Applied Cryptography and Network Security

William Garrison
bill@cs.pitt.edu
6311 Sennott Square

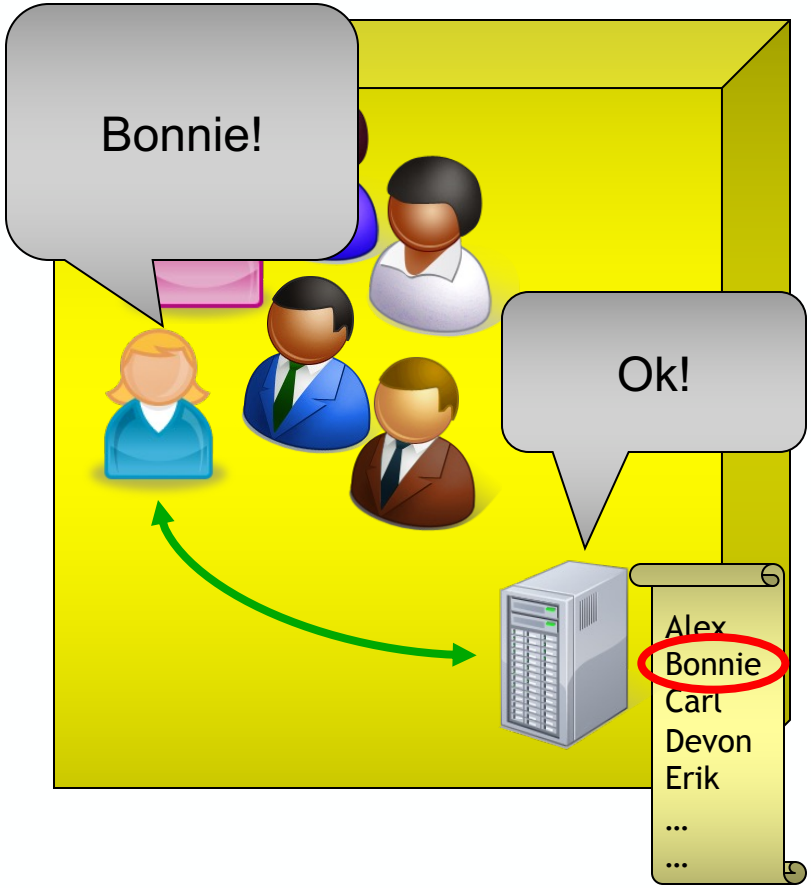
Authentication

University of Pittsburgh

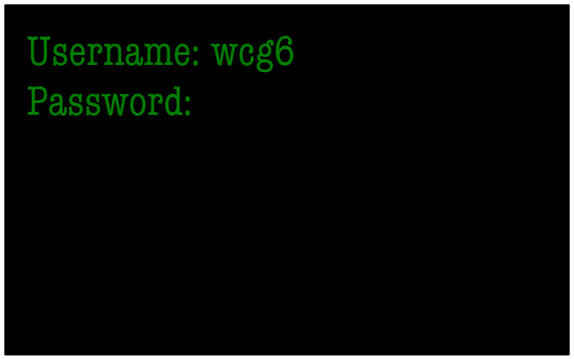
What is authentication? As related to identity?

Informally, identity defines who you are…

… while authentication is the process through which you prove it!



What's in a name?



Standard user IDs within a single domain

IP Address

Email Addresses

Subject: O=University of Pittsburgh,
OU=Computing and Info.,
OU=Computer Science,
CN=William Garrison

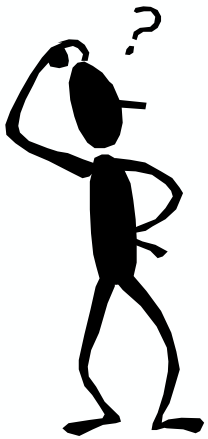
Digital Certificates

Authentication

Definition: Authentication is the process through which an identity is bound to a subject.

