Lecture 16 – HTTPS (SSL)

Web Application Development

October 27, 2022

Jeffrey L. Eppinger
Professor of the Practice
School of Computer Science
Carnegie Mellon University

Lecture Schedule – 2nd Half

(subject to change)

HW#7 Due

10/25 Software Engineering

10/27 HTTPS

11/1 Passwords, 2FA, OAuth

Specs Due

Present

11/3 Sprint #1 Presentations

11/8 Transactions

Sign-ups

11/10 Internationalization

Present

11/15 Sprint #2 Presentations

11/17 Scalability

11/22 Testing

11/24 Thanksgiving (no lecture)

11/29 Demo Week (no lecture)

12/1 Demo Week (no lecture)

12/6 Review for Final Exam

12/8 Best Project Awards

The Final Exam will be on Tuesday, December 13th at 1:00pm

Final Exam has been Scheduled

- Our exam will be Tuesday, December 12th @ 1:00pm!
- The schedule for the whole university is is posted here
  - Please check for any conflicts
  - Let me know by next week if you have any conflicts
  - There will be a quiz question about this (not today)

Homework Deadlines

- HW7 was due Monday night
- HW7 last “penalty-free” late day is Friday

Homework Statistics

	hw1	hw2	hw3	hw4	hw5	hw6	hw7
100	117	115	104	111	107	92	77
90s	4	6	6	2	4	7	5
80s	0	1	3	0	0	1	1
70s	0	0	1	0	0	2	0
60s	0	0	1	0	0	2	0
50s	0	0	0	0	0	0	0
40s	0	0	0	0	2	0	0
30s	0	0	0	0	0	2	0
20s	0	0	0	0	0	1	0
10s	0	0	0	0	0	2	0
00s	0	0	1	3	2	2	6
users	121	122	116	116	115	111	89

Project Proposals

- Feedback was sent out by e-mail this afternoon to all team members
  - Most teams' projects were fine
  - Many teams received suggestions to do more or less
  - Two teams that I'm worried about
  - One team that I asked to talk to
- I'll join OH tonight (at 8pm) if teams want to discuss their feedback
- If that doesn't work (or is too crowded), we can set up a time for tomorrow, etc

Project Plan

- Details posted on Canvas:
  - Project Specification (by 11/2)
  - Sprint #1 Presentations (on 11/3)
  - Demo signups (approx. 11/9)
  - Sprint #2 Presentations (on 11/15)
  - Project Demo (starting 11/28)

Project Specification

- 2+ pages in length
  - Product backlog – list of functionality for your project
  - Sprint #1 backlog – list of functionality and who will implement it
  - Product owner – who will coordinate Sprint #1
  - Data model – code is fine, you can include it or reference a file
  - UI mockup – drawings or HTML, include it or reference file(s)
- This is not meant to be committed in stone
  - You can change and update this as you progress
  - But we want to read it before your Sprint Presentations
- You do not get a “grade” for the spec, but it factors in

Sprint Presentations

- Sprint #1 presentations will be on 11/3
  - Via Zoom, during class time
  - We will divide you into groups and you will meet in breakout rooms
  - There is no separate grade for this
  - We want to see that you are making progress
  - Teams will each present for ~10 minutes
  - Non-presenting teams will provide feedback …
    - … along with the course staff
  - See details on Canvas
- Sprint #2 presentations will be on 11/15
  - Pretty much the same as Sprint #1, but you're further along

Project Demo

- Demos will be the week of 11/28
  - Demos will be outside of class time
  - We will run demo sign ups around November 9th
  - Demos will be ~30+ minutes in length
  - You must provide the URL to allow access to your site
    - We will test your site
  - You must provide access to your GitHub repo
    - We will review the code
    - We will not run your code
  - We will provide feedback on your project during the demo

