

# IT 4823

## Information Security Concepts and Administration

### Identification and Authentication



Notice: This session is  
being recorded.



Copyright © 2016 by Bob Brown



# Identity

- *Principal*: a unique entity, perhaps a person
- *Identity*: specifies a principal
- *Authentication*: binding of a principal to a representation of identity internal to the system;  
All access, resource allocation decisions  
assume binding is correct
- Identification: “*Who are you?*” [bbrown]
- Authentication: “*Prove it!*” [xyzy]

# Authenticating Identity

One or more of the following *factors*

- Something you **know** (*e.g.* password)
- Something you **have** (*e.g.* badge, smart card)
- Something you **are** (*e.g.* fingerprints, retinal characteristics)

Multi-factor authentication

- Two or more authenticators, *e.g.* something you know and something you have, such as password and smart-card.

# Uses of Identity

- Access control
- Accountability

# Authentication System

A 5-tuple:  $(A, C, F, L, S)$

- $A$ : information that proves identity
- $C$ : information stored on computer and used to validate authentication information
- $F$ : mapping function  
 $f: A \rightarrow C$
- $L$ : functions that tests identity  
 $l: A \times C \rightarrow \{\text{true}, \text{false}\}$
- $S$ : functions enabling entity to create or alter information in  $A$  or  $C$

## Simple Example

Password system, with passwords stored on line in clear text (very bad):

- $A$  is the set of strings making up passwords
- $C = A$
- $F$ : identity function  $\{ I \}$
- $L$ : single equality test function  $\{ eq \}$
- $S$ : function to set/change password

# Files and Objects

- Identity depends on system containing object, *i.e.* on implementation.
- Different names for one object
  - Human use, *e.g.*.. file name
  - Process use, *e.g.*.. file descriptor or handle
  - Kernel use, *e.g.*.. file allocation table entry, inode

# Users

- Exact representation tied to system
- Example: Unix systems
  - Login name: used to log in to system
    - Logging usually uses this name
  - User identification number (UID):  
unique integer assigned to user
    - Kernel uses UID to identify users
    - One UID per login name, but multiple login names may have a common UID



# Multiple Identities

Unix systems again

- Real UID: user identity at login, but changeable
- Effective UID: user identity used for access control;
- Saved UID: UID before last change of UID
  - Used to implement least privilege
  - Work with privileges, drop them, reclaim them later

# Groups

- Used to share access privileges
- First model: alias for set of principals
  - Processes assigned to groups
  - Processes stay in those groups for their lifetime
- Second model: principals can change groups
  - Rights due to old group discarded; rights due to new group added
  - This is a way to implement RBAC.
- A role is a group membership tied to function.

# Password Authentication

- Widely used user authentication method
  - user provides name/login and password
  - system compares password with that saved for specified login
- Authenticates ID of user requesting access and:
  - that the user is authorized to access system
  - determines the user's privileges, based on claimed identity.

# Passwords

- Sequence of characters:
  - Examples: 10 digits, a string of letters, *etc.*
  - Generated randomly, by user, by computer with user input
- Sequence of words:
  - Examples: pass-phrases “*open sez me!*”
- Algorithms:
  - Examples: challenge-response, one-time passwords

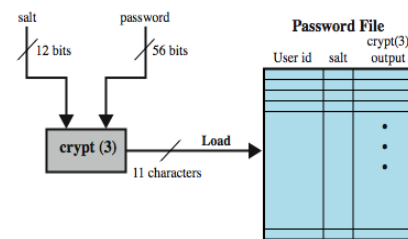
## Problem: Storage of Passwords

- Store as clear text
  - If the password file is compromised, *all* passwords are revealed
- Encrypt file?
  - Need to have decipherment, encryption keys in memory
  - Reduces to previous problem
- Store one-way hash of password
  - If file is compromised, attacker must still guess passwords or invert the hash (Note: automated “guessing” can be an exhaustive search.)

