Embedded Systems

Security
&
Wireless / Interference

Jem Berkes
http://berkes.ca

ECE, University of Manitoba

Device Security

- Any computer on a network is at risk
- Embedded/IoT are feature-rich computers
- Worms and viruses will try compromising the device

IoT Incident in 2016

- Thousands of IoT cameras compromised
- Taken over by “botnet” software
- The attack took down part of the internet!

How do bad guys get in?

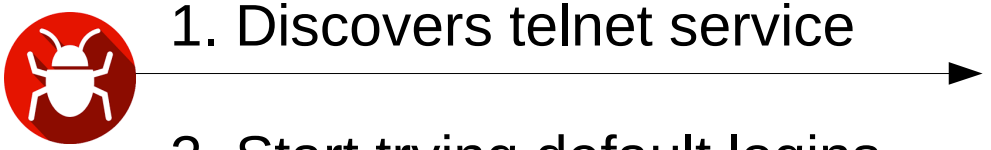
- Helps to understand some techniques attackers use
- Devices can be compromised many ways
- Three common types of vulnerabilities

Common vulnerability #1

- **Open service ports allowing logins**
  - ssh, telnet, http: login prompt
- *Plus* weak/default passwords

Common vulnerability #1

- **Open service ports allowing logins**
  - ssh, telnet, http: login prompt
- *Plus* weak/default passwords



1. Discovers telnet service

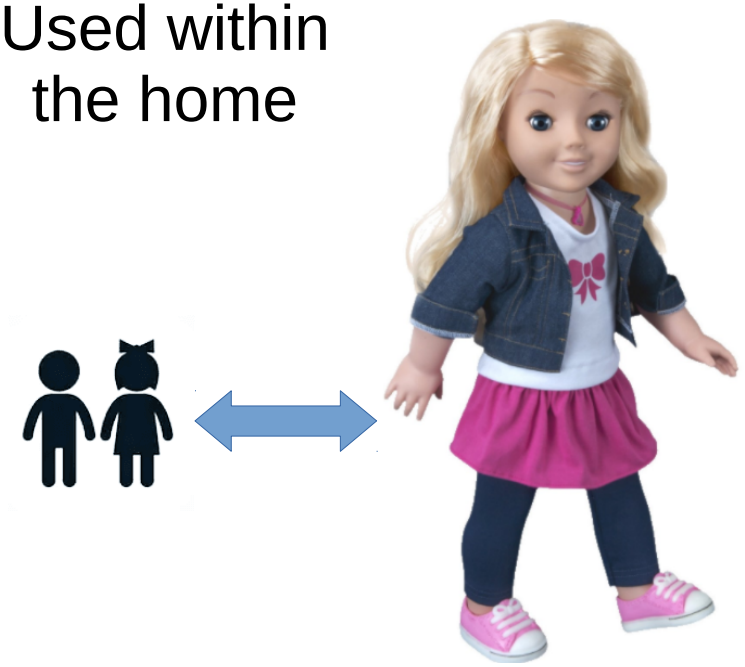
2. Start trying default logins
   admin : (no password)
   admin : admin
   ... brute-force search ...

3. If success, loads software

Common vulnerability #2

- **Unauthenticated open services**
- Anyone can connect!

