MONASH University
Information Technology

FIT2093 INTRODUCTION TO CYBER SECURITY

MONASH University
Information Technology

FIT2093 INTRODUCTION TO CYBER SECURITY

Lecture 2:
Authentication

Unit Structure

- Introduction to cyber security
- Authentication
- Access Control
- Fundamental concepts of cryptography
- Symmetric encryption techniques
- Introduction to number theory
- Public key cryptography
- Integrity management
- Practical aspects of cyber security
- Hacking and countermeasures
- Database security
- IT risk management & Ethics and privacy

Previous Lecture…

- **Defined what we mean by:**
  - computer, information, (inter) network security
  - properties of information security
  - different taxonomy of attacks
  - X.800 standard security architecture
  - Security trends and strategy

LN2: Outline

- **Introduction to user authentication**
  - using passwords
  - using tokens
  - using biometrics
- **User authentication issues**
- **Password management in Unix**
- **example application and case study**

User Authentication

- **fundamental security building block**
  - basis of access control & user accountability
- **is the process of verifying an identity claimed by or for a system entity**
- **has two steps:**
  - identification - specify identifier
  - verification - bind entity (person) and identifier
- **distinct from message authentication**

Means of User Authentication

- **four means of authenticating user's identity**
- **based one something you**
  - know - e.g. password, PIN (SYK)
  - possess - e.g. key, token, smartcard (SYH)
  - are (static biometrics) - e.g. fingerprint, retina (SYA)
  - do (dynamic biometrics) - e.g. voice, sign (SYA)
- **can use alone or combined**
- **all can provide user authentication**
- **all have issues**

Password Authentication

- **widely used user authentication method**
  - user provides name/login and password
  - system compares password with that saved for specified login
- **The user ID:**
  - determines that the user is authorized to access the system
  - determines the user's privileges
  - is used in discretionary access control

Password Vulnerabilities

- offline dictionary attack
- specific account attack
- popular password attack
- password guessing against single user
- workstation hijacking
- exploiting user mistakes
- exploiting multiple password use
- electronic monitoring

Countermeasures

- Controls to prevent unauthorized access to password file
- intrusion detection measures
- Rapid reissuance of compromised passwords
- account lockout mechanisms
- policies against using common passwords but rather hard to guess passwords
- training & enforcement of password policies
- automatic workstation logout
- encrypted network links

Password – possible to improve?

- **Educating users.**
- **User account management.**
- **Password management.**
- **Challenge-and-Response**
- **Trusted Third Party**

Educating User

- Never use a portion or variation of your account name or another account name.
- Never use a portion or variation of your real name, office or home address, or phone number.
- Never use words or variations of words found in any dictionary, especially /usr/dict words.
- Never use pairings of short words found in any dictionary (*dogcat*)
- Never use dictionary words or names spelled backwards (like *terces*)
- Never use syllables or words from a foreign language.
- Never use repeated character string (like AAAABBBB, LLOOVVEE)
- Never use passwords containing only numeric digits.
- Always use passwords at least seven characters long (how many words with 7 characters are possible?). Many Unix versions use only 8 characters while some new ones may permit 16 or more characters.
- Always use a mixture of upper- and lowercase characters. This is especially valuable rule.
- Always use at least one or two non-alphanumeric characters, like numeric digits, punctuation marks, dollar sign, control characters, caret, etc.

All these leads to a large number of combinations which may take the cracker program long enough to crack.

A quick counting

- **How many 2 digit PIN numbers are possible?**
- **Repeat above with digits not repeating within a PIN.**
- **With no repeating digits as well as mirror pins are the same (such as 45 is same as 54, etc.)**

User Account Management

- **Never create an account without password.**
- **Keep an eye on inactive accounts.**

Password Management

- Not allowing a password that is similar to the user name, derivative (copied) or the 'known' words in the system dictionary.
- Make sure that the rules in [illegible]s" are applied.
- Do not store the passwo[illegible]d in plain text form, rather store them scrambled, even better store a derivative of the password.
- Maintaining the shadow password.
- Implementing password a[illegible]

Use Encryption

Use Hashing

Specific to Unix

Will talk about this more later

Storing Password in a system

- **Two different strategies are used for storing the passwords in a system.**
  - Transform the password into something else (windows)
  - Use password itself to transform something else (Unix)
- **By the way where are the user passwords stored in Windows and Unix systems?**

Derivative of a Password

- **Use a one way hash function.**
- **What is a hash function?**
  - A function that can be used to turn data into relatively smaller format that may serve as digital representation of the data; however you will not be able to derive the data from its derivative
  - The derivative is known as **message digest**
- **Example:**
  - Take every third letter in a word
  - Srinivasan => iva

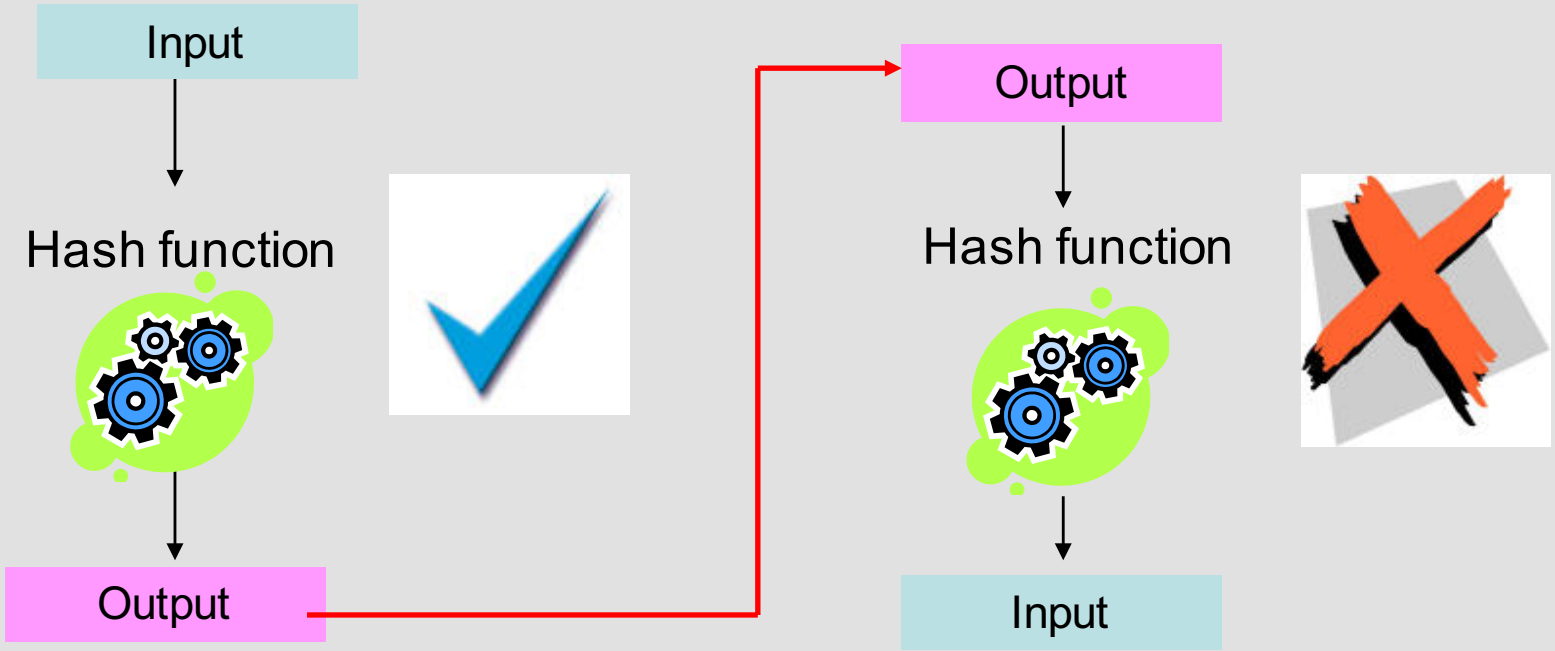
Typically smaller in size (length)