Since most computer security policy models are based on user identity, authentication is essentially the root of system security

Typically, people talk about three types of authentication:



Something you know

Something you have

Something you are

Something You Know

Informally, the system asks you a question that only you* know the answer to and verifies the correctness of the response

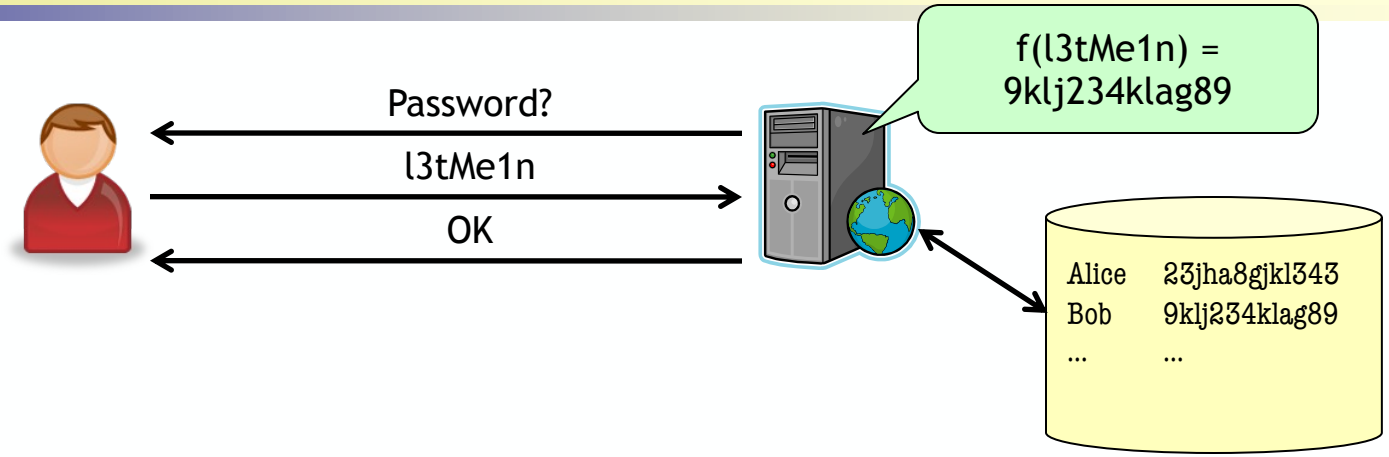
This by far the most commonly used authentication approach

- Passwords
- Passphrases
- PIN numbers
- “Site keys”
- …

Pro: These systems are easy to implement

Con: Many times, the assumption of secrecy is questionable

How does a password system work?



For each user, the password system stores complementary information

- Typically, it stores *f*(<password>) for some function *f*

Requirements on the password database

- Users do not need access
- Authentication process should have read access only
- Password update program should have write access

Although ubiquitous, password systems are not perfect

- Attacks possible against users and the system itself

Password systems assume that only the user knows their password

In general, people are helpful and thus subject to social engineering attacks

If users can choose their own passwords, they typically do an awful job

- For example, people choose passwords based on
  - Username and/or account names
  - Words from the dictionary (possibly with minor modifications)
  - Patterns from the keyboard (e.g., "asdfjkl;")
  - Family or pet names
  - Passwords from other accounts
  - …
- These are easy to guess!

As a result, the space of possible passwords is greatly reduced

- 26^8 = 208,827,064,576 8-character sequences
- About 29,000 8-character English words

Proactive password checkers try to eliminate these types of threats

Question: What are some issues surrounding proactive password validation?

Unfortunately, allowing the system to choose passwords doesn't buy us much…

In general, people are bad at memorizing random strings

- Studies show that the average person can remember about eight meaningful random things (characters, numbers, etc.)
- This is only one password!

These sort of defeat the purpose of a good password…

What does this mean?