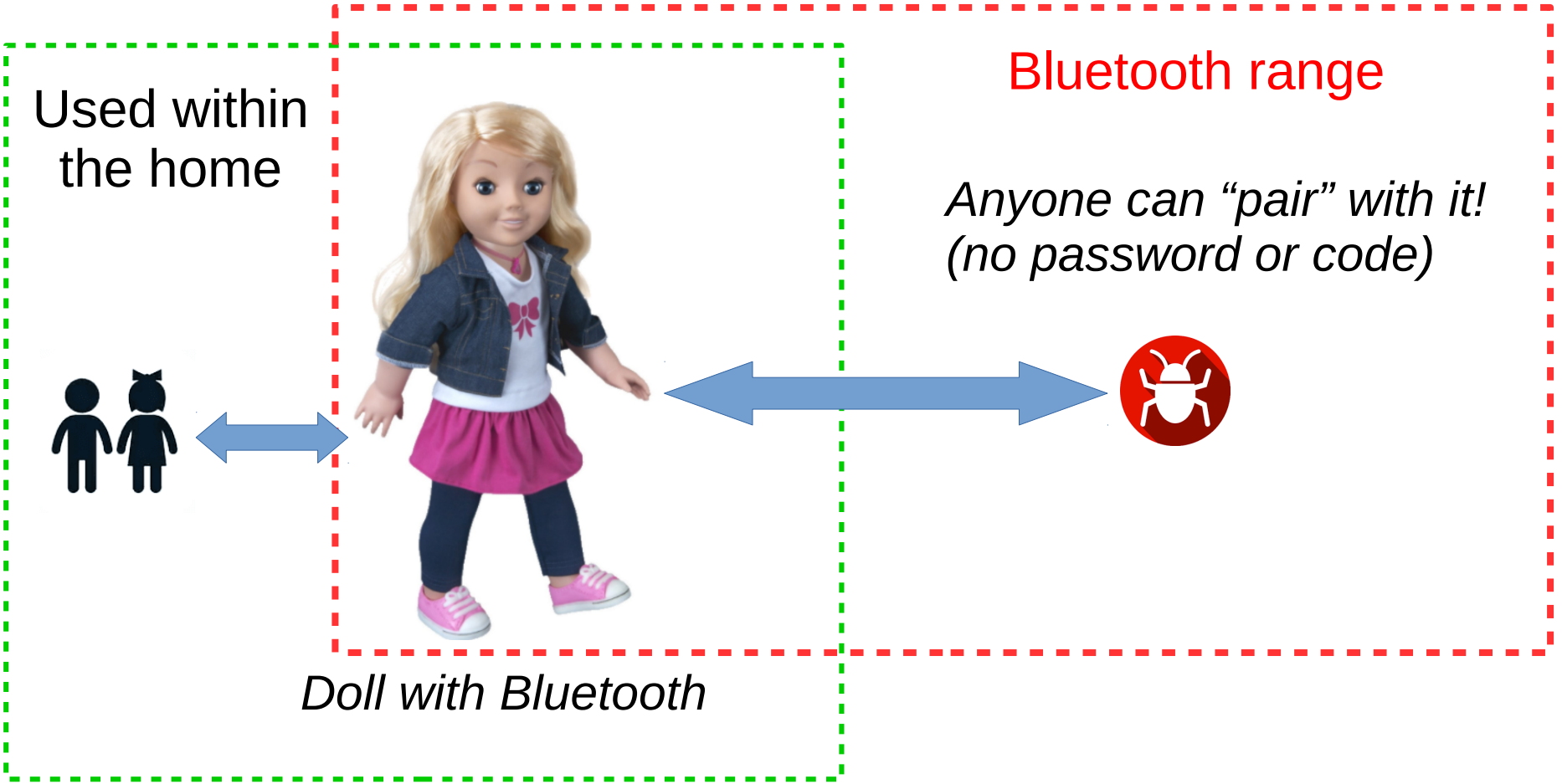
Used within the home



Doll with Bluetooth

Common vulnerability #2

- **Unauthenticated open services**
- Anyone can connect!



Common vulnerability #3

- **Outdated OS and software**
- Everything needs patching eventually
  - e.g. Wi-Fi “Krack”, major bug in Wi-Fi protocol
- Can’t just leave a device alone for 5 years

Solutions

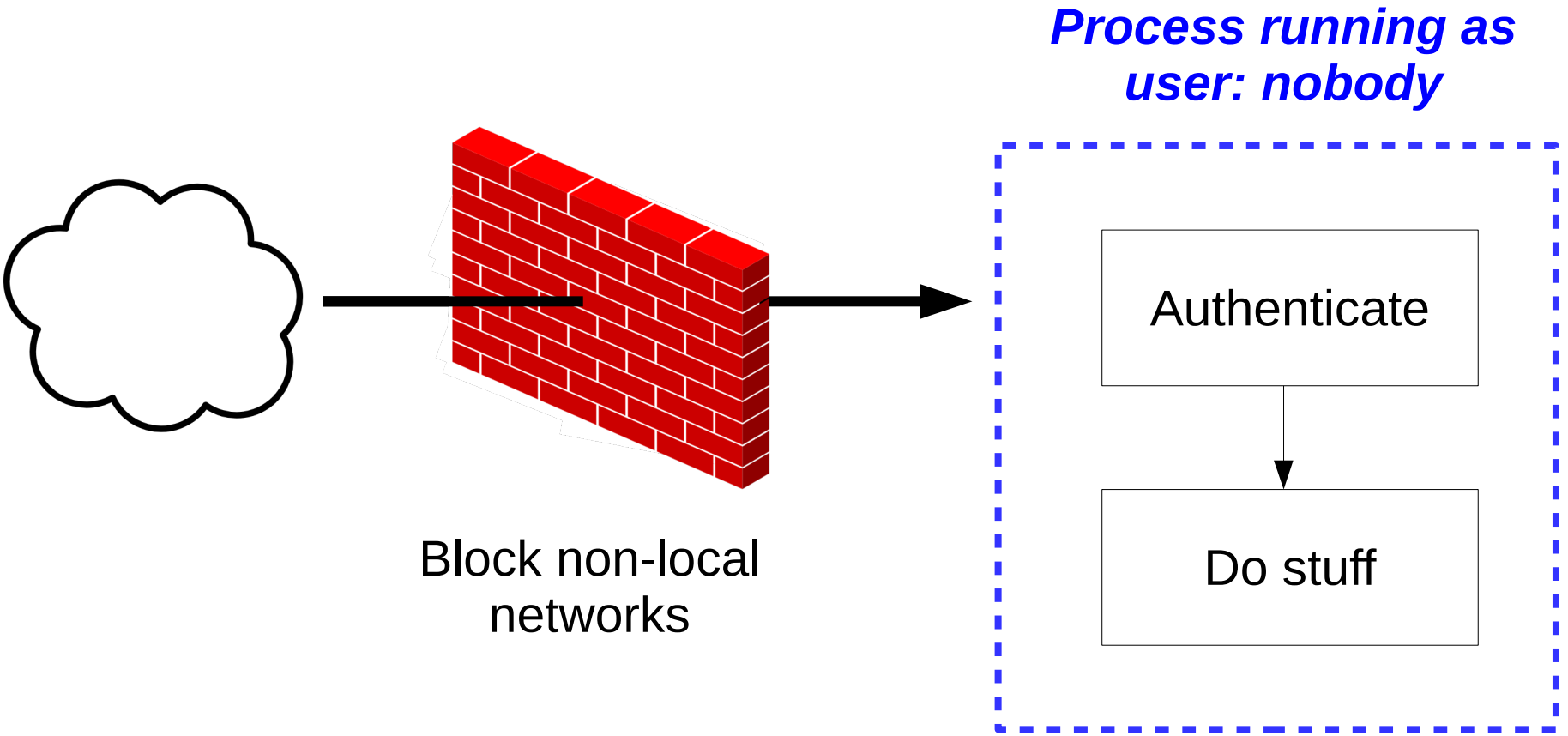
- Don’t run unnecessary services
  - Turn off services and wireless interfaces
- Use strong passwords
- **Don’t use default (hard-coded) passwords**
- Authenticate or validate connections
  - Don’t run totally open services
- Design with ability to update

