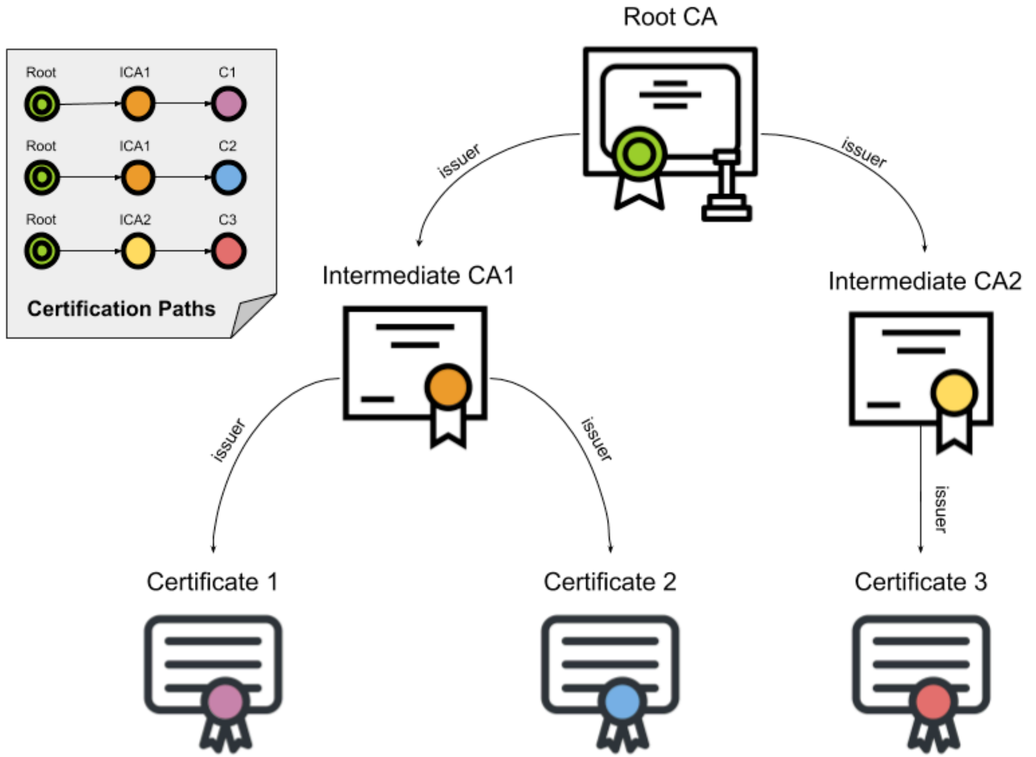
Javascript
Following up on URL/HTML

HTTPS and SSH

- HTTPS and SSH have the same basic approach
  - Do a public key / private key handshake
  - Exchange a symmetric key
  - Send bulk of material using symmetric key
- Problem
  - Public key???
  -
    - So how do you do "public / private handshake"?
      - Certificates!!