Also known as Hash value

Hash Algorithm

Cover up, Confidential

- **Provides a good way to disguise a password (when storing in a system) because it is a one way function.**



Hash Function - Collision

Collision

- **It is possible that two inputs produce the same output.**
- **Example:**
  - Take every third letter as a function
  - "parking" and "barking" => "rn"
- **A good hashing algorithm will produce small (almost negligible) number of collisions.**
- **Increasing the length of the hash code can reduce the probability of collision.**

Collision here → two words (messages) produce the same hash/message digest value

Secure Hash Function

- **For a given algorithm, it is computationally infeasible:**
  - to find a message that corresponds to a given message digest (one-way hash)
  - to find two different messages that produce the same message digest (**collision resistance**).
  - Any change to a message will, with a very high probability, result in a different message digest

Federal Information Processing Standard Publication (FIPS) No 180-2

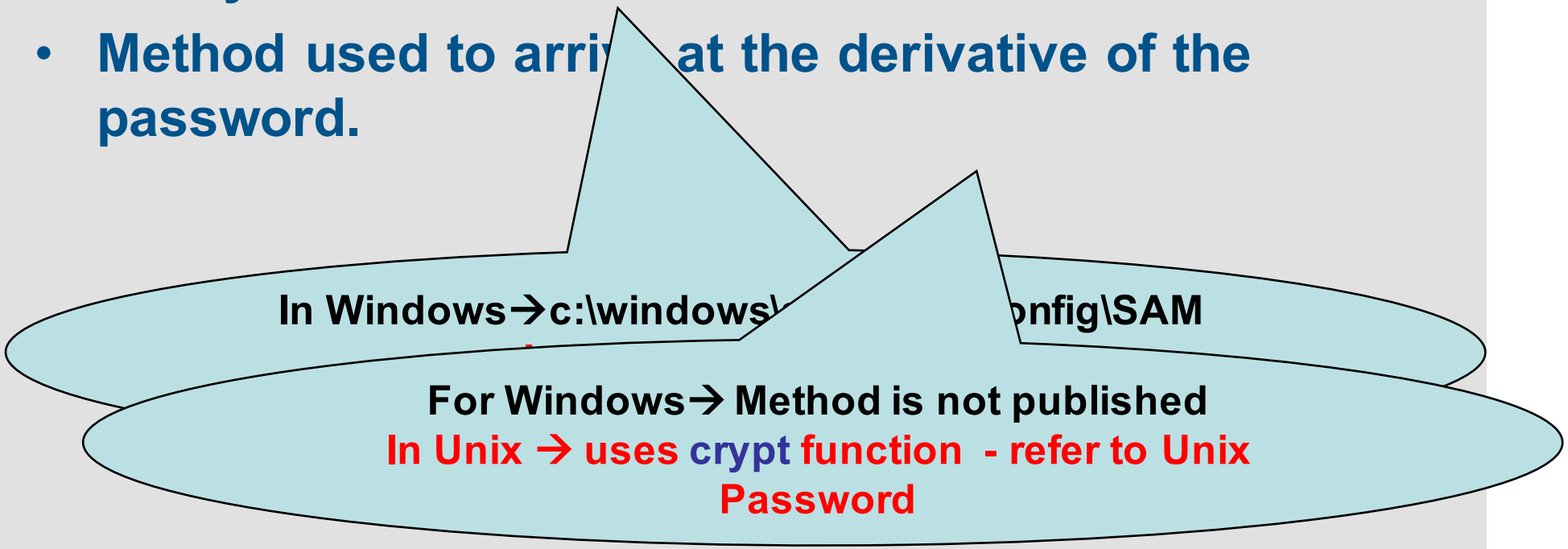
Crackers – Who are they?

- **People who want to steal (find) other legal users' passwords.**
- **How do they do this?**
  - Brute-force or exhaustive search
  - Remember for a given length of static passwords, the number of passwords is countable (finite)

How many 4 digit PIN or 8 character long passwords are possible?

Crackers – Need to have

- **Where is the password or its derivative stored in the system?**
- **Method used to arri at the derivative of the password.**



MONASH University
Information Technology

Other ways to make life difficult for crackers are:

1. Shadow Password – why do we need it?

Specific to Unix

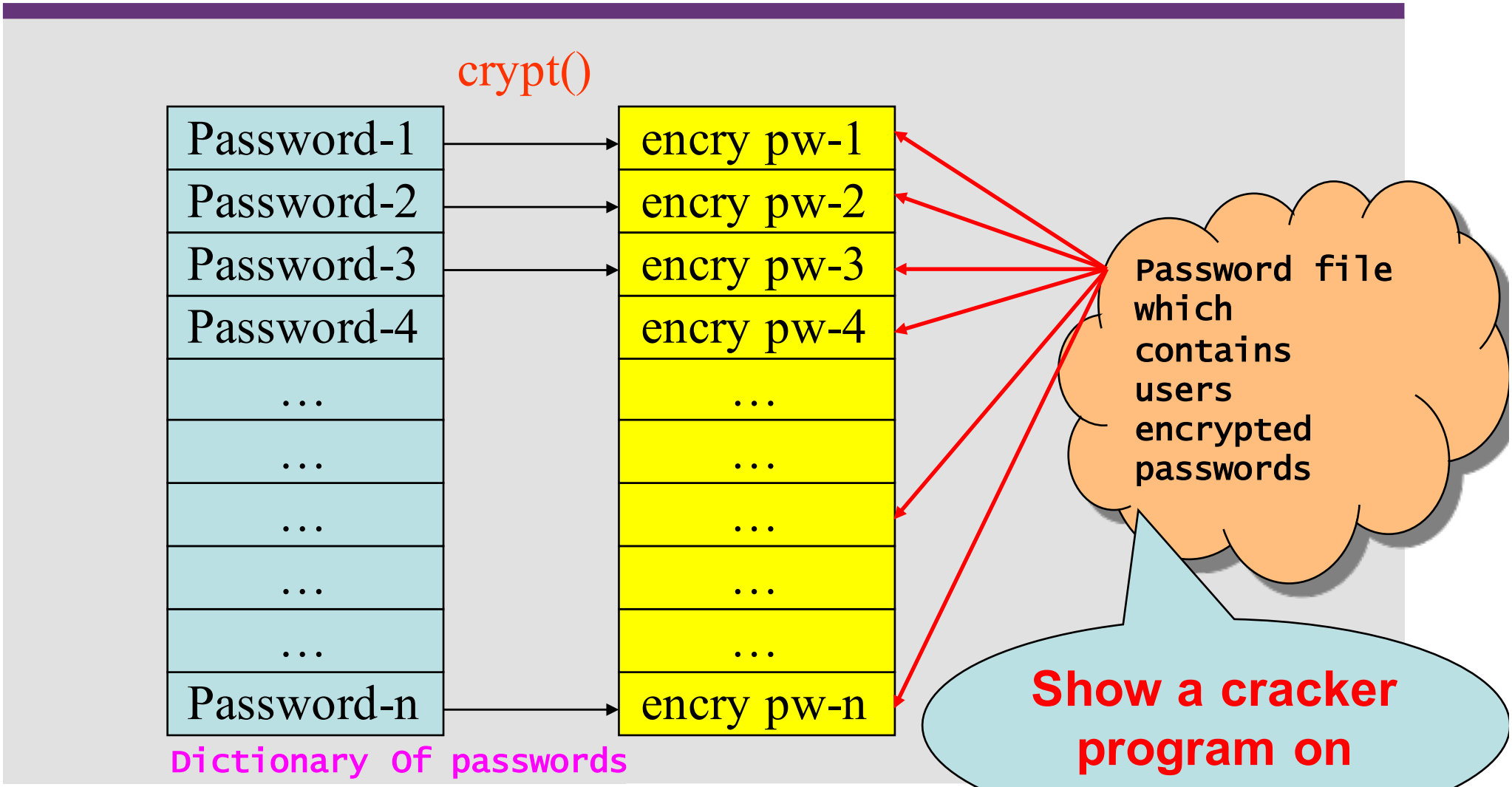
- **System without a shadow password keeps the information about the user details in a file that can be accessed by the normal user in the system.**
- **Normal users in the system may need to access the password file because it may contain information about the users such as full name.**
- **We can encrypt the password and keep the encrypted password in the password file, however, it is still possible to attack.**
  - A cracker can encrypt possible passwords and compare the result of encryption with the encrypted password in the password file.
- **To prevent the attack but allowing normal users to access the user information:**
  - Remove the encrypted password replace it with an "x" or "*" in the password file and place the encrypted password in a file accessible only by the superuser (shadow password file).

Access control to a file

Password Crackers – dictionary attack

- **Password Crackers are known to use dictionary attack**
  - create a dictionary of possible passwords
  - generate the corresponding derivative passwords for the words in the above dictionary (assuming no salt)
  - Match the entries in the generated passwords against the actual passwords on the system for a possible match.
  - All the above operations can be done off-line!!

Dictionary Attack – How it works?



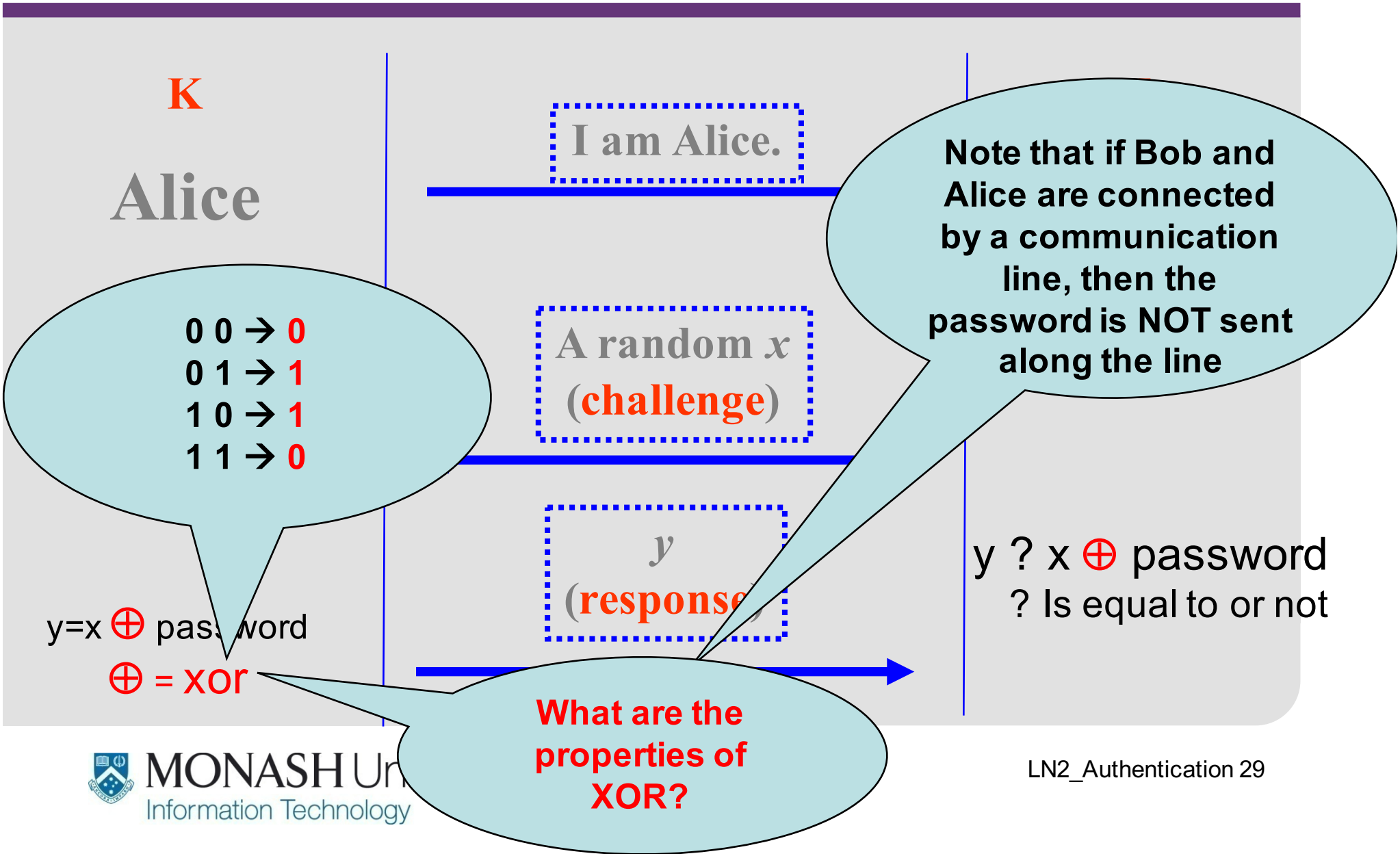
How many passwords are possible?