Design for Security

- Many companies don’t get this right
- Easy to overlook
- Write security into the design process
  - Allocate time for it
- Rules of thumb
  - Don’t invent your own; use existing tech
  - Layered security

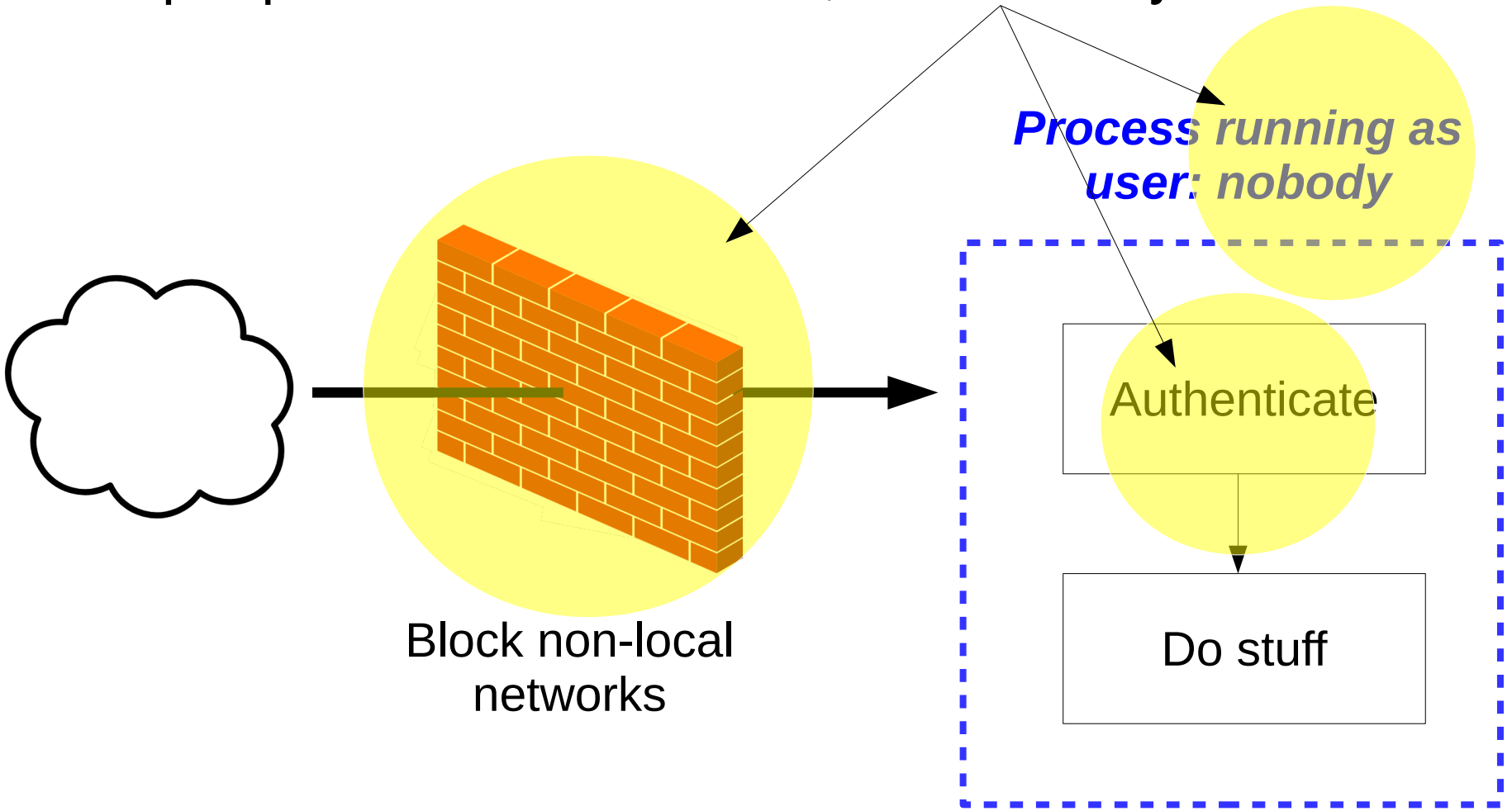
Layered Security

- Multiple protection measures; redundancy



Layered Security

- Multiple protection measures; redundancy



Wi-Fi Security

- Hot spot security modes
  - **Open**: no password, anyone can connect, unsafe
  - **WEP**: old standard, broken, unsafe
  - **WPA**: old standard, broken, unsafe
  - **WPA2-TKIP**: uses old algorithm, unsafe
  - **WPA2-AES**: currently best option (in 2020)

Wi-Fi: SSID

- SSID (Service Set Identifier) is hotspot name
- Publicly broadcast and visible to all
- Assume SSID is visible to everyone
- Hiding SSID doesn’t enhance security
    - In product, don’t use SSID to authenticate

