Security Concepts

- HTTP/HTTPS
- Public Key Encryption
- Browser Certs

HTTP

HTTP is plain-text - everything is human readable

To anyone/anything able to read network traffic

Not great security

- Personal data
- Passwords

Can be altered as well as read

- MITM - "man in the middle" attacks

Can be copied and sent again

- "Replay attack"

HTTPS

"S" for Secure

Uses public-key encryption

Headers/bodies are encrypted

- Prevents reading

Sender each way can be validated

- Prevents alteration

Basic Encryption

"Ciphers" are simple encryption

"Corpus" (message) is altered by applying a set of rules

Decryption is applying the same rules in reverse

- ROT-13 (shift english alphabet 13 characters)
- "book codes" (convert letters/words to positions on page/text of a book)

Problems with basic encryption

- Depends on a shared secret (the rules or a key)
    - How do you initially exchange the secret?
- No proof sender isn't someone else with the shared secret

Hashing (One Way)

- "hashing" is not encryption
- A one way hash is a reductive transformation
    - Ex: group drivers by last plate digit
- Hashing alg a value always gives same result
- Knowing value means you can know hash result
- Knowing Hash result doesn't tell you orig value

Cryptographically Secure Hashing

- Hash algorithm + Hash Result
    - You can calculate a value to give hash result
    - Not the actual original value ("collision")
- Cryptographic secure - collision "difficult"
    - "difficult" changes over time
    - Better computers, math shortcuts
        - '91 - MD5 published
        - '04 - 1 hour to calc useful collision
        - '06 - 1 minute
        - today - seconds

Using Hashes

- Secure hashes are used verify information
    - Versus a potential intentional attack
- Insecure hashes are used to error-check values
    - "Checksum"
    - Valuable against typos, but not attackers

Public Key - Behind the scenes

Two core features:

- Two "keys"
    - Each unlock the others' messages
- Alg + one key doesn't reveal other key

TL;DR: Mathematical magic

A math problem that is hard to reverse

- 8134 cubed isn't too hard
- Cube root of 538161350104 is much harder

Public Key Essentials

Two "keys" (big numbers stored as big strings)

- A "public" key - **no secrecy**
- A "private" key - **never shared, ever**

You always need the OTHER key to decrypt

- Msg + **Public** key = encrypted value
  - Decrypt with matching **private** key
- Msg + **Private** key = encrypted value
  - Decrypt with matching **public** key

Signatures

- No shared secret
  - Public keys are PUBLIC
  - Private key not shared even with trusted
- Encrypt using target's public key
  - Only they can read (using their private key)
- Encrypt using your own with private key
  - Anyone can decrypt using public key
  - Your public key working proves you sent it
- Encrypt (w/ your private key) a hash of message
  - "Signature"

Browsers

- Browsers maintain a list of "trusted" public keys
    - Certificate Authorities (CA)
- HTTPS sites have a "Certificate" "signed" by a CA
    - "chain of trust" says key is from claimed site
- Browsers CAN be configured with a set key pair
    - Usually make one up for short-term use
    - Site identity validated
    - Browser/User identity is NOT validated
    - User IS trusted to be same user over session
    - Only same user has private key to decrypt

Getting a SSL certificate

- Used to cost $$$ (~$200/year)
    - Now available freely via **Let's Encrypt**
        - Many hosts offer for free w/o LE
    - Other Cert providers still exist ($)

https://letsencrypt.org/getting-started/

- Involved to set up
- Need a registrar (you "buy" the domain name)
- Need a host (who has the IP address)
- You put a value on website to verify yours to LE

Summary - HTTPS

- HTTP is plain-text - insecure
- HTTPS encrypts information **in transit**
- HTTPS uses **public key encryption**
    - Two keys, no shared secret
    - Prove sender or receiver
    - "Signatures" apply to unencrypted messages
        - Rely on "chain of trust"
    - Browsers trust a list of CAs
        - HTTPS is only as secure as the CAs
        - CAs validate identity w/signed "cert"
- letsencrypt.org provides free certs