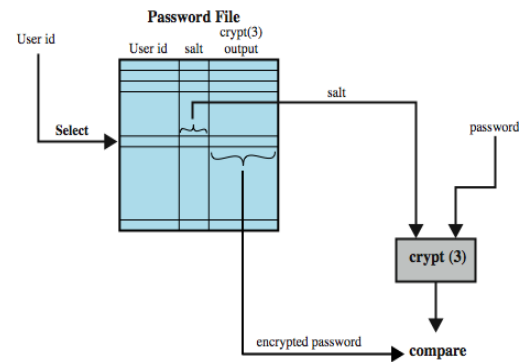
# Dictionary Attack

- An attacker compromises the (hashed) password file.
- The hash algorithm is known, so the attacker hashes every word in a large dictionary.
- If a hash in the password file matches a hash in the dictionary file, the corresponding dictionary word is the password. *Oops.*

# Use of Hashed Passwords



(a) Loading a new password



(b) Verifying a password

# Preventing Attacks

- Hide one of  $a$ ,  $f$ , or  $c$ 
  - Prevents obvious attack from above
  - Example: Unix/Linux shadow password file hides  $c$ 's
- Block access to all  $l \in L$  or result of  $l(a)$ 
  - Prevents attacker from knowing if guess succeeded
  - Example: preventing *any* logins to an account from a network
  - Prevents knowing results of  $l$  (or accessing  $l$ )
  - Not always practical



# Strength of a Password

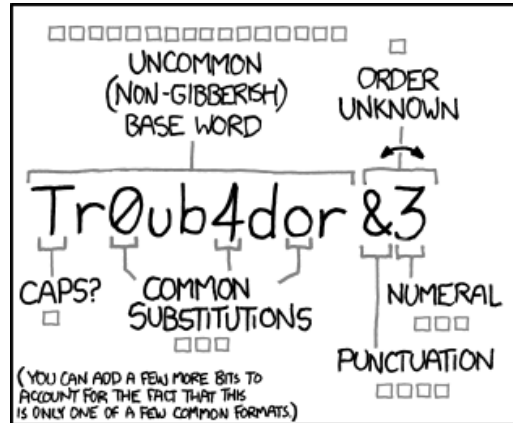
- Expressed as “bits of entropy,” number of possible values if the password is selected randomly.
- Example: A bank ATM PIN
  - Alphabet: decimal digits – 10 symbols, so 3.32 bits (roughly) per digit.
  - Four digits, so  $3.32 \times 4 = 13.28$  bits
  - About  $2^{13.28}$  or about 5,000 tries to guess a random PIN (5,000 is about half of  $2^{13.28}$ )
  - *Only* if the PIN is chosen randomly. (If I used my birth year, ten guesses would likely be enough.)

## Another Example

- Alphanumeric passwords (letters and numbers)
  - If case-insensitive, there are 36 symbols and 5.17 bits of randomness.
  - Make it case-sensitive: There are 62 symbols and 5.95 bits. Adding case-sensitivity increased entropy by less than one bit!
  - Eight character password:  $5.95 \times 8 = 47.6$  bits

## Yet Another Example

- Assume a password from the list of 500 most common passwords:  
<http://www.whatsmypass.com/the-top-500-worst-passwords-of-all-time>
- About 9 bits of entropy. ( $2^9 = 512$ )
- So, about 256 searches
- $\frac{1}{4}$  second at 1,000 per second!



~28 BITS OF ENTROPY

□□□□□□□□ □

□□□ □□□

□□□□ □

$2^{28} = 3 \text{ DAYS AT } 1000 \text{ GUESSES/SEC}$

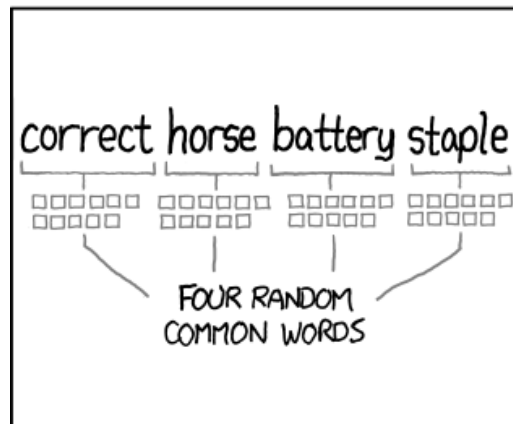
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)

DIFFICULTY TO GUESS: **EASY**

WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE 0s WAS A ZERO?

AND THERE WAS SOME SYMBOL...

DIFFICULTY TO REMEMBER: **HARD**



~44 BITS OF ENTROPY

□□□□□□□□□□

□□□□□□□□□□

□□□□□□□□□□

□□□□□□□□□□

$2^{44} = 550 \text{ YEARS AT } 1000 \text{ GUESSES/SEC}$

DIFFICULTY TO GUESS: **HARD**

THAT'S A BATTERY STAPLE.