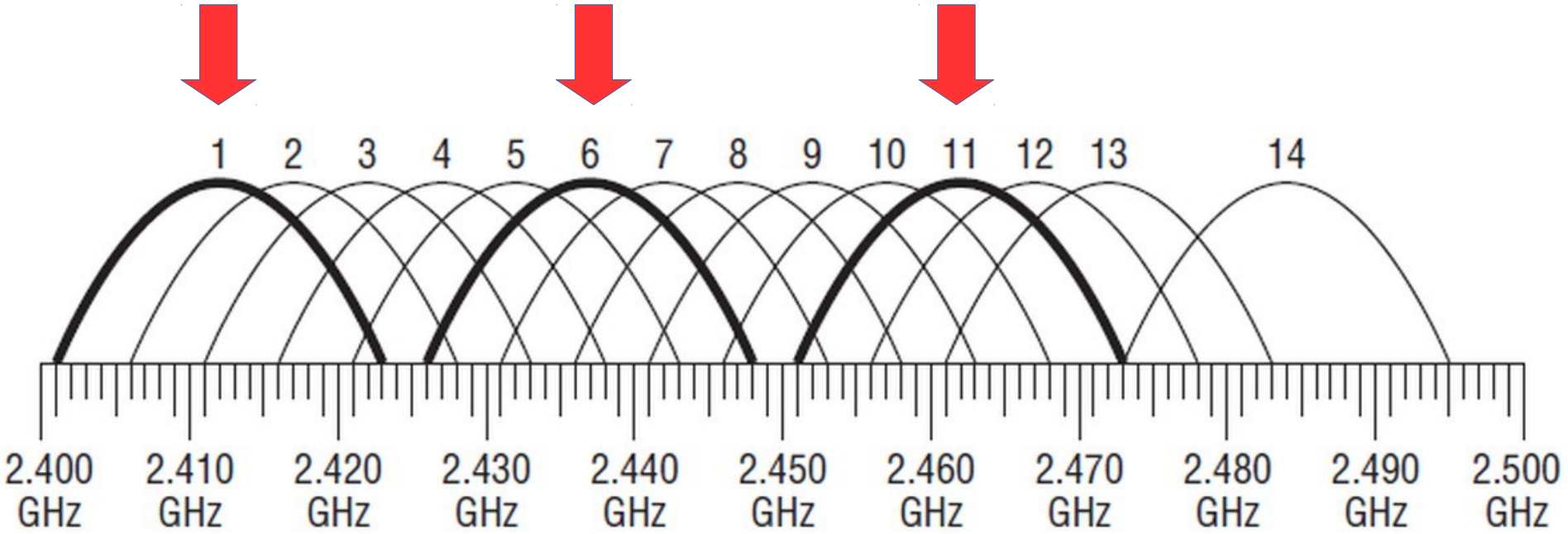
Wireless / Interference Issues

IoT & Wireless

- IoT is nearly always wireless
  - Wi-Fi
  - Bluetooth
  - Near Field Communication (NFC)
  - ZigBee
- Often competing in same frequency range