- **Let us assume two characters a and b and password of length 2, then possible passwords are**
- **aa, ab, ba, bb → $4 = 2^2$**
- **If length is 3 then no of passwords =**
- **aaa, aab, aba, abb, baa,bab, bba, bbb = $8 = 2^3$**

2. Password Aging

- **Forces the users to change their password regularly.**
  - Prompt the user with change password screen during the logging process when the “age” of the password has reached the maximum allowable limit.
  - If someone else knows the password, the damage is only for that period of time before it is changed.
- **Additional security measure to password aging.**
  - Maintain password history and prevent re-use of earlier passwords.

3. Challenge-and-Response protocol

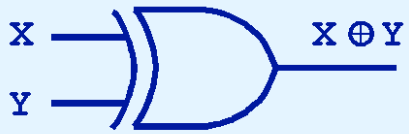


XOR

- **The output of the XOR operation is True (1) only when the values of the inputs differ**

X XOR Y

X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0



Associative law

$$(x \oplus y) \oplus z = x \oplus (y \oplus z)$$

$$x \oplus x = 0$$

$$x \oplus 0 = x$$

$$x \oplus 1 = \bar{x}$$

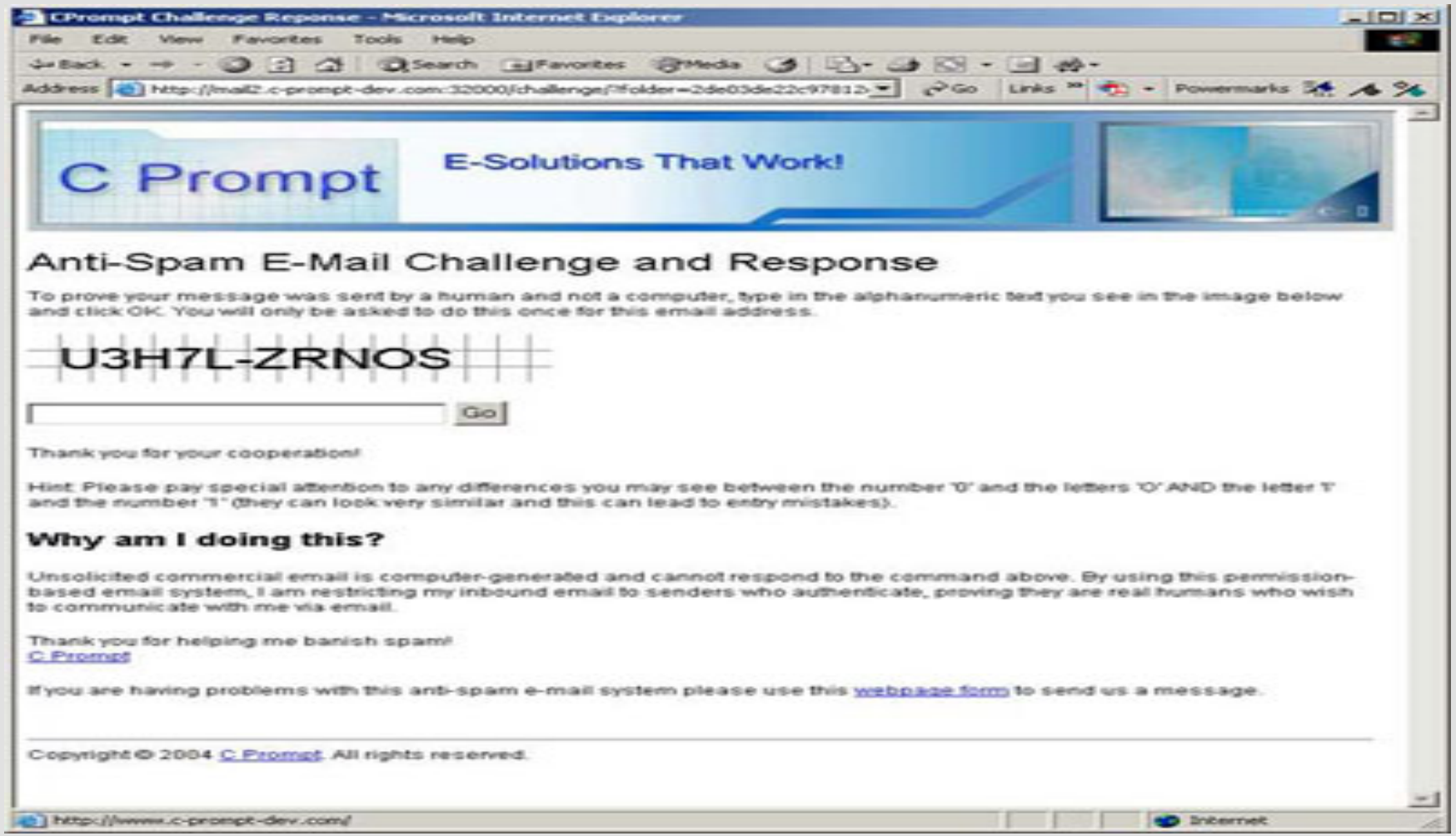
Properties of XOR

Challenge-and-Response protocol

- **Both sides need to know the function**
- **Problem with XOR (⊕)**
  - if you can intercept **x** and **y**, then you can get the password by **x ⊕ y**
  - **XOR is a reversible function, efficient operator to implement**
  - **If p is the password, x is the challenge and y = p ⊕ x, then x = p ⊕ y**
  - **What are the values of p ⊕ ¬X, ¬ p ⊕ X, p ⊕ ¬ y and ¬ p ⊕ y ?**
  - **Instead use one way hash fu[illegible]on where y = h(x)**

Home work!