CORRECT!

DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

[http://imgs.xkcd.com/comics/password\\_strength.png](http://imgs.xkcd.com/comics/password_strength.png)

## Strength of a Password

Anderson's formula:

- $P$  probability of guessing a password in specified period of time
- $G$  number of guesses tested in one time unit
- $T$  number of time units
- $N$  number of possible passwords ( $|A|$ )  
This is really the bits of entropy.
- Then  $P \geq TG/N$

# Example

- Goal
  - Passwords drawn from a 94-character alphabet
  - Can test  $10^9$  guesses per second
  - Probability of success to be  $\leq 0.5$  over a 365 day period
  - What is minimum password length required?
- Solution
  - $N \geq TG/P = (365 \times 24 \times 60 \times 60) \times 10^9 / 0.5 = 6.31 \times 10^{16}$
  - Choose  $s$  such that  $\sum_{i=0}^s 94^i \geq N$
  - So  $s \geq 9$ , meaning passwords must be at least 9 characters long. But *only* if they're random!

## Approaches: Password Selection

- Random selection:  
Any password from  $A$  equally likely to be selected
- Pronounceable passwords
- User selection of passwords
- Pass phrases

# Pronounceable Passwords

- Generate phonemes randomly
  - Phoneme is unit of sound, *e.g. cv, vc, cvc, vcv*
  - Examples: *helgoret, juttelon* are;  
*przbqxdf, zxrptglfn* are not
- Problems:
  - too few sequences
  - Equivalent sequences are not equally memorable:  
bliptab vs blitbap



# Users Make Bad Choices

- Users may pick short passwords
  - In one study, 3% were three characters or less, easily guessed
  - System can reject choices that are too short
- Users may pick guessable passwords
  - So, crackers use lists of likely passwords
  - One study of 14000 encrypted passwords guessed nearly 1/4 of them
  - It would take about 1 hour on fastest systems to compute all variants, and only need 1 break!

# User Selection

People pick easy to guess passwords:

- Based on account names, user names, computer names, place names
- Dictionary words (also reversed, odd capitalizations, control characters, “elite-speak”, conjugations or declensions)
- Too short, digits only, letters only
- License plates, acronyms, social security numbers
- Personal characteristics or foibles (pet names, nicknames, job characteristics, *etc.*)

# Users Pick Bad Passwords

*Pearls Before Swine* by Stephan Pastis

August 22, 2011



[http://www.gocomics.com/pearlsbeforeswine?ref=comics#mutable\\_661909](http://www.gocomics.com/pearlsbeforeswine?ref=comics#mutable_661909)

# Picking Good Passwords

- “LlMm\*2^Ap”
  - Names of members of two families
- “OoHeO/FSK”
  - Second letter of each word of length 4 or more in third line of third verse of Star-Spangled Banner, followed by “/”, followed by author’s initials
- What’s good here may be bad there:
  - “DMC/MHmh” is bad at Dartmouth (“Dartmouth Medical Center/Mary Hitchcock memorial hospital”)
  - Decatur, Georgia has DeKalb Medical Center
- Why are these now bad passwords everywhere?

# Proactive Password Checking

Analyze proposed password for “goodness”

- Always invoked
- Can detect, reject bad passwords for an appropriate definition of “bad”
- Discriminate on per-user, per-site basis
- Needs to do pattern matching on words
- Needs to execute subprograms and use results
  - Spell checker, for example
- Easy to set up and integrate into password selection system

## Example: *passwd+*

Provides a little language to describe proactive checking

- `test length("$p") < 6`  
If password under 6 characters, reject it
- `test infile("/usr/dict/words", "$p")`  
If password in file /usr/dict/words, reject it
- `test !inprog("spell", "$p", "$p")`  
If password not in the output from program spell, given the password as input, reject it (because it's a properly spelled word)

## About “Forcing” Compliance

- “Your password must be eight characters and must have at least one capital letter, one small letter, one digit, and one punctuation mark.” — *The Security Nazis*
- That is *less* secure than a *random* password from the 94 printable characters.
- Random password: about 6.55 bits of entropy per character, 52.4 bits for an 8-character password.

## That Restricted Password

- One capital letter 4.70 bits
- One small letter 4.70
- One digit 3.32
- One punctuation 5.05
- Four unrestricted 23.62
- Total 43.97 bits
- The unrestricted password, *if selected randomly*, is over 256 times more secure than the forced password.