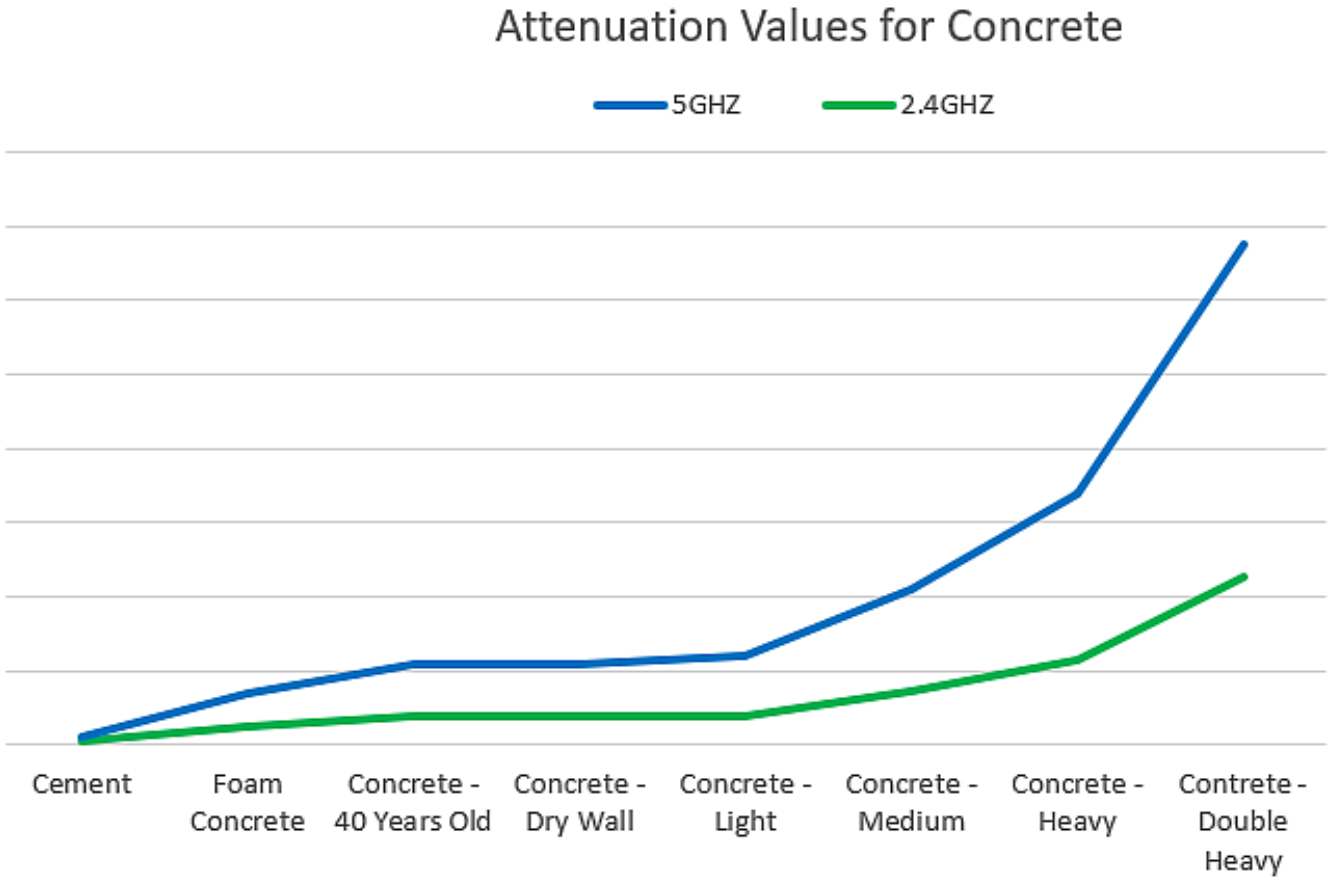
2.4 GHz Band

- Very crowded, interference is common!
  - Wi-Fi, Bluetooth, ZigBee, microwave ovens
- Try selecting the **channel**



5 GHz Band

- Less crowded, but still expect interference
- If using Bluetooth (2.4 GHz), use 5 GHz Wi-Fi
- Shorter range



Reducing Interference

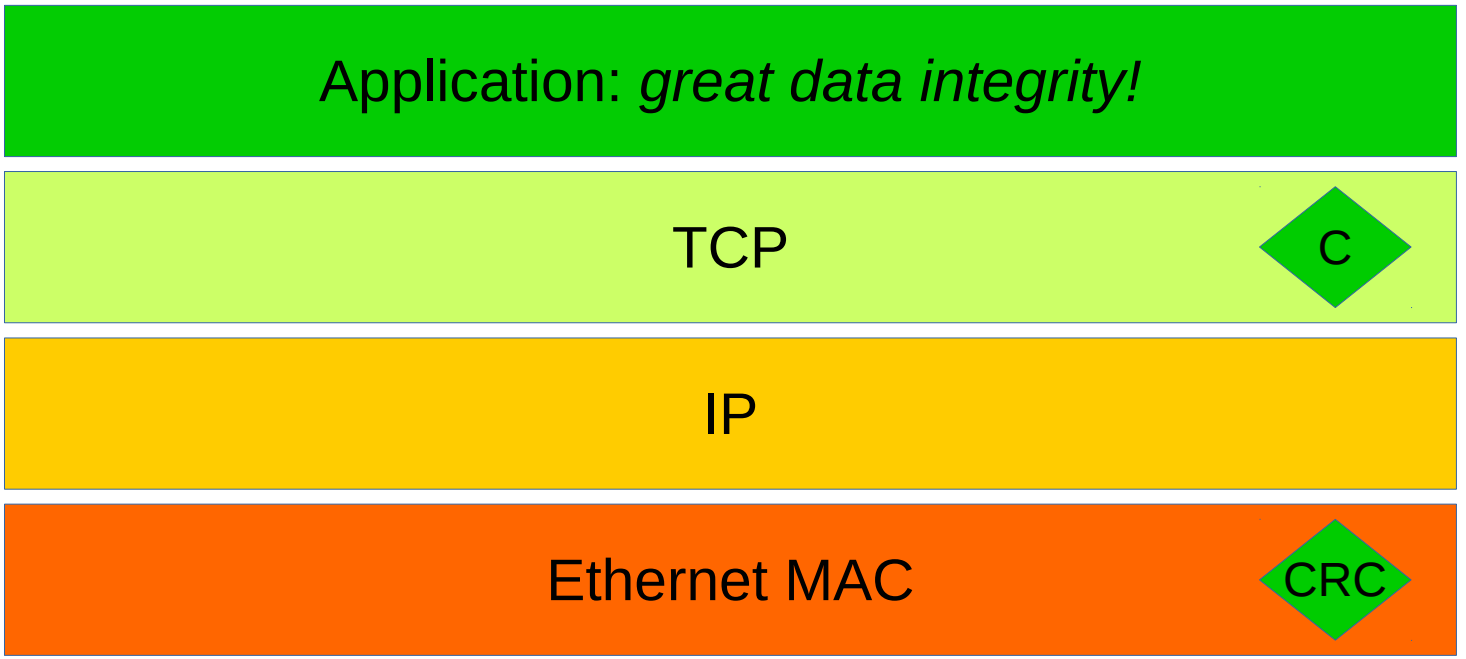
- Turn off radio interfaces you don’t need!
- Turn off transmitters when not needed
  - Sleep modes (Wi-Fi)
  - Saves power too

Software Design Considerations

- Assume radio interference will happen
- Expect outages/no connectivity
- Don’t assume continuous connections
  - Build this into your protocol
- Consider data integrity and corruption risk