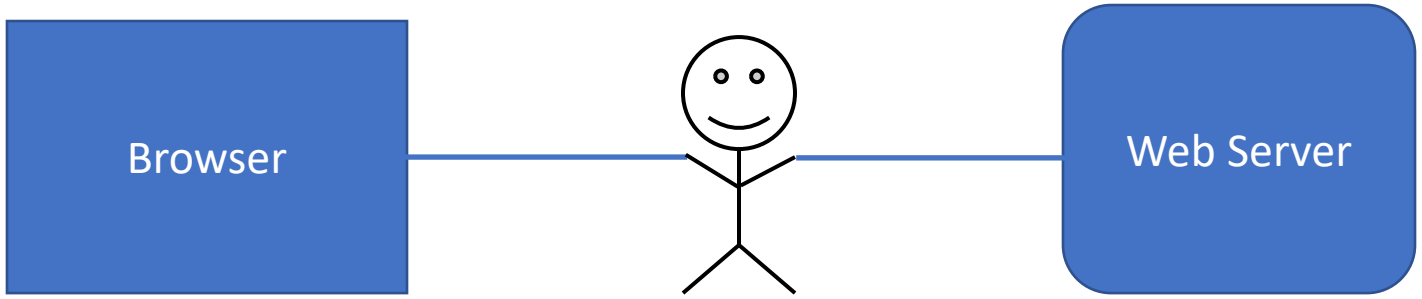
Outline for Today

- ✓ Course Administration
- → HTTPS (SSL aka TLS)

Attacking the Network

- Examples
  - Man-in-the-middle Attack
  - Sniffing
  - Spoofing
- We assume the network is not secure
- We must guard against a compromised network

Man-in-the-Middle



- Someone that can intercept network traffic
- Can read the messages (coming and going)
- Can change the messages before sending them on (to the correct or incorrect destination)

Sniffing, Eavesdropping, etc

- You can listen to the traffic going by on the net
- This is typically traffic on your subnet
  - Still it can be most interesting
  - If you can plug in to the backbone…
- E.g., use the Ethereal in promiscuous mode

Spoofing

- Pretending to be someone you're not
- IP spoofing
  - Pretending to a "client" you're not (with a specific IP address)
- E-mail Spoofing
  - The story of Satish Dharmaraj
- DNS spoofing
  - Pretending to be a server that you're not
  - Fool a DNS server to give out incorrect IP addresses for DNS Names
    - Polluting DNS server caches or modifying responses en route
  - DNSSEC protocol addresses spoofing, but most places not using it
- Note: also be careful of typos or similar characters attacks:
  - http://mytimes.com
  - http://paypa1.com

Question for you?

What are the “big three” concepts in network security?

- Authentication
- Authorization
- Privacy

Terms Defined

- Authentication
  - Knowing with whom you are communicating
    - User knowing the server and/or server knowing the user
      - Which is more important??
- Authorization
  - User having privilege to perform an operation on server
- Privacy
  - Communicating without others knowing what's said
  - Intermediaries cannot change what was said
  - Typically includes protection from replay attack

    (Typically does *not* provide secrecy of communication. Others can know communication occurred)

¿Quiz?

- Which of the “big three” protect you from:
  - Sniffing?
  - Spoofing?
  - Man-in-the-middle?

Most Sites…

…just prompt for username and password

…maintain their own DB of users and passwords

- Like in your homework

- Increasingly, sites are using third-party authentication, e.g., “use your Facebook acct”

How to Improve Security: Use SSL

- Server Authentication => SSL
  - User (client) authentication => SSL (Optional)
  - Which is more important (client or server)
- Privacy (encryption) => SSL

- Note: SSL does not provide authorization
  - This is an application specific decision

SSL

- Secure Socket Layer
  - A layer on top of TCP
  - A security model based on strong encryption techniques

