Shannon proved it is both unbreakable and fundamental!
- Beautiful simplicity
- Incredibly powerful technology
The unbreakable cipher

- U.S. Patent 1,310,719
- **Vernam Cipher** – one-time pad (OTP)

The NSA has called this patent "perhaps one of the most important in the history of cryptography."

and fundamental!

- Beautiful simplicity
- Incredibly powerful technology
Is $\oplus$ a good stream cipher?

<table>
<thead>
<tr>
<th>Plain Text</th>
<th>Key</th>
<th>Cipher Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Vernam Cipher Encrypt

Plaintext

"Hi"

1101000 1101001

⊕⊕⊕⊕⊕⊕⊕ ⊕⊕⊕⊕⊕⊕⊕

Random OTP Key

1110100 1001101

"tM"

Cipher Text

0011100 0100100

\x1c$"
Vernam Cipher Decrypt

Cipher Text

\(\text{\textbackslash x1c$}\)

0011100 0100100

Random OTP Key

1110100 1001101 "tM"

Plain Text

1101000 1101001 "Hi"
Symmetric Key Cryptography

- Confidentiality via shared keys
  \[ E_K(M) = C \]
  \[ D_K(C) = M \]
- OTP is impractical because key length equals message length
- Alternatives
  - Stream Ciphers: RC4, A5/1,2,3 (GSM...)
  - Block Ciphers: AES, DES, Blowfish
The treasure chest analogy

Alice  →  Bob
The treasure chest analogy

Alice → Eve → Bob

Bad, can easily be intercepted and opened, by the nefarious Eve!
The treasure chest analogy

Alice  →  Bob
Our first very simple protocol.
Hash Message Authentication Code (HMAC, MAC)

• Hash message using a hash keyed with shared key
• Produce MAC
• Alice or Bob verify integrity of messages based on these hashes
Problem: Replay Attacks

- Eve can send messages again... with observed HMAC
- Fix: introducing nonces
- Random bitstrings used only once
- Provides “sessions” for HMACs
Review: Symmetric

- **Confidentiality** – Stream/Block Ciphers
- **Integrity** – HMAC
- **Authentication** – HMAC and nonce
Perfect crypto, what next?

- Yes, we have the technology
- But, we have a different problem
- How can we share the one-time pads?
- Fundamental problem in cryptography:

Key Distribution
Kerberos: Central Key DB

- **Key Distribution Center**
  - Database of clients and secret keys
  - Handles key distribution in symmetric case
- **Trusted Arbitrator Service**
  - Secure network authentication to servers etc.
- Based on Needham-Schroeder's protocol
- From MIT's Project Athena
Kerberos: Authentication Steps

1. Request for ticket-granting ticket
2. Ticket-granting ticket
3. Request for server ticket
4. Server ticket
5. Request for service
## Kerberos: Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>client</td>
</tr>
<tr>
<td>s</td>
<td>server</td>
</tr>
<tr>
<td>a</td>
<td>client address</td>
</tr>
<tr>
<td>v</td>
<td>valid times</td>
</tr>
<tr>
<td>t</td>
<td>timestamp</td>
</tr>
<tr>
<td>$K_x$</td>
<td>x's secret key</td>
</tr>
<tr>
<td>$K_{x,y}$</td>
<td>Session key for x and y</td>
</tr>
<tr>
<td>${m}K_x$</td>
<td>m encrypted with $K_x$</td>
</tr>
<tr>
<td>$T_{x,y}$</td>
<td>x's ticket to use y</td>
</tr>
<tr>
<td>$A_{x,y}$</td>
<td>Authenticator from x to y</td>
</tr>
</tbody>
</table>
Kerberos: The protocol

\( K_c \) – one-way hash of client password

\( T_{c,s} = s, \{c,a,v,K_{c,s}\}K_s \) – ticket

\( A_{c,s} = \{c,t,\text{key}\}K_{c,s} \) – authenticator, session key optional

1. Client to Kerberos: \( c, tgs \)
2. Kerberos to Client: \( \{K_{c,tgs}\}K_c, \{T_{c,tgs}\}K_{tgs} \)
3. Client to TGS: \( \{A_{c,s}\}K_{c,tgs}, \{T_{c,tgs}\}K_{tgs} \)
4. TGS to Client: \( \{K_{c,s}\}K_{c,tgs}, \{T_{c,s}\}K_s \)
5. Client to Server: \( \{A_{c,s}\}K_{c,s}, \{T_{c,s}\}K_s \)
# Diffie Hellman Key Exchange [Wikipedia]

<table>
<thead>
<tr>
<th>Alice</th>
<th>Evil Eve</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice and Bob exchange a Prime (P) and a Generator (G) in clear text, such that $P &gt; G$ and $G$ is Primitive Root of $P$&lt;br&gt;$G = 7$, $P = 11$</td>
<td>Evil Eve sees $G = 7$, $P = 11$</td>
<td>Alice and Bob exchange a Prime (P) and a Generator (G) in clear text, such that $P &gt; G$ and $G$ is Primitive Root of $P$&lt;br&gt;$G = 7$, $P = 11$</td>
</tr>
<tr>
<td>Alice generates a random number: $X_A$&lt;br&gt;$X_A = 6$ (Secret)</td>
<td></td>
<td>Bob generates a random number: $X_B$&lt;br&gt;$X_B = 9$ (Secret)</td>
</tr>
</tbody>
</table>

**Step 1**

| $Y_A = G^{X_A} \pmod{P}$<br>$Y_A = 7^6 \pmod{11}$<br>$Y_A = 4$ | | $Y_B = G^{X_B} \pmod{P}$<br>$Y_B = 7^9 \pmod{11}$<br>$Y_B = 8$ |