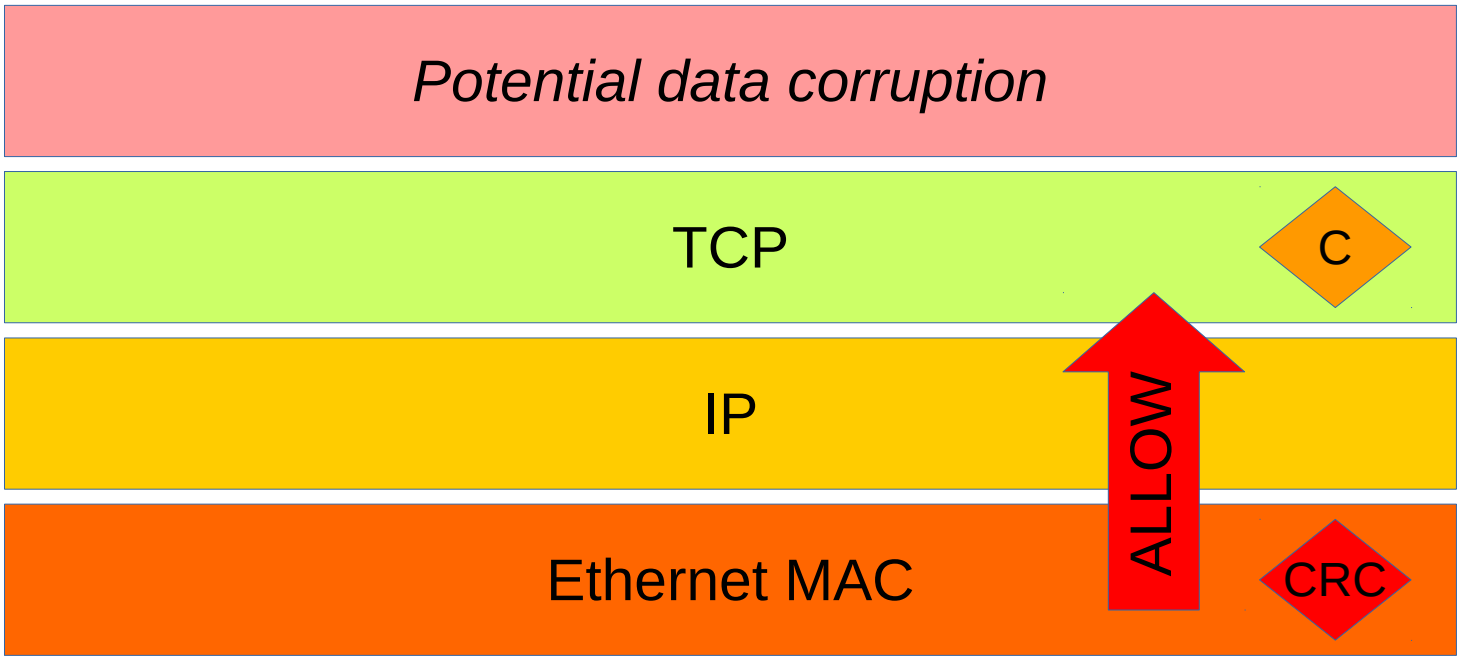
Data Integrity: Ideal

- Ethernet has strong CRC (error check)
- TCP has additional checksum, though weaker

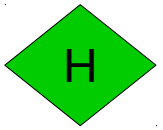


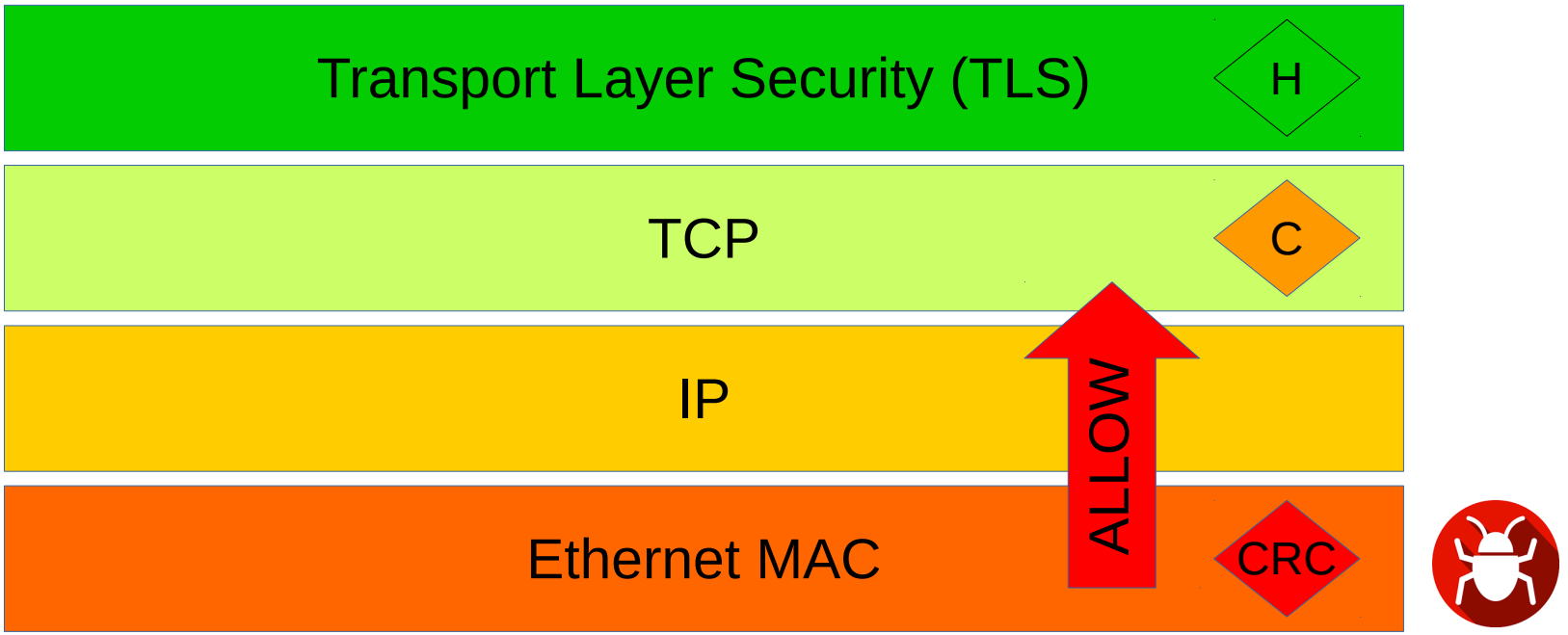
Data Integrity: Not So Ideal

- Bad frames come through (malicious? deliberate?)
- TCP checksum isn't strong enough