Crypto Concepts You Need to Know

One-way Hashing

Secret Key Encryption

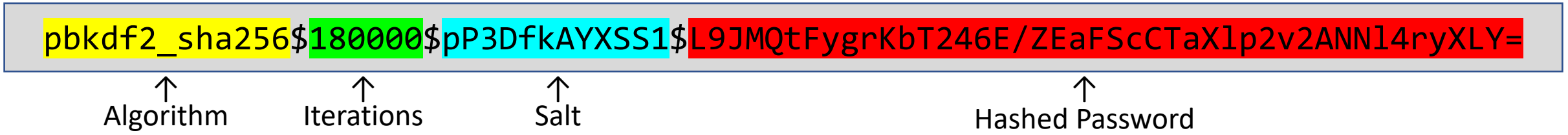
Public Key Encryption

Certificate Authority

Certificate

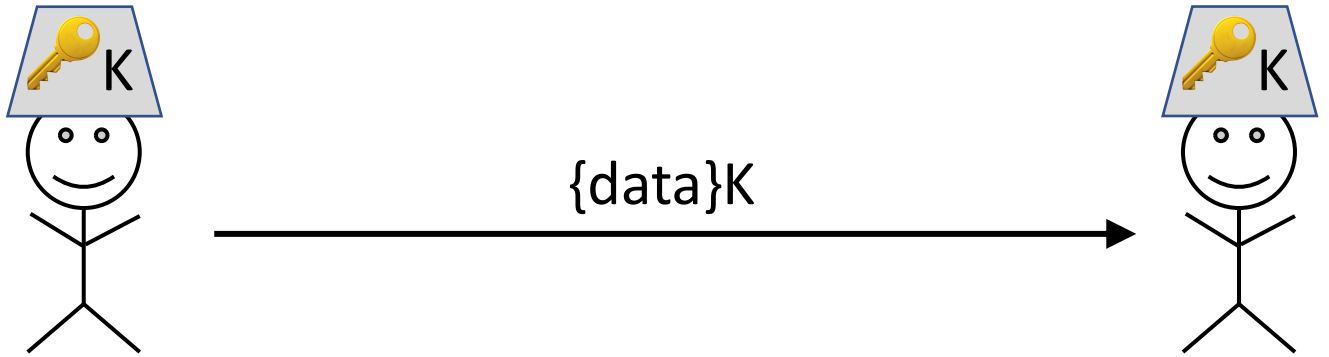
Hashing (aka Message Digests or One-Way Hashing)

- A hash function is a one-way encoding of data
  - Same input, same output
  - Different output, different input
- Easy (relatively) to compute the hash function – H(data)
- Hard to compute the hash function's inverse – H'(data)
- We only store hashed passwords on disk
  - To prevent passwords from being compromised if our servers are broken into
  - Check out the hashed passwords in Auth User Table



Secret Key Cryptography (Aka Symmetric, Private Key Crypto)

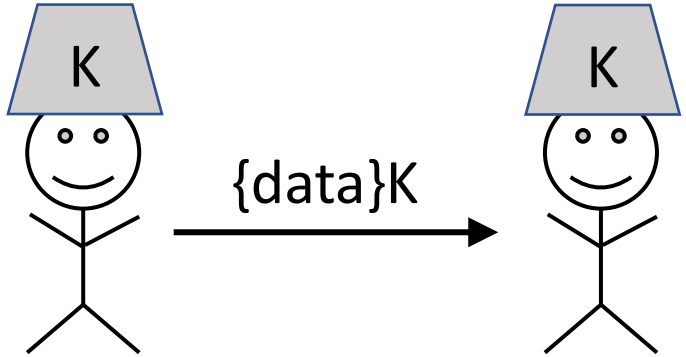
- Like in the old movies and spy books
- One key (K)
  - Shared Secret
  - Used to encrypt and decrypt
  - Notation: {data}K



Public Key Cryptography (aka Asymmetric Key Crypto)

- Key Pair (key 1 & key 2)
  - Either key can be used to encrypt (key 1 or key 2)
  - You can only decrypt using the "other key" (key 2 or key 1)
  - One key is given out (the public key)
  - The other key is kept secret (the private key)
  - Notation: For entity X, we have keys X_{pub} & X_{priv}
- A public key can be given out freely to
  - Encrypt data sent to the holder (X) of the private key
    - Notation: {data}X_{pub}
  - Verify messages signed by the holder (X) of the private key
    - Notation: {data}X_{priv}
    - Trick: send a signature with the data which is the hash of the data encrypted w/private key
      - So we are really sending data + {H(data)} X_{priv}