**Step 2**

| Alice receives $Y_B = 8$ in clear-text | Evil Eve sees $Y_A = 4$, $Y_B = 8$ | Bob receives $Y_A = 4$ in clear-text |

**Step 3**

| Secret Key $= Y_B^{X_A} \pmod{P}$<br>$\text{Secret Key} = 8^6 \pmod{11}$<br>$\text{Secret Key} = 3$ | | Secret Key $= Y_A^{X_B} \pmod{P}$<br>$\text{Secret Key} = 4^9 \pmod{11}$<br>$\text{Secret Key} = 3$ |

**Step 4**
One-Way Functions

- Given $x$, $f(x)$ is trivial to compute
- Given $f(x)$, $x$ is hard to compute
- Example: increase entropy, break a plate
- Math: what we really want are trapdoor one-way functions
Trapdoor One-Way Functions

• Given $f(x)$ and $y$, $x$ is trivial to compute
• $y$ is some secret information
• Example: take apart a $x = \text{watch}$, pieces $= f(x)$, $y = \text{assembly instructions}$
• Math: $16 \times 24 = 384$
  • $x = 16$, $f = \times$, $y = 24$
Trapdoor One-Way Functions

- Given \( f(x) \) and \( y \), \( x \) is trivial to compute
- \( y \) is some secret information

Caveat: No proof these exist, nor even evidence that they can be constructed mathematically.

- \( x = 16, \ f = *, \ y = 24 \)
Asymmetric Key Cryptography

- Confidentiality via private key
  \[ E_{\text{pub}}(M) = C \]
  \[ D_{\text{priv}}(C) = M \]
- Distribute public key, hide private key
- You made these with `ssh-keygen -t rsa`!
- Very practical, but generally slow
- Often (RSA, etc.) asymmetric methods are used to exchange symmetric keys for fast symmetric ciphers
The treasure chest analogy

Alice  ➔  Bob

B'  ➔  B
The treasure chest analogy

New protocol, no need to have the same key!
Digital Signing

- $S_{\text{priv}}(M)$ – sign by encrypting (RSA)
- $V_{\text{pub}}(M)$ – verify via decrypting (RSA)
- Can sign entire messages
- But, often signing a hash is good enough
- Hashes are often shorter—quicker to compute
Getting to Identity/Authenticity

- Send a **nonce**
- Used **only once**!
Review: Asymmetric

- Confidentiality – Public key encryption
- Integrity – Sign message with private key
- Authentication – Send a nonce challenge, use sign and verify
Digital Certificates

- Issued to **prove identity**
- Requires trusted third parties
- We call these **certificate authorities**
- Or just trusted entities in a web of trust
- Used to implement TLS, HTTPS
- x.509 – standardizations
Bob's Public Key

\[ S_{CA}(B') \]

CA Private Key

B' Signed

Bob's Certificate

Bob's Identifying Information
Alice uses the CA's public key to verify Bob's identity and obtain a trustable public key for Bob.
Public Key Infrastructure (PKI)

- **Certificate Authorities**
  - Bind public keys to certain entities ($K_B$, with Bob)
  - DigiNotar – hacked, along with other CAs
  - Admin Password: `Pr0d@dm1n`
  - Iranian-based forged Google, and more certificates

- **Web of Trust**
  - P2P model, let many others sign your public key
  - Place trust in certain signatures
  - GnuPG, PGP → implement this
Really? Yes!
HTTPS = HTTP+TLS

Netscape made SSL, IETF made TLS based on SSL

HTTP (Application)
Secure Transport/TLS
Transport Layer (TCP)
Network Layer (IP)
Link Layer (Ethernet)
Hardware Layer

HTTP is unmodified!

Port 443 is dedicated for this.
TLS—RFC 2246

• Negotiate
  1) Data integrity hash—HMACs
  2) Symmetric-key cipher for confidentiality (DES, 3DES, AES)
  3) Session key establishment (DH, RSA)
  4) Compression algorithm*

• HMACs and ciphers are keyed in both directions
• 6 keys needed total! All delivered with a shared master secret
TLS Handshaking [RFC 2246]

Figure 1. Message flow for a full handshake

* Indicates optional or situation-dependent messages that are not always sent.
What's going on?

- Negotiation Hello's == protocols, crypto methods, compression
- Server certificate (signed public key)
  - Validate with browser set of CA's
- Client sends encrypted value to server, server decrypts proving private key ownership
- Secret value used to derive symmetric session keys for encryption and MACs
Really? Yes!

Your connection to encrypted.google.com is encrypted with 128-bit encryption.

The connection uses TLS 1.0.

The connection is encrypted using RC4_128, with SHA1 for message authentication and ECDHE_RSA as the key exchange mechanism.

The connection is not compressed.
TLS Data Stream

1) Data arrives as **stream** (TCP expected!)

2) TLS segments into **chunks**

3*) Session key **encrypts** chunks, **MAC** algorithm used to create TLS **record with short header**

4) Records form **byte stream** for TCP layer
Takeaways

- **Serious challenges** in communicating securely
- **Don't** design your own
- Practical solutions **combine multiple methods**
- **Defense in depth** is needed in the real-world—cryptography alone is not enough
Resources

- Textbook CH8
- Beware of Snake Oil, Phil Zimmerman
  - Easy read, available online
- Applied Cryptography, Bruce Schneier
- RFC's
- OpenSSL (www.openssl.org)
GitHub:

git clone git://github.com/theonewolf/15-441-Recitation-Sessions.git