Authentication

Goal: Verify identity of User

- Something you know
  - Password
- Something you have
  - A key (car key, house key)
  - securID (RSA Token)
  - Credit card
- Something you are
  - Thumbprint
  - Eyeprint
  - Faceprint
- Multifactor authentication
  - Requires multiple forms of authentication
    - Generally means from different categories
    - Such as 2 factor authentication (like password + securID)

Authenticating Passwords

What does the server store?

- Raw password
  - Bad
  - System admins can read these
    - Evil ones can access accounts on other server if you reuse password
  - Attackers can use these to log into your account
- Store password hash
  - $H(password)$
  - $H$ is hard to invert
  - Low collision probability
  - Can be brute forced to get raw password
    - Also vulnerable to pre-computed tables
      - Attack users in parallel
- Store password hash + salt
  - $H(salt \ || \ password)$
  - Store random salt (1 salt per password)
  - Not vulnerable to pre-computed tables
    - Attack 1 user at a time
- Multiple hashing + salt
  - Store salt and $H^k(salt \ || \ password)$
    - $k$ is different for every user and is NOT stored
  - When a user logs in you keep applying the hash function until there is a match (will be after $k$ hashes)
    - Can have threshold so we don't run into an infinite loop on invalid password
Biometrics

- Unchangeable
- Public
  - Fingerprints
  - Iris prints
  - Faces
- Private (a bit more)
  - Blood vessel patterns
- Inherently shared across accounts
  - Might need to use the same fingerprint for different bank accounts
- Must store raw biometric scan
  - Insider threats
  - Can't hash because of the need for fuzzy recognition

Iris Biometrics

Take image of iris -> output 2048 bit string

\( x, x' \) (two different readings) for same iris

Hamming weight: the number of bits that are 1
\[ h(x \oplus x') \approx 200 \]
\[ h(x \oplus y) \approx 1024 \]

In networking we can use Error Correcting Codes to correct bit errors as a result of transmission
ECC(x) = c
Transmit
\( x | | c \)

Received as
\( x'| | c' \)

Assuming at most 2-3 bits are flipped
\( \text{ECC}^{-1}(x'| | c') = x \)

For iris scanning we can apply this concept to mask most bits as a hash
(This concept is called Fuzzy Extractors)
At registration we compute \( x, c=\text{ECC}(x) \)
store \( H(x), c \)

Login

We read \( x' \)
Let \( x''=\text{ECC}^{-1}(x'| | c) \)
Verify \( H(x'') \)
Can use this same concept to encrypt a hard drive (use $H(x) \mid \mid$ password) as encryption key. This way the hard drive is encrypted under password and biometric

Stored: $c$, salt, $E_k(k)$, $E_k(HD)$

Input: password, $x'$

$k' = H(password \mid |ECC-1(x' \mid |c)\mid |salt)$

Use $k'$ to get $k$, use $k$ to get $HD$

This solves the problem of not having to re-encrypt the entire HDD if $x$ is leaked, we just need to re-encrypt the key.

**Laptop authentication**

User wears bluetooth wristband
HDD encrypted too

Upon loss of contact laptop does the following
1. Encrypt RAM w/key $k$ stored on wristband
2. Delete its copy of $k$

Upon regaining connection
1. Retrieve $k$
2. Decrypt RAM