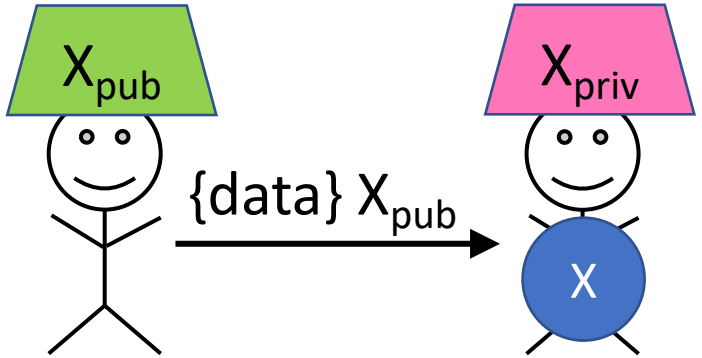
Secret Key Crypto

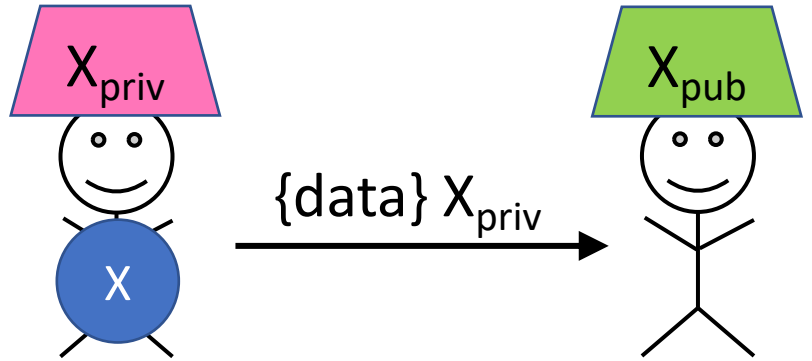


Sending

Signing

Public Key Crypto





Note: {data} X_{priv} *is usually implemented as* data + {H(data)} X_{priv}

Comparison

- Public & secret key crypto are (both) very secure
  - Unless the keys are compromised
- Public key crypto is computationally expensive
  - Secret key crypto is relatively fast
- It's hard to distribute the secret key between communicating parties
  - This is why we like public key cryptography
  - We just use public key to distribute secret keys which are then used

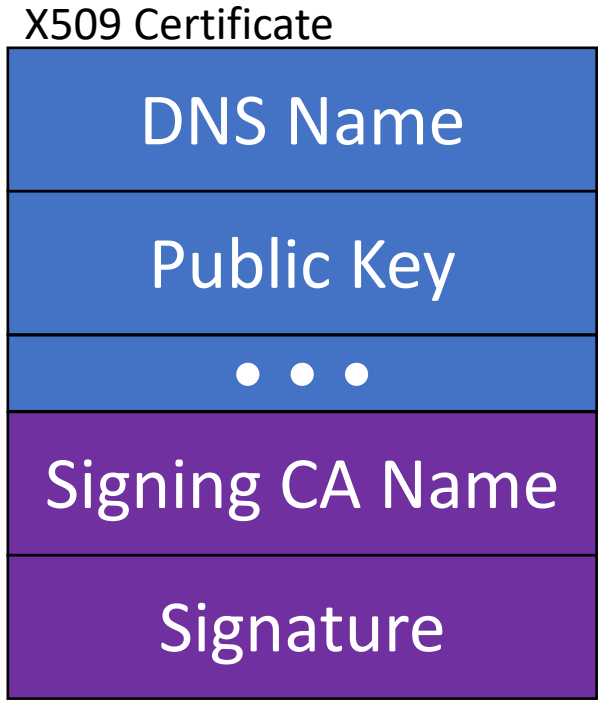
Certificate Authority

- A Certificate Authority (CA) confirms an entity's public key
  - Usually this will be a server's public key
- Companies get paid to do this
  - They "check out" the requestor
    - Now-a-days domain registrars provide this service
  - They issue a "certificate" with the information
  - Certificates are signed with the CA's private key
- CA's public key is "well-known"
  - It's in an additional certificate
  - Pre-installed or added to your configuration