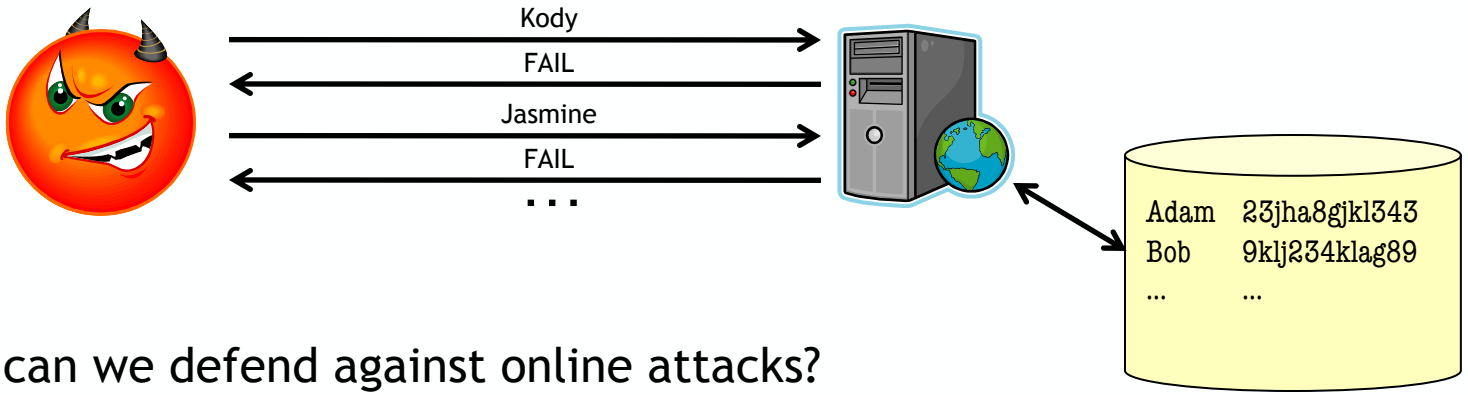
- The same password ends up getting used for multiple systems
- People write down their passwords

Writing down passwords doesn't always need to be a bad thing

- Allows for more secure passwords (too hard to remember)
- Keep password sheet in a secure place
  - Wallet, locked cabinet
- Use an (encrypted) password saving program
  - 1Password, KeePass, LastPass, Apple's keychain
  - Browser's autofill (sometimes)

If passwords are reusable (which most of them are) we can attack the system itself

Such an attack can either be conducted online or offline



How can we defend against online attacks?

- Make it expensive to carry out multiple guesses
  - Exponential backoffs in wait time
  - Solve a CAPTCHA or proof of work before password entry
  - …
- Lock accounts
  - Many ATMs "eat" cards after a set number of failures
  - Online banks often lock accounts after three failures

Although some attackers have amazing patience, online attacks are less fruitful than offline attacks

In an offline attack, the adversary has access to the complementary information associated with some account

The attacker can either guess passwords and check whether *f*(<pwd>) matches the complementary information, or she can use a precomputed mapping of complementary information to passwords

This is just a hash table lookup!

How can we defend against these lookup attacks?

- You can't stop the attacker, so you need to make her job costly
- e.g., force the computation of a new dictionary for each password

---

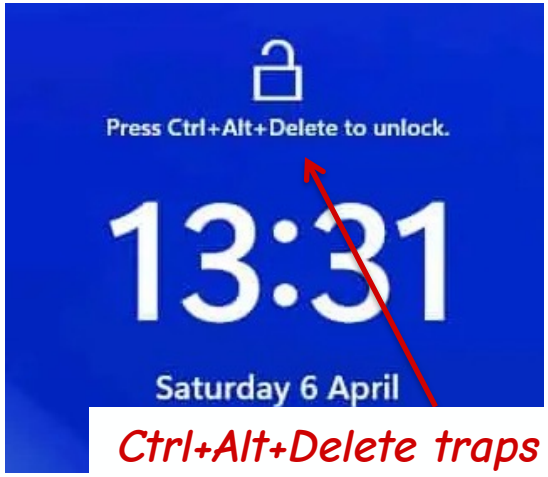
Example: Salting in older Unix systems

- The user's password is used as a key to the `crypt()` function
  - `crypt()` is effectively DES encryption 25 times
  - A random 12-bit salt is used to permute some of the DES tables
  - The result is that encryption depends on the key (password) and the salt (random)
- The username, salt value, and the "encrypted password" are stored
- Evolution of crypt approaches: bcrypt 1999, scrypt 2009, Argon2 2015

Question: What are the strong points of salting? The weaknesses?