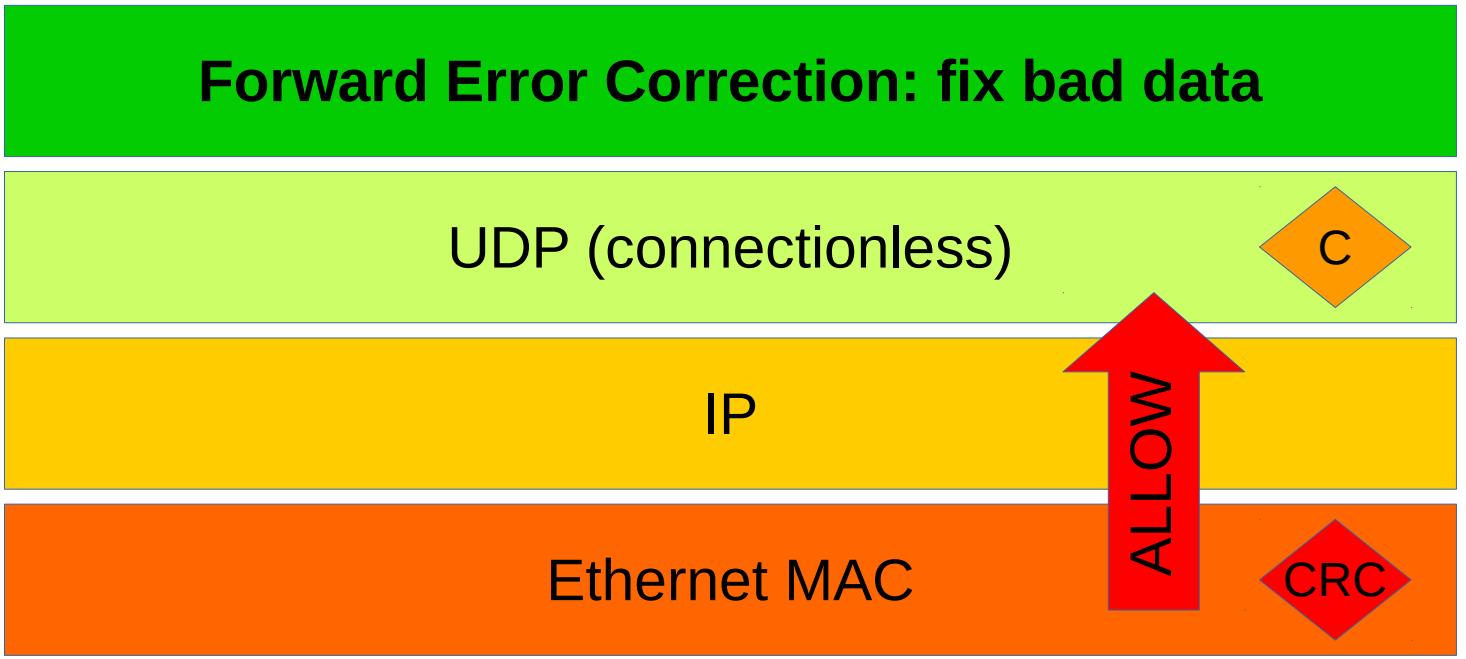
Can Add TLS

- Strong integrity and authenticity checks
- Hash will prevent corrupt data 



Low Latency

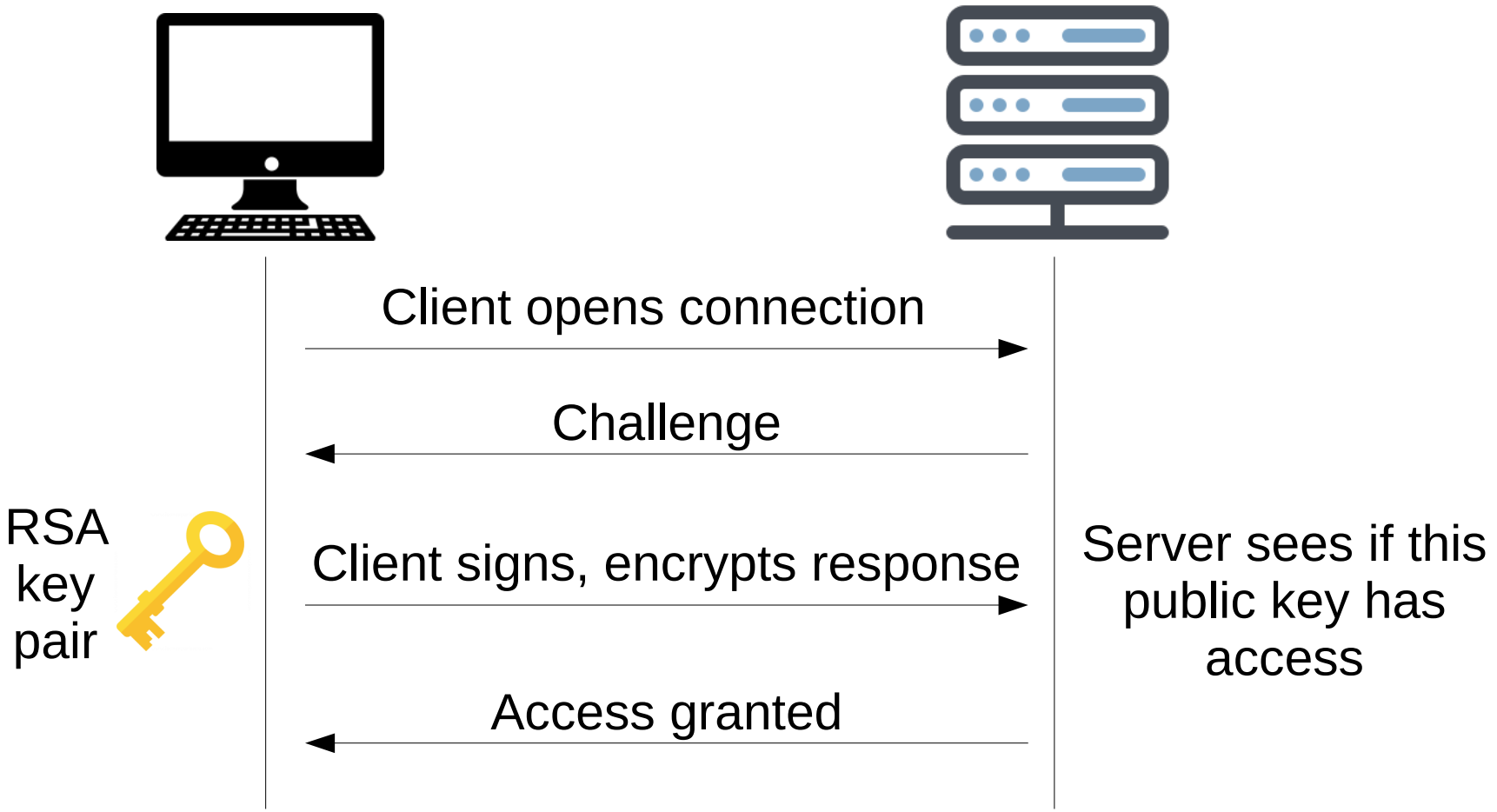
- *Deliberately* allow bad frames (don't drop them)
- Protect data at top layer, Forward Error Correction



