# 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

# PUBLIC KEY - BEHIND THE SCENES

TL;DR: Mathematical magic

Imagine a math problem that is hard to reverse.

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

Same idea: A math problem that is easier to do in one direction

# PUBLIC KEY ESSENTIALS

Two "keys"

- A "public" key - **no secrecy**
- A "private" key - keep it secret

One-way encryption:

- Msg + **Public** key = encrypted value that needs **private** key to read
- Msg + **Private** key = encrypted value that needs **public** key to read

Notice how you need the OTHER key to decrypt

# WHAT IT MEANS

- No shared secret - public keys are PUBLIC
- Messages encrypted with private key more likely to be legit
- You can "sign" an unencrypted message by attaching an encryption of message/message checksum using your private key

# BROWSERS

- Browsers maintain a list of "trusted" public keys
    - Certificate Authorities (CA)
- HTTPS sites have a private key, along with a signed "Certificate" from a CA saying that key is theirs
- Browsers CAN be configured with a particular key pair, but generally make one up and share it with the site for short-term use
    - So site identity is validated via CA, but user identity is NOT.
    - ...but user is trusted to be the same user over duration of browser session

# SUMMARY

- HTTP is plain-text - insecure
- HTTPS protects information **in transit**
- HTTPS uses public key encryption
    - Browsers trust a list of CAs
        - HTTPS is only as secure as the CAs