Challenge and Response - Example



One Time Password

- **Challenge will vary every time and hence each challenge is used only once**
  - Short time or One-off password
- **Other examples:**
  - SMS, token for internet banking

4. Trusted Third Party for Authentication

- **It facilitates two 'strangers' to 'trust' each other and exchange information.**
  - Trust => you're who you claim to be
- **Real life example**
  - VicRoads is a trusted third party because it produces a driver's license that many of us use and accept it as a means to authenticate an individual.
- **Digital world**
  - A trusted party can produce a 'digital certificate' that can be used by an individual to prove his/her identity to another interested party with whom he wants to communicate.
  - Example: Kerberos

Something you have (SYH)

- **House key, (unlocked) mobile phone**
- **Mere possession of this enables valid authentication.**

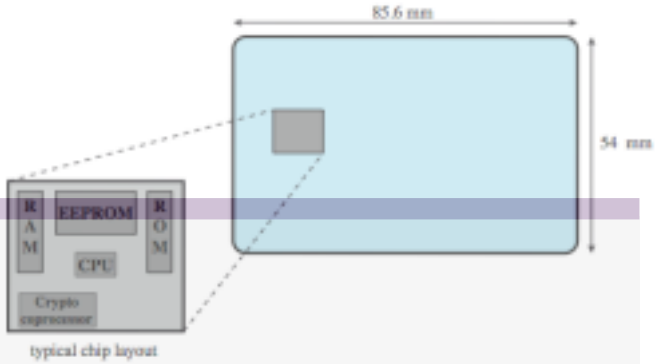
Token Authentication

- **object user possesses to authenticate, e.g.**
  - embossed card
  - magnetic stripe card
  - memory card
  - smartcard

Memory Card

- **store but do 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

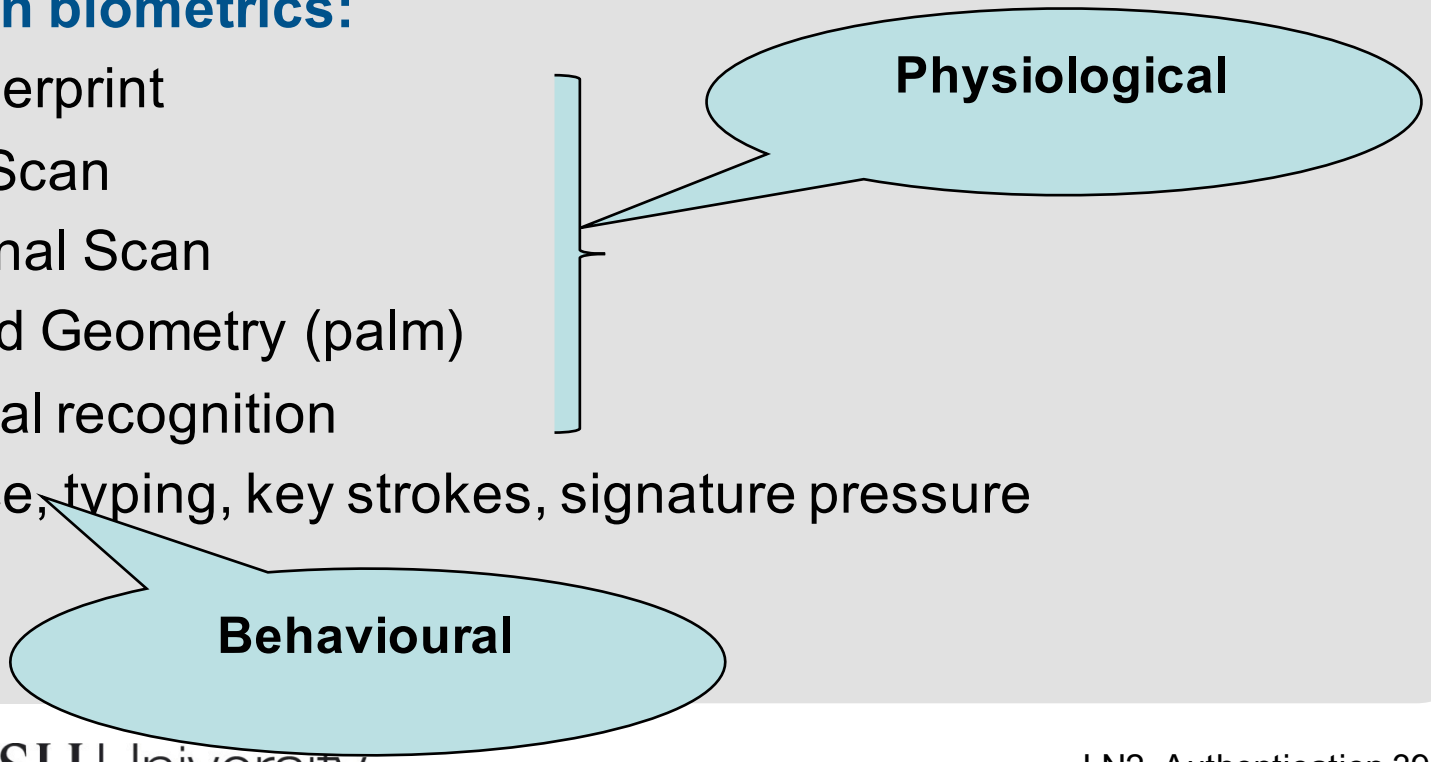
Smartcard



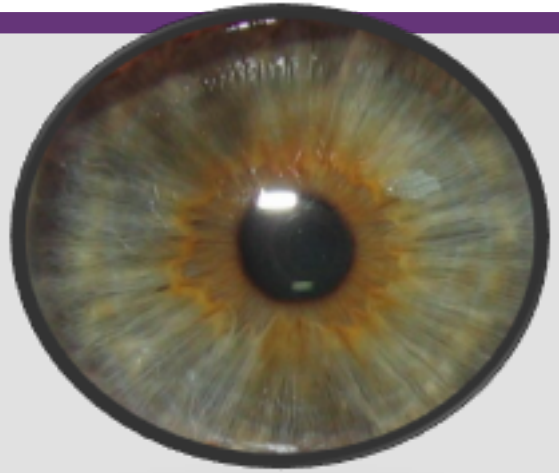
- **credit-card like**
- **has own processor, memory, I/O ports**
  - wired or wireless access by reader
  - may have crypto co-processor
  - ROM, EEPROM, RAM memory
- **executes protocol to authenticate with reader/computer**
- **also have USB dongles**

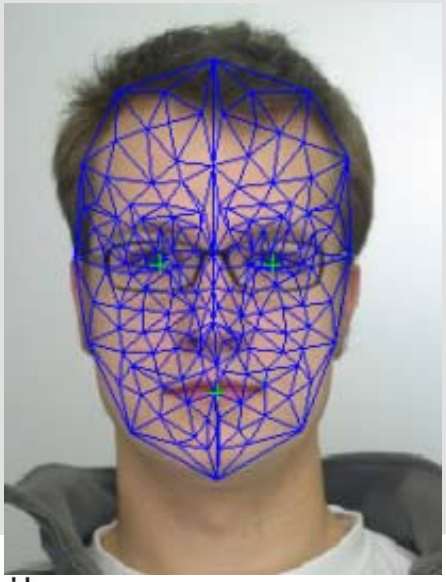
Something You Are (SYA)