Certification Authorities

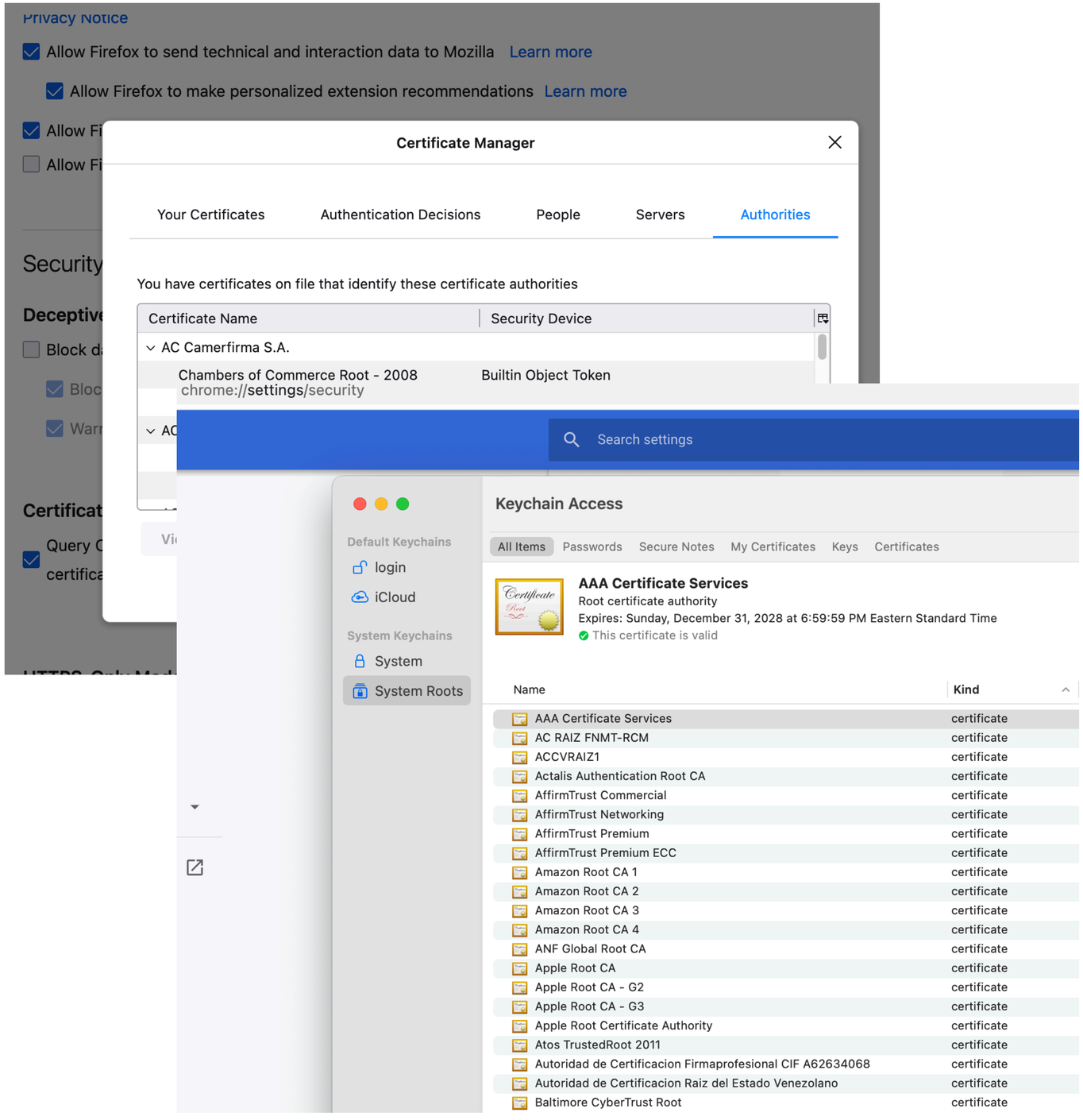
- Idea -- every browser is shipped knowing a set of "certification authorities"
  - Comodo SSL, RapidSSL, Thawte SSL, Sectigo SSL, GeoTrust SSL, Symantec SSL
- A site wanting to serve pages via https installs a "certificate" that is digitally signed in a chain leading to a root CA
- The certificate is sent to site user who can validate the cert by checking to see if it is signed by a root CA.
  - So the root CA list is the equivalent of the SSH "authorized_keys" file
  - Unpacking the certificate gets you
    - the public key of the sender without the sender having to pre-distribute the key OR simply send the key in clear text.
  - Other info about the sender (the origin URL)
    - Compare the origin URL in cert with actual location
  - NOTE: this does not guarantee that I am talking with the entity I intend to be talking.



https://www.ssl.com/article/browsers-and-certificate-validation/#:~:text=The%20browser%20verifies%20key%20usage,-The%20%E2%80%9Ckey%20usage&text=Examples%20of%20such%20purposes%20include,meant%20only%20for%20CRL%20signing.