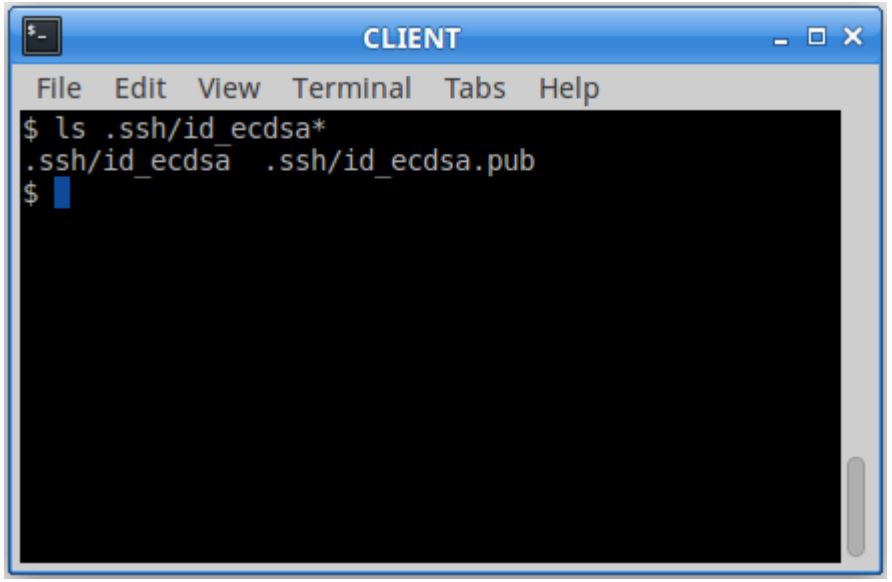
Extra Slides:
SSH

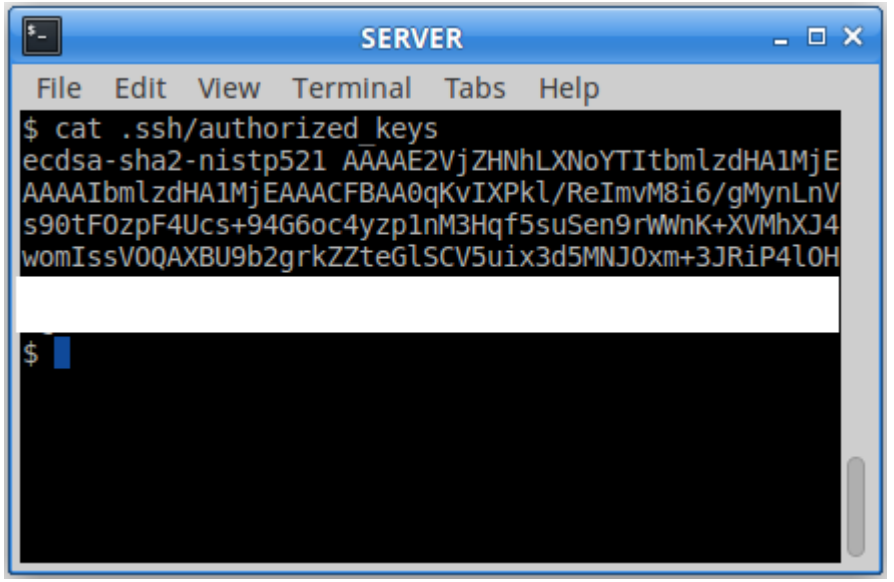
SSH Key-Based Authentication

- Much stronger than a password
- Eliminates the weak password / brute-force problem



SSH Key-Based Authentication





- Private and public key (.pub)
- Create with **ssh-keygen**

- Copy client's .pub key
- authorized_keys file