Certificate

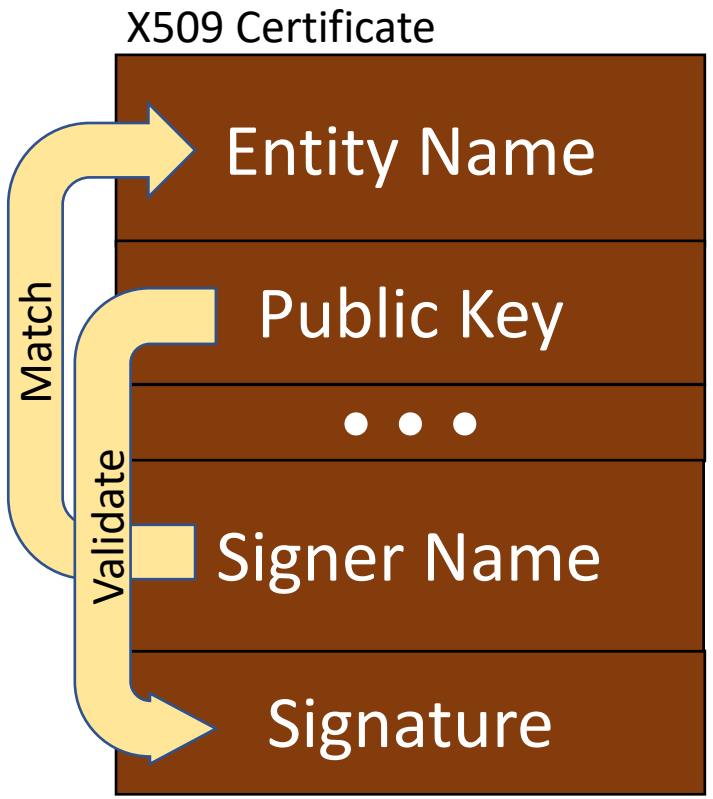
(Specifically an X509 Certificate)

- Contains information about an entity
  - Usually, the entity is a web server
- If the entity is a web server
  - Server's DNS name
  - Server's public key
  - … other identifying information
- It will also contain signature information
  - Name of signing certificate authority
  - Signature
    - Digest (one-way hash) of the above info signed using certificate authority's private key