# Password Aging

- Force users to change passwords after some time has expired
- How do you force users not to re-use passwords?
  - Record previous passwords
  - Block changes for a period of time (bad idea!)
- Give users time to think of good passwords
  - Don't force them to change before they can log in
  - Warn them of expiration many days in advance

# One-Time Passwords

- Password that can be used exactly *once*
  - After use, it is immediately invalidated
- Challenge-response mechanism
  - Challenge is number of authentications;  
response is password for that particular number
- Problems
  - Synchronization of user, system
  - Generation of good random passwords
  - Password distribution problem

22  
The average  
number of online  
passwords that  
each UK citizen  
has.

# How passwords are discovered...

Attackers use a variety of password discovery techniques, including the use of powerful tools that are freely available on the Internet.



## Social Engineering

Attackers can use social engineering skills to coerce users into revealing their passwords.



## Shoulder Surfing

Observing someone typing in their password.



## Manual Guessing

Attackers use personal information 'cribs' (such as name, date of birth, etc.) to guess common passwords.



## Key Logging

An installed keylogger intercepts passwords when they are typed into a device.



## Interception

Passwords can be intercepted as they are transmitted over a network.



## Brute Force

Automated guessing of billions of passwords until the correct one is found.



## Stealing Passwords

Attackers can steal passwords that have been stored insecurely. This can include handwritten passwords hidden close to a device.



## Searching

Searching IT infrastructure for electronically stored password information.

4  
Average no.  
of websites that  
users access  
using the same  
password.

BLACKLIST  
THE MOST  
COMMON  
PASSWORD  
CHOICES.

MONITOR FAILED  
LOGIN ATTEMPTS,  
AND TRAIN  
USERS  
TO REPORT  
SUSPICIOUS  
ACTIVITY.



CPNI

Centre for the Protection  
of National Infrastructure

## ...and how to improve your system security.

The following advice will reduce the workload on your users, making your system more secure as a result.

### Help users generate appropriate passwords

Put technical defences in place so that simpler passwords can be used.

Steer users away from choosing predictable passwords, and prohibit the most common ones.

Encourage users to never re-use passwords between work and home.

Train staff to help them avoid creating passwords that are easy to guess.

Be aware of the limitations of password strength meters.

### Help users cope with 'password overload'

Only use passwords where they are really needed.

Use technical solutions to reduce the burden on users.

Allow users to securely record and store their passwords.

Only ask users to change their passwords on indication or suspicion of compromise.

Allow users to reset passwords easily, quickly and cheaply.

DON'T STORE  
PASSWORDS  
IN PLAIN TEXT  
FORMAT.

USE ACCOUNT  
LOCKOUT,  
THROTTLING OR  
MONITORING  
TO HELP PREVENT  
BRUTE FORCE  
ATTACKS.

PRIORITISE  
ADMINISTRATOR  
AND REMOTE  
USER  
ACCOUNTS.

CHANGE ALL  
DEFAULT  
VENDOR-  
SUPPLIED  
PASSWORDS  
BEFORE DEVICES  
OR SOFTWARE  
ARE DEPLOYED.

© Crown Copyright 2015

[https://www.gov.uk/government/uploads/system/uploads/attachment\\_data/file/458857/Password\\_guidance\\_-\\_simplifying\\_your\\_approach.pdf](https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/458857/Password_guidance_-_simplifying_your_approach.pdf)

# Hardware Support

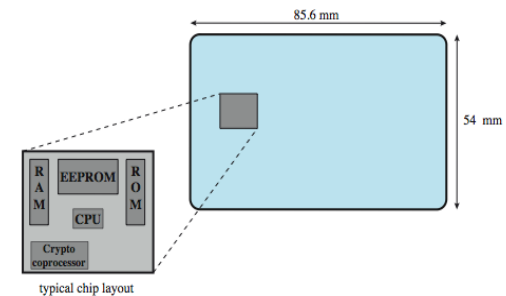
- Token-based
  - Used to compute response to challenge
    - May encipher or hash challenge
    - May require PIN from user
  - magnetic stripe card
  - memory card
  - smartcard
- Temporally-based
  - Every minute (or so) different number shown
    - Computer knows what number to expect when
  - User enters number and fixed password