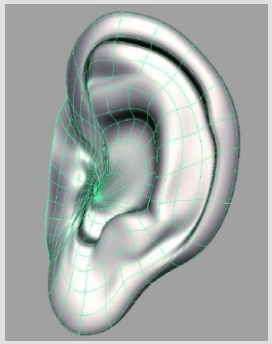
- **Use of human Biometric(s)**
- **A characteristic of the body**
- **Presumed to be unique and invariant over time.**
- **Common biometrics:**
  - Fingerprint
  - Iris Scan
  - Retinal Scan
  - Hand Geometry (palm)
  - Facial recognition
  - Voice, typing, key strokes, signature pressure

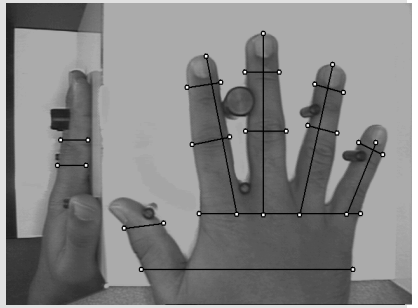


Iris recognition & Retinal Scan





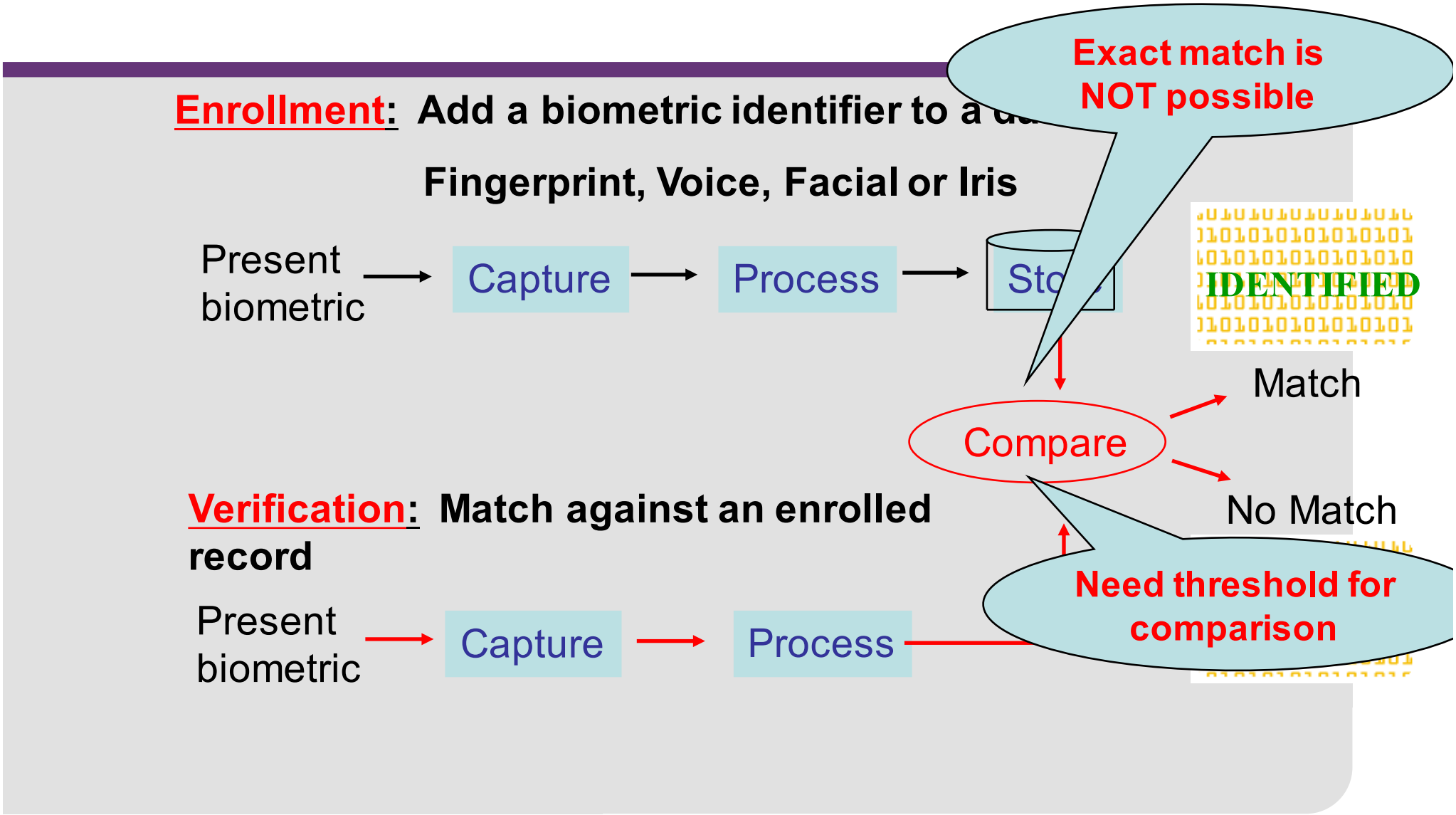




Stable vs Alterable Biometric

- **Stable biometric**
  - Does not change from one session to another.
  - Relatively static
  - Fingerprint, Face, Hand, Iris, Retina
- **Alterable biometric**
  - Can be altered easily if necessary
  - Voice, Keystrokes pattern

How Biometrics Authentication Works?



Face Recognition(1)

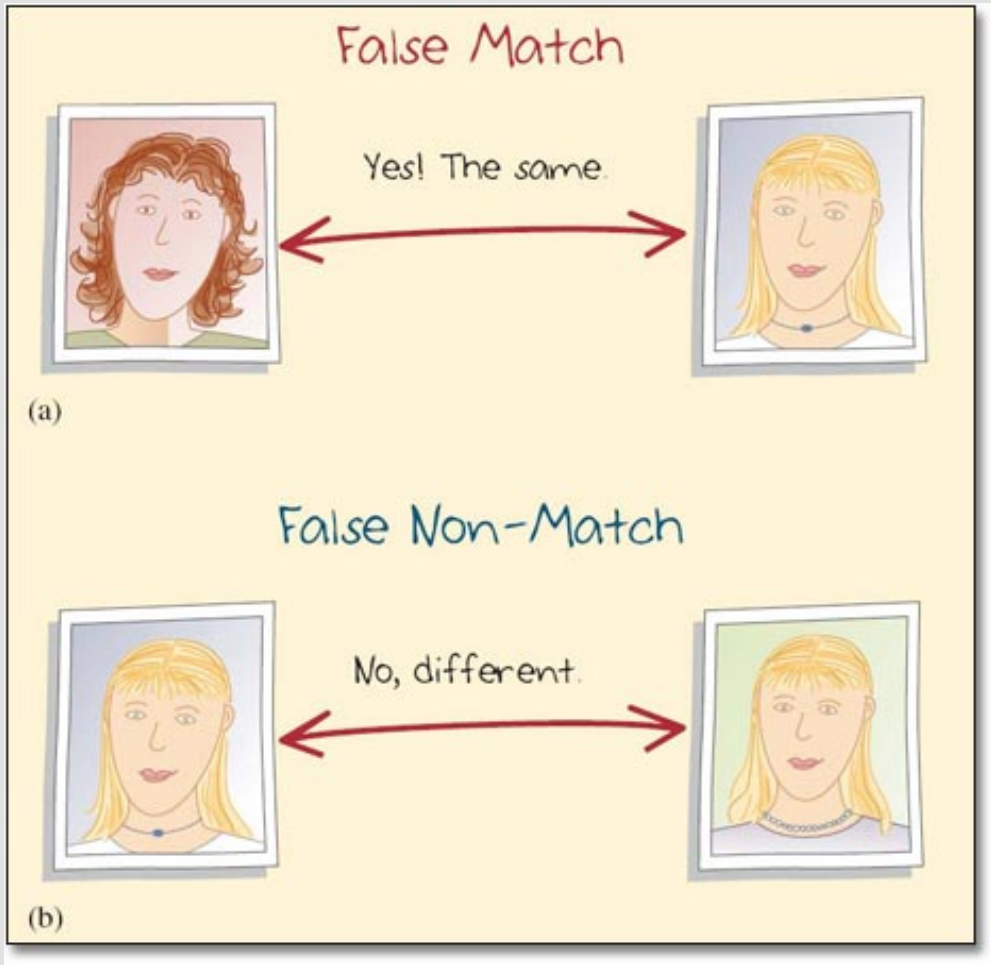
- **The oldest method of biometric!**
- **There is widespread acceptance (and requirement !) for photo ID**
- **The issuing of *other* authentication devices (like passwords, key cards, digital signatures) usually depends on facial recognition by the agents of the issuing authority.**
- **Photo-ID may not be reliable, but has a very significant deterrent (restriction) effect.**