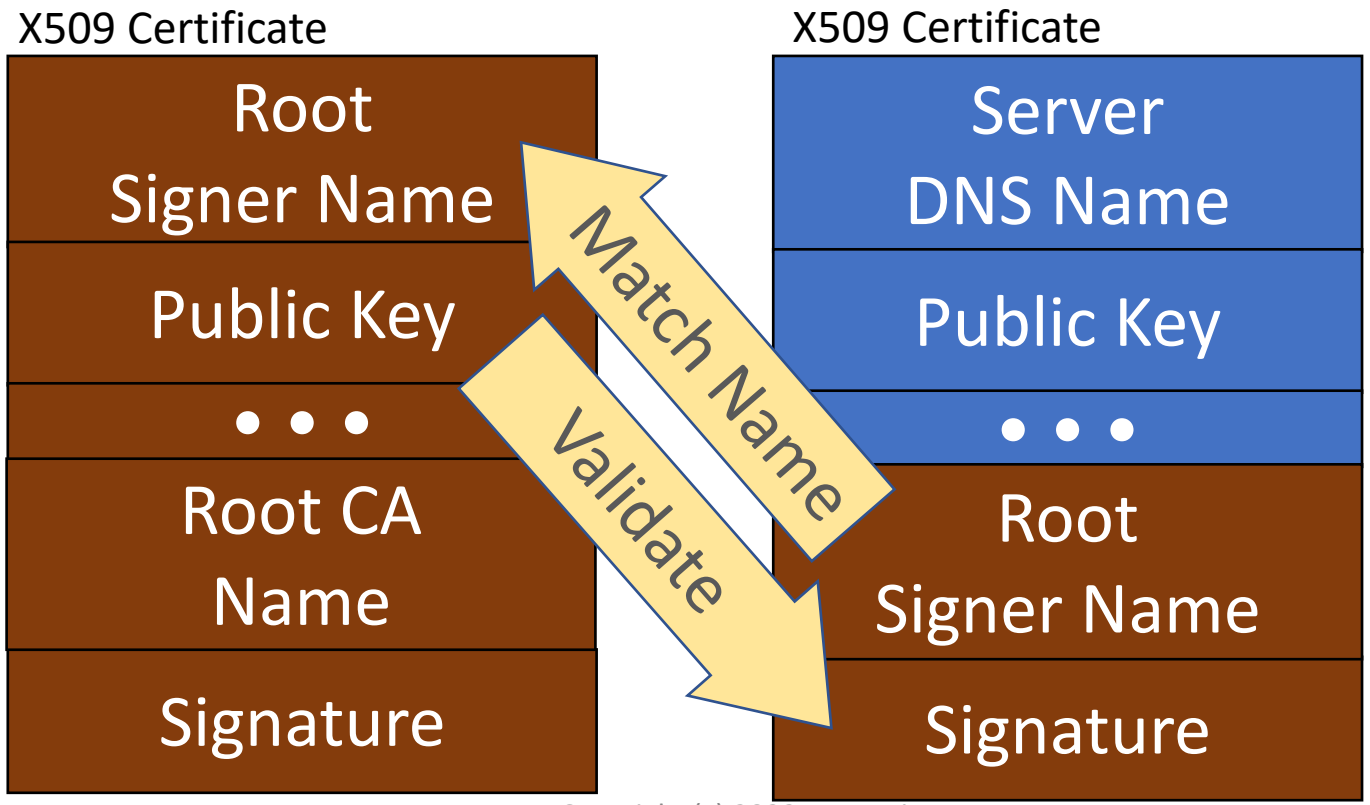
Certificate Chains

- Three types of certificates:
  - Root certificate (self-signed by a certificate authority, often pre-installed)



Certificate Chains

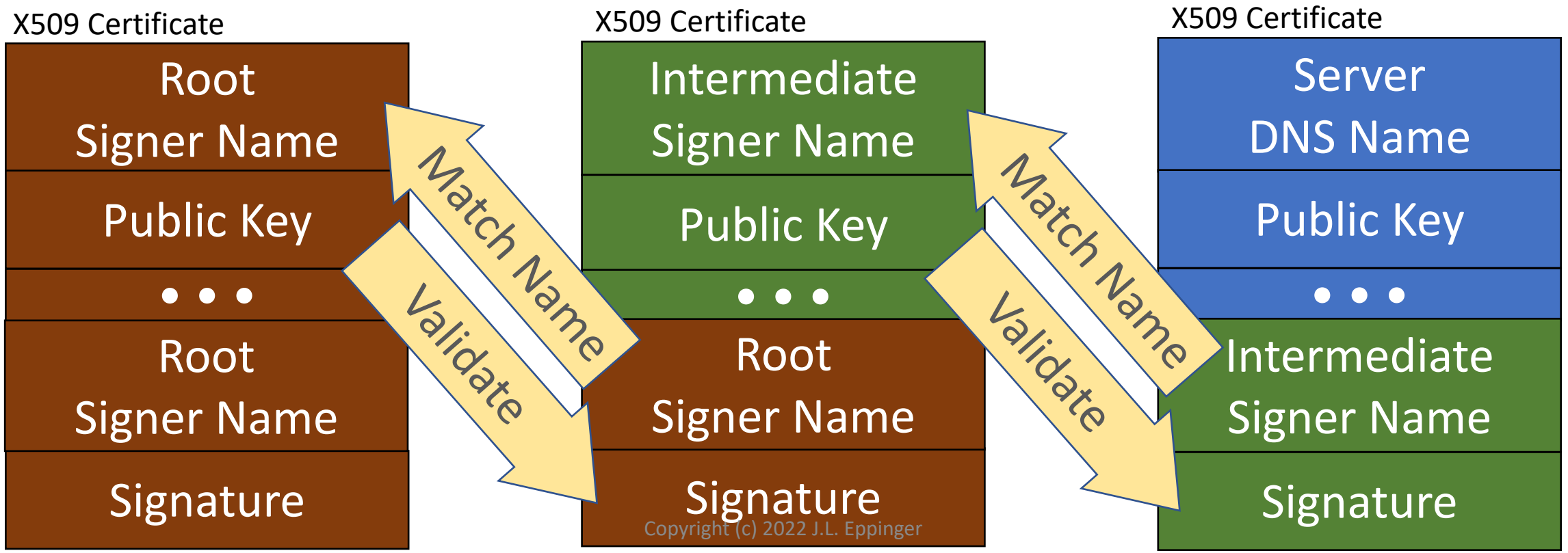
- Three types of certificates:
  - Root certificate (self-signed by a certificate authority, often pre-installed)
  - End-entity certificate (identifies the web server)