# Memory Card

- Stores but does not process data
- magnetic stripe card, *e.g.* bank card
- Electronic memory card
- Used alone for physical access
- With password/PIN for computer use
- Drawbacks of memory cards include:
  - need special reader
  - loss of token issues
  - user dissatisfaction

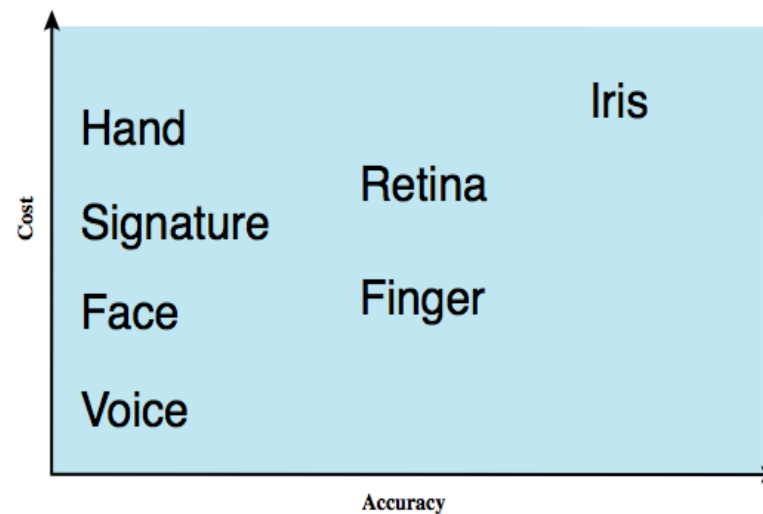
# Smartcard

- Credit-card like
- Has own processor, memory, I/O ports
  - wired or wireless access by reader
  - may have crypto co-processor
  - ROM, EEPROM, RAM memory
- Executes protocol to authenticate with reader/computer