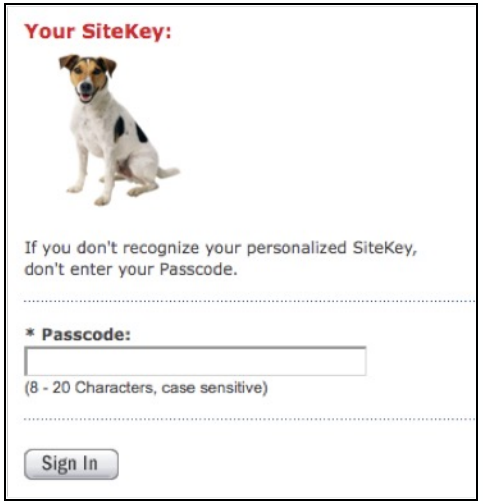
A password authenticates the user, but how does the user authenticate the authentication process?

Approach 1: Trusted paths in Windows



Ctrl+Alt+Delete traps to the OS and cannot be intercepted by other programs

Approach 2: SiteKeys



These approaches are a first line of defense that allows users to detect tampering with the authentication process

Question: What are some problems with these approaches?

Most password systems are weak because passwords can be used more than one time

Challenge/response systems can address this problem

- System sends user a challenge
- Users computes *f*(challenge, secret) and returns results
- System checks the correctness of *f*(challenge, secret)

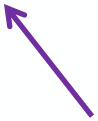
For such a system to be of any use, knowledge of old (challenge, response) pairs should provide no information about future login attempts

---

Example: Encrypting a random challenge using a shared key

- If a semantically secure encryption algorithm is used, the interception of one challenge reveals no information about other challenges

---



This is (roughly) how military identify-friend-or-foe (IFF) systems work

Something You Have

Challenge/response schemes often involve complicated calculations

- Digital signatures
- Repeated hashing
- Time-dependent functions
- …

Problems:

1. Password reuse leads to attacks
2. People are basically terrible calculators

Solution: Give people a *token* that does these things for them!

- Small device (hook to keychain, keep in wallet, etc.)
- Capable of performing “simple” calculations that people can’t do

You are probably somewhat familiar with token-based authentication

Magnetic strip cards are used at ATMs and for building entry

- Unique ID number stored on magnetic stripe
- ID number used as database key
- Access granted if database says so

Radio Frequency ID (RFID) cards are often used for “no touch” entry

- Many, many uses
  - EZ-Pass
  - Inventory control (libraries, warehouses, stores, etc.)
  - Pet identification
  - PRT bus passes
  - …
- Two types of tag: passive and active
- Many privacy concerns with some uses of RFID

2FA: What happens if a token is lost?

To protect against misuse of lost tokens, many token-based schemes rely on two-factor authentication

- Something you have (the token)
- Something you know (a PIN or password)

You use two factor authentication every time you go to the ATM

- First present the card, which defines who you should be
- Then enter the PIN, which verifies this identity

Two-factor authentication now has a fairly long history

- Google 2-Factor
- Pitt SSO
- Steam Guard
- …

Older (though not extinct) tokens were designed to be disconnected, single-purpose, and strongly secure

These devices can perform sophisticated calculations and are often able to solicit user input during the authentication process

- Typically, these were disconnected by design
- Tamper-resistant, no software update channel

Time-based one-time PIN

- Token contains a 128-bit secret
- Every 30-60 seconds, token generates a new pass code
- To log in:
  - Access service (which knows the same secret)
  - (Possibly) Enter PIN number or use fingerprint
  - Enter code from token
- "Duress PINs" help protect against physical threats

Challenge-response versions also exist (PIN pad for challenge)

Newer tokens are based on public-key cryptography to reduce the need for pre-shared secrets