Certificate Chains

- Three types of certificates:
  - Root certificate (self-signed by a certificate authority)
  - End-entity certificate (identifies the web server)
  - Intermediate certificate (signs other certificates while protecting root private key)

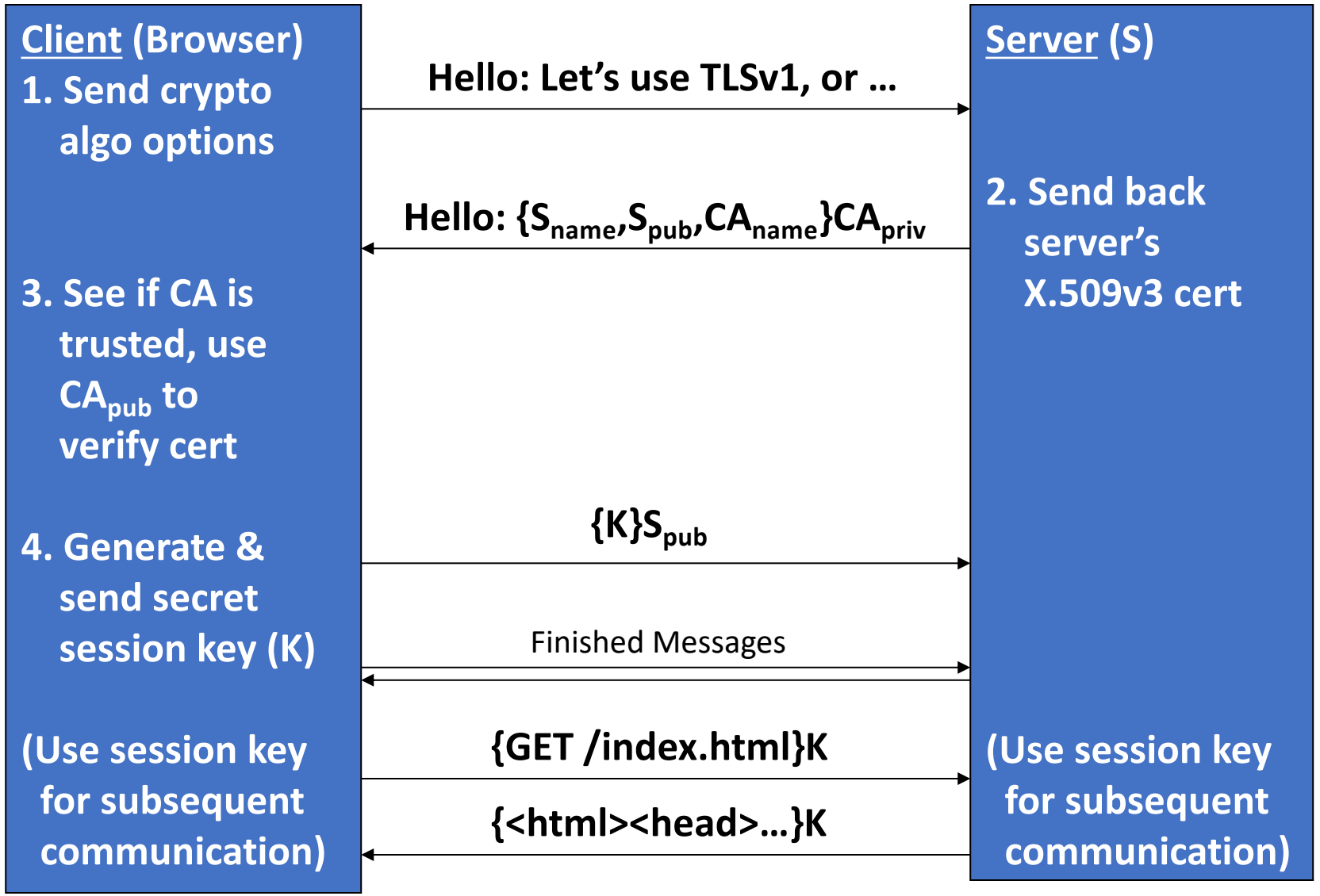
Just validating the signatures doesn't mean you can trust it!