Matching vs. Non-Matching

These two types of error are known as false match and false non-match.

A false match is when two pieces of biometric data from different people are judged to be from the same person, as in (a). This type of error is sometimes called a false accept (FALSE POSITIVE).

A false non-match is when two pieces of biometric data from the same person are judged to be from different people, as in (b). This type of error is sometimes called a false reject (FALSE NEGATIVE).



Matching vs. Non-Matching templates

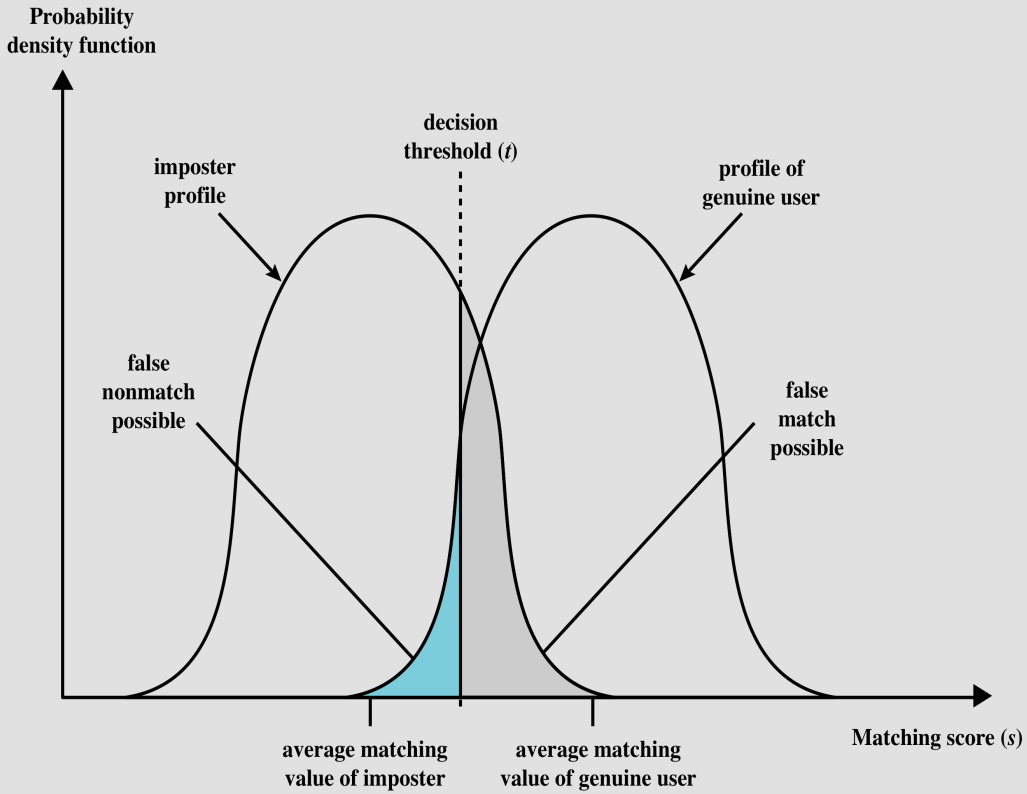
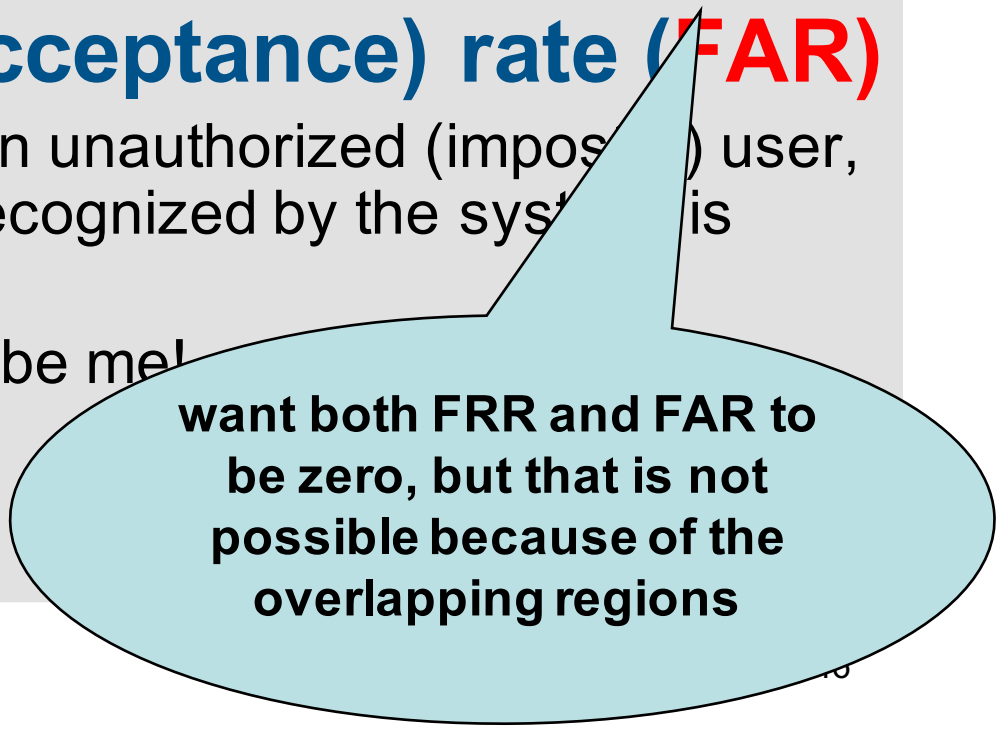


Figure 3.8 Profiles of a Biometric Characteristic of an Imposter and an Authorized Users In this depiction, the comparison between presented feature and a reference feature is reduced to a single numeric value. If the input value (s) is greater than a preassigned threshold (t), a match is declared.