FIDO Alliance: Open standards including U2F (universal 2nd factor) security keys using public-key crypto

- USB, NFC, or Bluetooth connection to exchange larger values
- Different keypair generated per site
  - For efficiency of storage, this may be stored by the service, encrypted with a key only the token knows
- How do we prevent MITM attack*?
  - Consider: A want to connect to B but E is impersonating; E connects to B pretending to be A, and passes on challenges and responses
  - Some protocols encode the service name or pubkey in the response, others check a signature on the session keys

WebAuthn ("passkeys") are meant to replace passwords day-to-day

- In addition to using a dedicated device, secrets can be stored in a protected region on a general computing device
  - What does this mean? Pros/cons?

Something You Are

Authentication through physical features is not new!

Imprinting

Uniforms

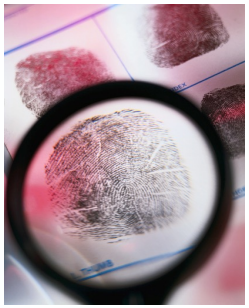
Survival

Furthermore, people often

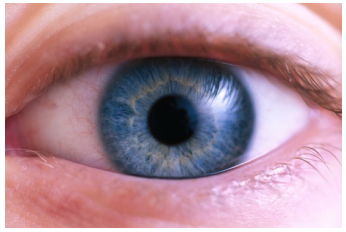
- Choose terrible passwords
- Forget good passwords
- Lose hardware tokens

So why not follow nature and identify people by physical makeup?

Biometric authentication does exactly this!