Who Do You Trust?

- There's a list of trusted Certificate Authorities
  - Firefox comes with pre-installed certificates
  - On MAC there is the KeyChain which has pre-installed certificates
    - Safari and Chrome use the trusted certificates in the KeyChain
  - On Windows, there is a Certificate Manager / Certificate Store
    - Chrome and Edge use the trusted certificates in a certificate store
  - You can install additional certificates into your store/keychain
- One of the certificates in the chain must be found in the installed list

SSL – Secure Socket Layer



Notes

- “Finished” messages contain hashes of the messages to verify no tampering by intermediaries
- Optionally, client certificates can also be sent to authenticate the client to the server
- Certificate have a lot of additional information
  - Dates, versions, encryption function names
  - Alternative subject names

Secure Sockets Layer => Transport Layer Security

SSL was created by Netscape

It standardized by the IETF and named TLS

SSL and TLS provide:

- Server Authentication
- Client Authentication (optional)
- Private Communication

Let’s Check it Out…

- https
- Crypto algorithms used
- View server’s certificate
- View CA’s certificate
- View list of trusted CAs

Remember: There are two ciphers

- The expensive public-key cipher
  - Consists of two keys: one public, one private
  - These are each typically 1024-bit or 2048-bit keys
  - But has great key distribution properties
- The inexpensive symmetric cipher
  - These are typically 128-bit or 256-bit keys
  - Need to distributed the symmetric key
    - SSL uses public-key encryption to distribute the symmetric key

Approx Time to Crack AES (Symmetric Cipher)

- Assuming 10 billion guesses per second

Key Length	Time to Crack
56 bits	400 seconds
128 bits	10^{18} years
256 bits	10^{56} years

Source: EE Times, 2012

- Note: Age of universe is 10^{10} years

Setting up SSL on Django

- It's automatically set up when you deploy to Heroku
  - It's easily done in other PAAS environments, too : AppEngine
- On EC2, you can configure Apache HTTP Server to use SSL
  - You must get a CA to sign your certificate
  - You can usually pay your domain registrar to do this
    - I pay $200-$300/year for the certificates for cmu-webapps.org
  - You can use Let's Encrypt, the free CA
  - I have posted instructions for using these two options
    - See today's lecture materials on Canvas

Let’s Encrypt

- It’s a recent, free CA
- Let’s Encrypt requires you get a domain name for your site
  - You cannot use IP address or ec2-xxx-xxx-xxx-xxx.compute-1.amazonaws.com
- Let’s Encrypt has an automated process to verify your Apache server is running with the given DNS name and then to sign and install a certificate for that DNS name
  - The certificates are good for only 90 days but are easily renewed by running another script
- A significant fraction of certificates are now issued by Let’s Encrypt
  - See: https://letsencrypt.org/stats
  - My personal website uses Let’s Encrypt: https://www.jeffeppinger.com

Levels of CA Validation

- Domain Validated (DV)
  - CA checks that you own the domain using WHOIS or DNS
- Organization Validated (OV)
  - CA contacts the organization and confirms their identity
- Extended Validation (EV)
  - Two signed application docs + letter from accountant/notary/gov't
  - Verify signatures, WHOIS info, legal existence, financial info, physical existence
  - Certificate will only be valid for two years!
- Note: No one seems to care which you do
  - Most people are now just going with DV

What does SSL Give You?

- SSL can be used for any TCP/IP communication
- Once you have SSL
  - You have privacy
  - You have server authentication
- User authentication can be done using
  - Your own userids and passwords
  - Client certificate exchange added in SSL 3
    - We don’t see this often in the web-world

HTTPS Quiz

- See the link to the Google Form on Canvas
  - Modules => Lecture #16