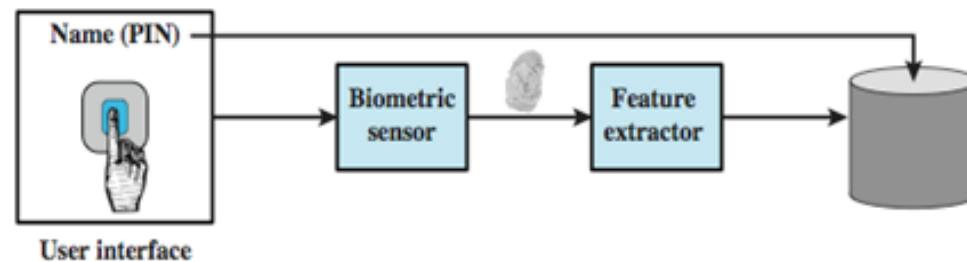
# Biometric Authentication

- Authenticate user based on one of their physical characteristics

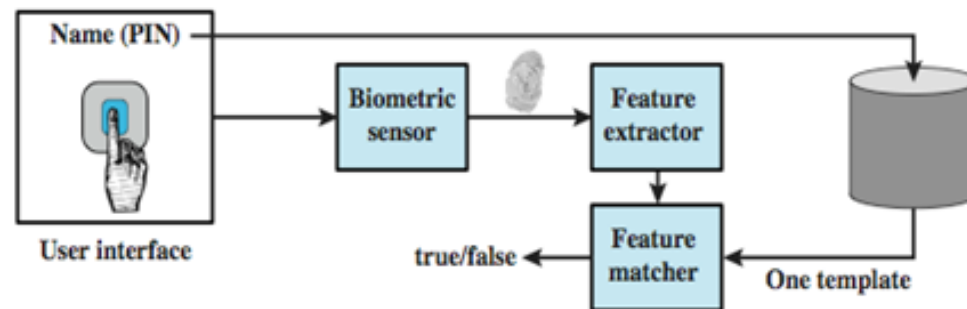




# Operation of a Biometric System

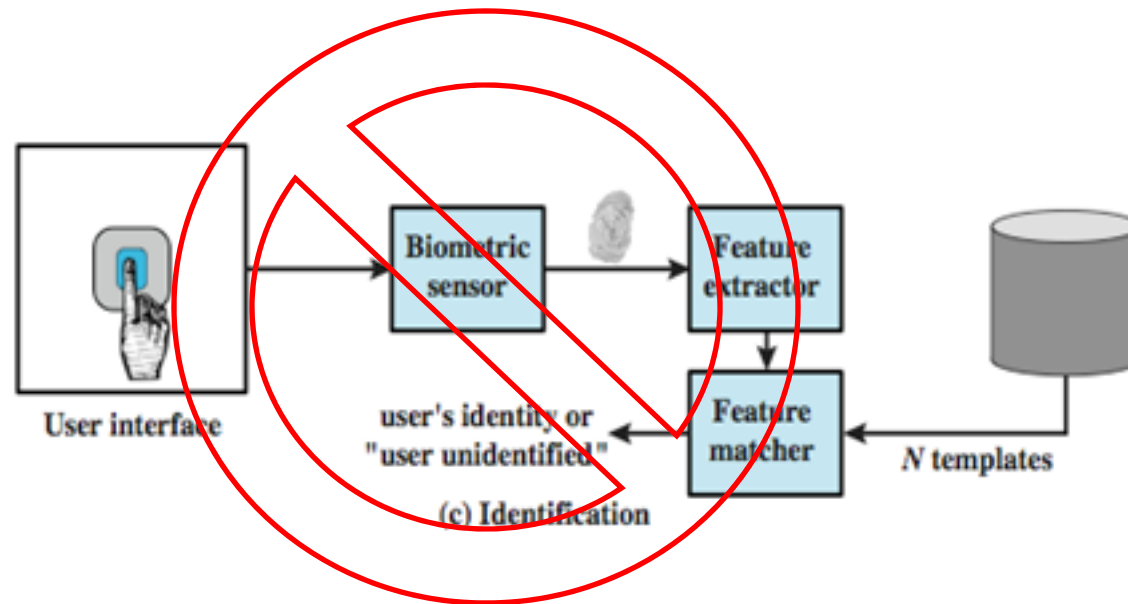


(a) Enrollment



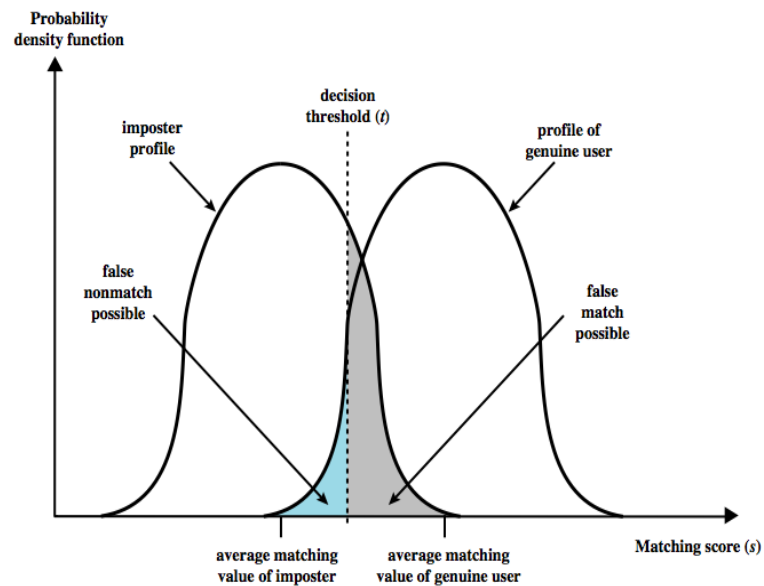
(b) Verification

# Biometric Identification



# Biometric Accuracy

- Multiple tries never get identical templates
- So, there are problems of false match / false non-match



# Assumptions

- Assumes biometric device accurate *in the environment in which it is used!*
- Assumes transmission of data to validator is both tamperproof and correct.

## Other Problems

- Acceptance of biometrics.
- Cost.
- Normal variability.
- Single point of failure.
- Recognition speed.
- Possibility of forgery. (“I, Captain Hook, present my hand to confirm my identity...”)

# Identification and Authentication

- It is not uncommon for system designers to confuse these very different concepts.
- Example: consider the Social Security Number as identifier and authenticator.
  - It's a pretty good identifier.
  - Is it a good authenticator?
  - OK... *what were they **thinking** of?*
- Biometric authentication vs. biometric identification.

## “Knowledge-Based Authentication”

- A thoroughly bogus idea!
- Premise: Only you will know things like:
  - Your former address
  - Your mother’s maiden name
  - Your Social Security Number.
- All of these can be researched.
- **Authenticators should never be researchable!**

# Questions

