Client / Server Big Picture

Language for these communications?
What does HTTP request look like?

- User enters URL:
  http://www.cs.usna.edu/

- Browser sends request to www.cs.usna.edu:
  GET / HTTP/1.0
  User-Agent: Mozilla/3.0 (compatible; Opera/3.0; Windows 95/NT4)
  Accept: */*

- What would GET line be if URL were...
  http://www.cs.usna.edu/it/news.html
What does HTTP response look like?

- If okay, server sends back response:
  HTTP/1.1 200 OK
  content-length: 4303
  accept-ranges: bytes
  server: Apache/2.0.54 (Unix) PHP/5.0.4
  connection: close
  etag: "328f-10cf-1c8181c0"
  x-pad: avoid browser bug
  date: Tue, 25 Oct 2008 17:58:32 GMT
  content-type: text/html

  <?xml version = "1.0"?>
  <html> <head> ...

Variants of the HTTP request

HEAD / HTTP/1.0

GET /cgi-bin/query.pl?str=dogs&lang=en HTTP/1.0

POST /cgi-bin/query.pl HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 16

str=dogs&lang=en

GET /img1.jpg HTTP/1.1
Host: www.host1.com

GET /img6.jpg HTTP/1.1
Host: www.host1.com
Connection: close

Variants of the HTTP response

- Status codes
  - 200 OK
  - 301 Moved permanently
  - 400 Bad request
  - 403 Forbidden
  - 404 Not found
  - 500 Internal server error
  - 503 Service unavailable

Exercise: How do the HTTP request and response look like?

http://www.cs.usna.edu/adina/welcome.pl?username=ac

welcome.pl

#!/usr/bin/perl
use strict;
use CGI qw(:standard);
use CGI::Carp qw(warningsToBrowser fatalsToBrowser);

my $username = param("username");

print "Set-Cookie: Username=$username; \n";
print header();
print start_html();
print h1("Welcome $username");
print end_html();
HTTPS: HTTP over SSL

<table>
<thead>
<tr>
<th>HTTP</th>
<th>FTP</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Various</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Encryption

• Encode / decode the data

• Need:
  – algorithm
  – key

• Symmetric encryption

• Asymmetric encryption

  – Interesting property: you can switch encryption and decryption key and get the same results
SSL Protocol Highlights

- Browser connects to SSL-enabled server
- Computers agree on encryption method
- Server sends its digital certificate (contains the public key)
- Browser and server generate session key
- Further communications are encrypted using the session key

HTTP over SSL: How to use it?

- Example: need to submit login information securely; script to execute is login.pl
- Secure invocation:
Digital Certificates

• Bob got a public key from Amazon. Is it really Amazon’s key?