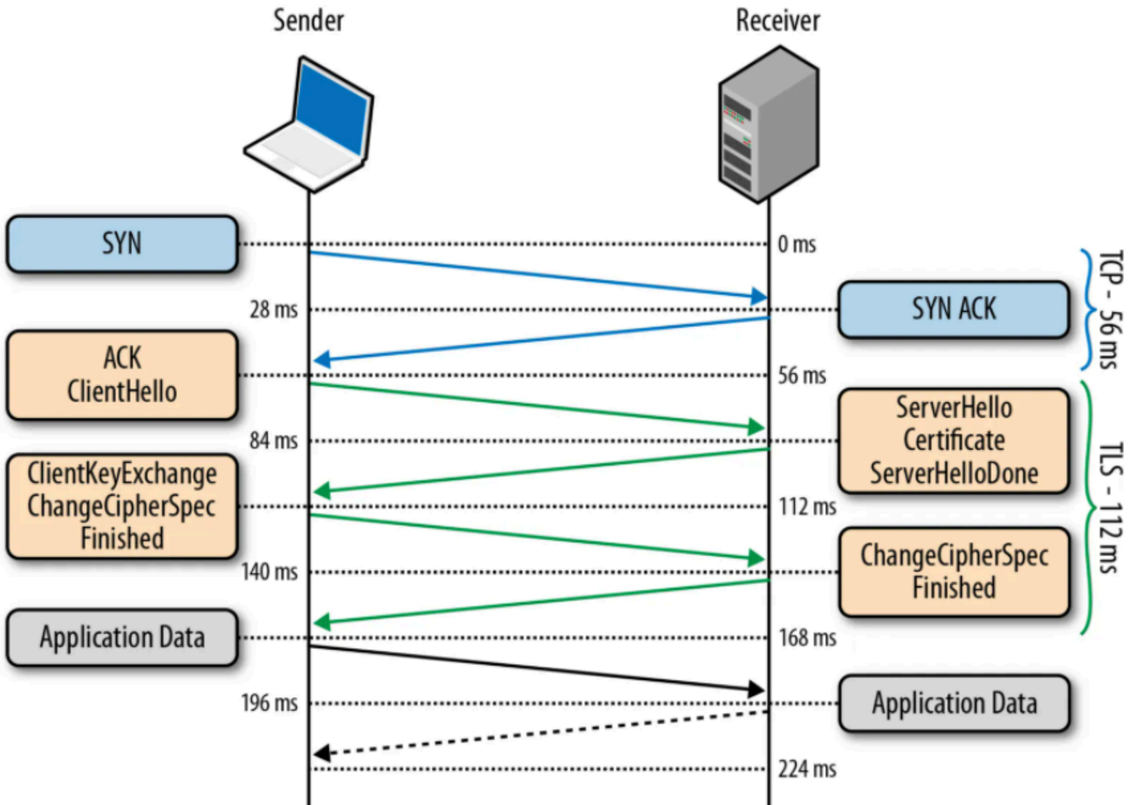
CAs

- There are lots
  - Firefox and Chrome have slightly different lists, and different ways of access



The cost of communicating

- Latency increase by switching to HTTPS : the initial SSL handshake (green) requires two (extra) roundtrips before the connection is established, compared to just the one roundtrip required (blue) to establish a TCP connection to the plain unencrypted HTTP port..
- Bandwidth Increase : The used bandwidth will increase slightly as the header size will increase by a number of bytes for protocol reasons and the effective payload will decrease a due to the framing overhead, and some ciphers will use padding as well. (max 6-7% increase in bandwidth).
- CPU Load : The most computational expensive part is the public key exchange, after which a relatively efficient symmetric cypher is used.



So for satellite internet 300km satellites =
2ms to go up to satellite and back

Suppose other parts of transmission = 26ms

Everything done on server or client takes oms

Then http requires at least 112 ms

https requires 224ms

Minimum 1/4 second

Note: 600km satellites add only 2ms per exchange total of 16ms

Costs (they multiply)

- Problem: Most web pages require loading a lot of other web pages.
- So that comm overhead is not just once
  - CNN: 316 (or more)
  - google: 23
  - Class web page: 12
- Fortunately most of this is done in parallel

HTTP is stateless and connectionless

- from protocol viewpoint every request is independent
  - a browser initiates an HTTP request and after a request is made, the client waits for the response. The server processes the request and sends a response back after which client disconnect the connection.
  - The server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different requests across the web pages.
- So you do the whole connection thing every time
  - The exchanged symmetric key is used
    - Client side:
      - encrypt the request headers and body sent up to server
      - decrypt server headers and response

Request & Response Headers

- Request Headers -- Information sent with your request
  - User agent string
  - What kind of material you will accept
  - site cookies

- Information about the page in addition to the page
  - cookies
  - meta info about page (last change) etc
  - header sent first, based on header browser may not ask for body
    - effectively another 2x28ms exchange

Cookies! (C is for Cookie, its good enough for me)

- A cookie is information saved by your web browser. When you visit a website, the site may place a cookie on your web browser so it can recognize your device in the future. If you return to that site later on, it can read that cookie to remember you from your last visit and keep track of you over time
- Most importantly, cookies are used to create the illusion of state.
  - Third party cookies
    - used by many ad companies and data brokers track you across the internet. They can see which sites you go to and use that to build a profile of you and your interests
  - Firefox, Safari, Brave -- NO
  - Chrome -- YES
  - Note: even without third party cookies and a non-chrome browser, Google knows who you are.

You	**Site**
Start	
	have a cookie
Store Cookie	
next page pass cookie	
	Get cookie you are logged in and have done something