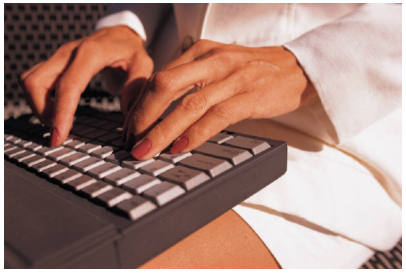


Fingerprints

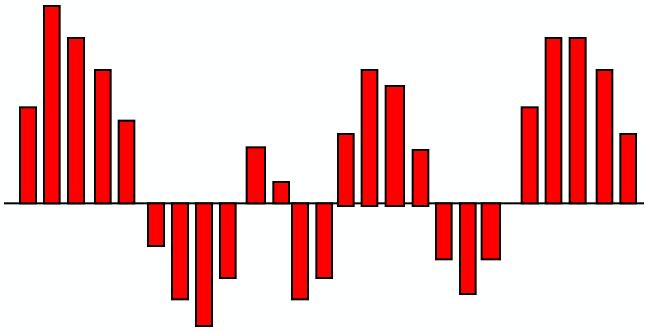


Iris/retina patterns

Hand geometry



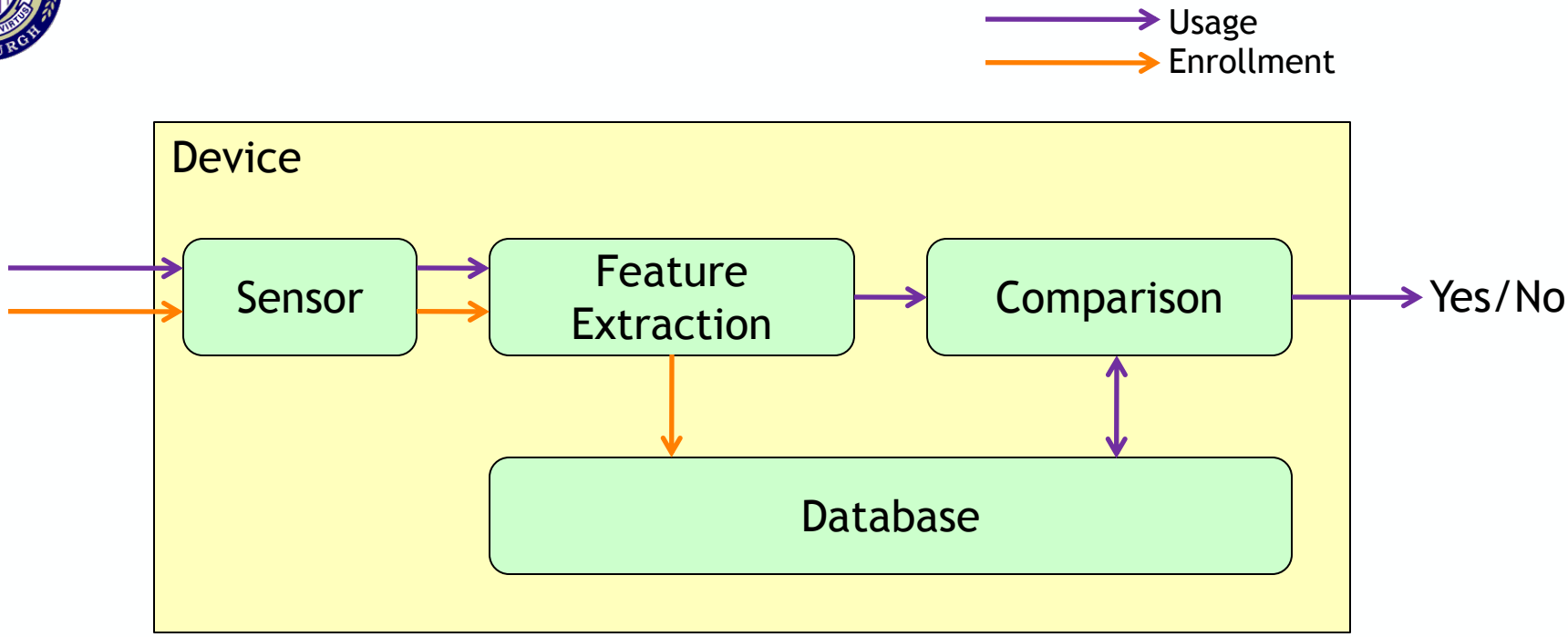
Typing patterns



Voice recognition

Facial recognition

Biometrics work by sampling some physical phenomena and comparing this sample with a recorded sample



What happens if we can "steal" a biometric sample?

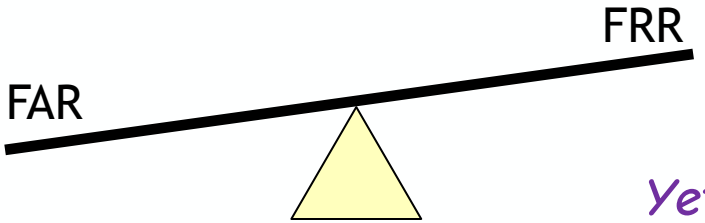
- If we can bypass the sensor, we're in trouble!

To protect against threats across devices, each database should be based on a unique salt (as in password systems)

When considering biometric authentication, several areas deserve special attention

False acceptance rate (FAR) versus false rejection rate (FRR)

- Each false acceptance is an unauthorized entry (bad)
- Each false rejection is a denial of service (also bad)



Yet another of Saltzer and Schroeder's principles…

How costly is the approach being considered?

Some biometrics are very intrusive, which leads to problems with psychological acceptability

- e.g., Retinal scans involve shining beams of light into the eye
- Not harmful, but many people do not like this
- Result: Retinal scans mostly used in high-assurance environments
- Face unlock

The security provided by biometrics depends on the environment in which they are deployed

In particular, two dimensions play a very large role

- Is the biometric login monitored or unmonitored?
- Will the biometric provide primary authentication, or is it secondary?

Example: The "stolen finger" attack

	Monitored	*Unmonitored*
Primary		
Secondary		

Conclusions

Passwords are the most commonly used method of authenticating users

A better approach is to combine something you know (a password) with something you have (an authentication token)

Biometric authentication (i.e., something you are) is a good secondary mechanism, but the FRR/FAR tradeoff makes their use as primary authenticators questionable

- Also good as one factor of two-factor authentication

My long-term predication for strong authentication (which seems to be coming true with WebAuthn):

- Something you have plus (something you know or something you are)