Security: outline

- networking security
- security principles
- encryption
- authentication

networking security

- "in the clear" protocol can be easily broken when information is snooped: telnet, ftp, http, many email protocols
- encrypted protocols are secure against many attacks, including someone examining the data: ssh/scp, https, secure POP/IMAP, PGP
  - most protocols are not secure against *traffic analysis*
- *host security* is more concerned with installing applications, running foreign code, firewalls/NATs, etc
  - without host security it is hard to have network security

security principles

- it is usually better to have more security than less security
- security that inconveniences users is more likely to be resisted or circumvented
- security can lock out people who should have access
- data requiring security should not be sent unencrypted over the Internet
  - because some of the links may be accessible to adversaries
- data requiring security is still occasionally sent unencrypted over the Internet

security: attack and defense

- Alice, Bob, Charlie and Eve
- attackers just need one way to get information
  - may not be a direct way
  - some information gives access to other information
- defenders can set everything up the strongest possible way
  - which software to run
  - firewalls, ssh/ssl, etc
- knowledge gives power
  - whoever knew about heartbleed could use it to snoop

networking security

- given that the hosts are secure and the networks are not, can we communicate securely?
- authentication: who created this message?
  - digital signatures
- confidentiality: who can read this message?
  - different types of encryption
    - asymmetric (public-key) encryption, e.g. RSA, Elliptic Curves
    - symmetric (secret-key) cryptography, e.g. AES, DSA
  - generally regarded as safe if the (private) key is secret
    - may be vulnerable if quantum computing is successful

encryption

- mathematical function encrypt(K, m) gives c
  - to decrypt, decrypt(K,c) gives m
  - for public key systems, decrypt(K', c) gives m
- the only secret is the key, K or K'
- key must be chosen at random and have a sufficient number of bits
  - the number of bits depends on the technology of the day

one-time pad

- symmetric key system
- key must have as many bits as the message
- encrypt(K, m) = K XOR m
- decrypt(K, c) = K XOR c
- demonstrably secure as long as the key is only used once
  - what can happen if the key is used twice?

authentication

- auth(m,token) gives signed message
- verify(signed message, token) gives true/false
- if sender and receiver have the same token, compute h = hash(message + token)
  - send m = message + h
  - receiver compares hash(message + token) to h
  - only someone with the token can verify the message
  - if the hash is *cryptographically secure*, it is hard to obtain the token given only m