How Accuracy is Measured?

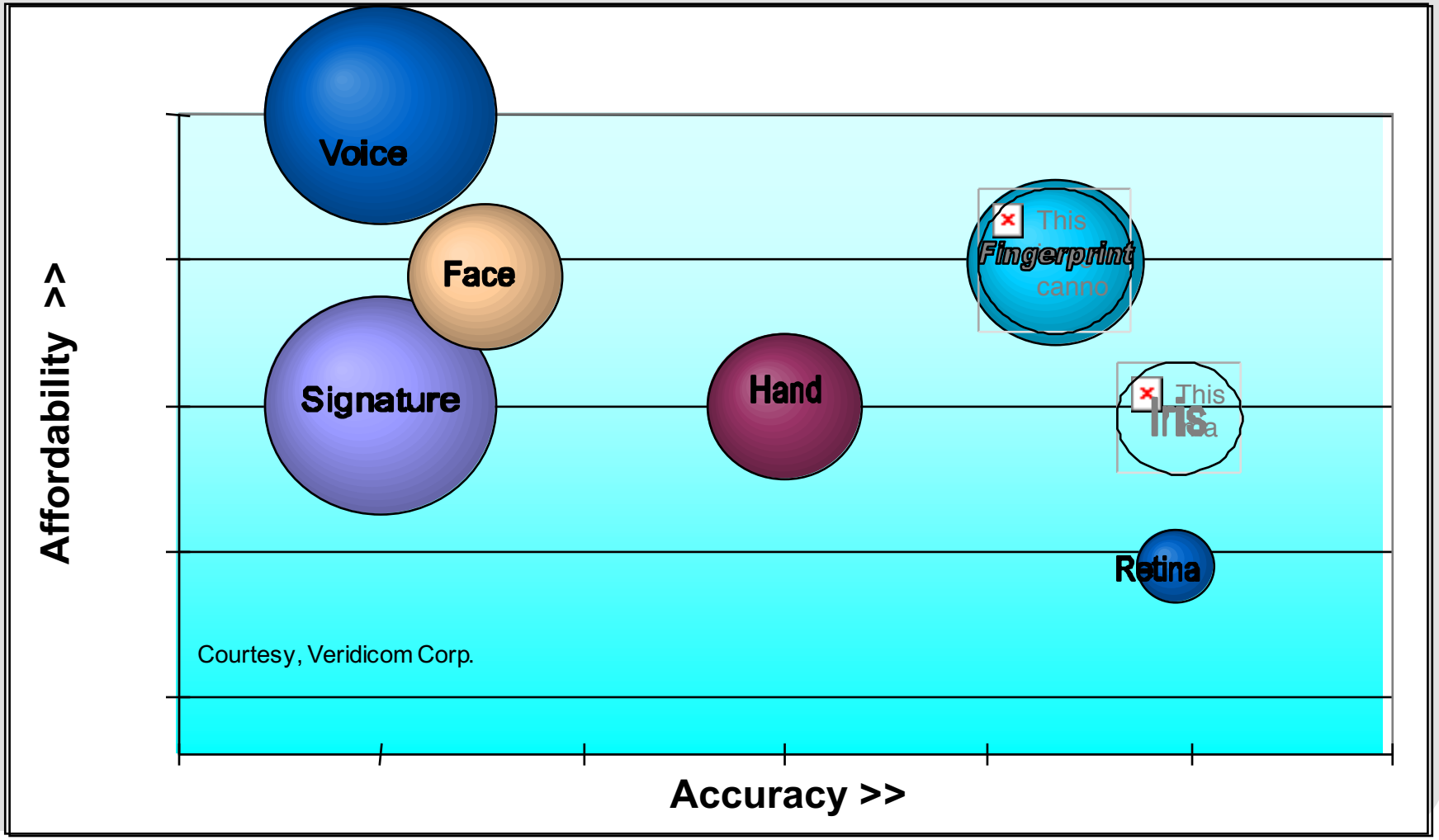
- **False Negative (rejection) rate (FRR)**
  - Measures how often an authorized (genuine) user, who should BE recognized by the system, is not recognized.
  - I am not recognised as me!
- **False Positive (acceptance) rate (FAR)**
  - Measures how often an unauthorized (impos) user, who should NOT be recognized by the sys is falsely recognized.
  - You are pretending to be me!



Technology Comparison

	Iris	Face	Finger	Signature	Voice
Accuracy	Very High	Medium	High	High	Medium
Ease of Use	Medium	Medium	High	High	High
Barrier to Attack	Very High	Medium	High	Medium	Medium
User Acceptability	Medium	Medium	Medium	Very High	High
Long Term Stability	High	Medium	High	Medium	Medium
Interference	Coloured Contacts	Lighting Aging, Glasses, Hair	Dryness Dirt,	Changing Signatures	Noise, Colds,

Accuracy v. Affordability v. Acceptability



What happens if “biometric” info is stolen?

- **Unlike static password that can be changed easily when a malicious activity is detected, it is not possible to change stable biometric information.**
- **It is possible to change the alterable bio**
  - e.g. change the phrases spoken for vo
- **Some biometric information is easi duplic forge compared to others fingerprint is easier to obtain compared t iris.**
- **Multimodal can be used to overcome the issue.**
  - For example, use the combination of fingerprint and iris.
- **“Behavior traits (genetic)” as biometric?**

Two or more biometrics

Summary

		User Authentication		
		SYK	SYH	SYA
Security terminology		Password, secret	Token	Biometric
Support Authentication by		Secrecy or obscurity	Possession	Uniqueness, personalisation
Defense mechanism		Closely kept	Closely held	Forge-resistance
Example	Traditional	Combination Lock	Metal Key	Driver's license
	Digital	Computer Password	ATM Card	Fingerprint
Drawback		Less secret with each use	Insecure if lost	Difficult to replace

MONASH University
Information Technology

Password Management in UNIX

Password Files

- **/etc/passwd**

  -rw-r--r-- 1 root root 9434 Mar 14 16:13 passwd

  Sample entry:

  maria:x:510:510:Maria Brody:/home/maria:/bin/bash

  On a system without a shadow password, the hashed password is kept in the */etc/passwd*

- **/etc/shadow**

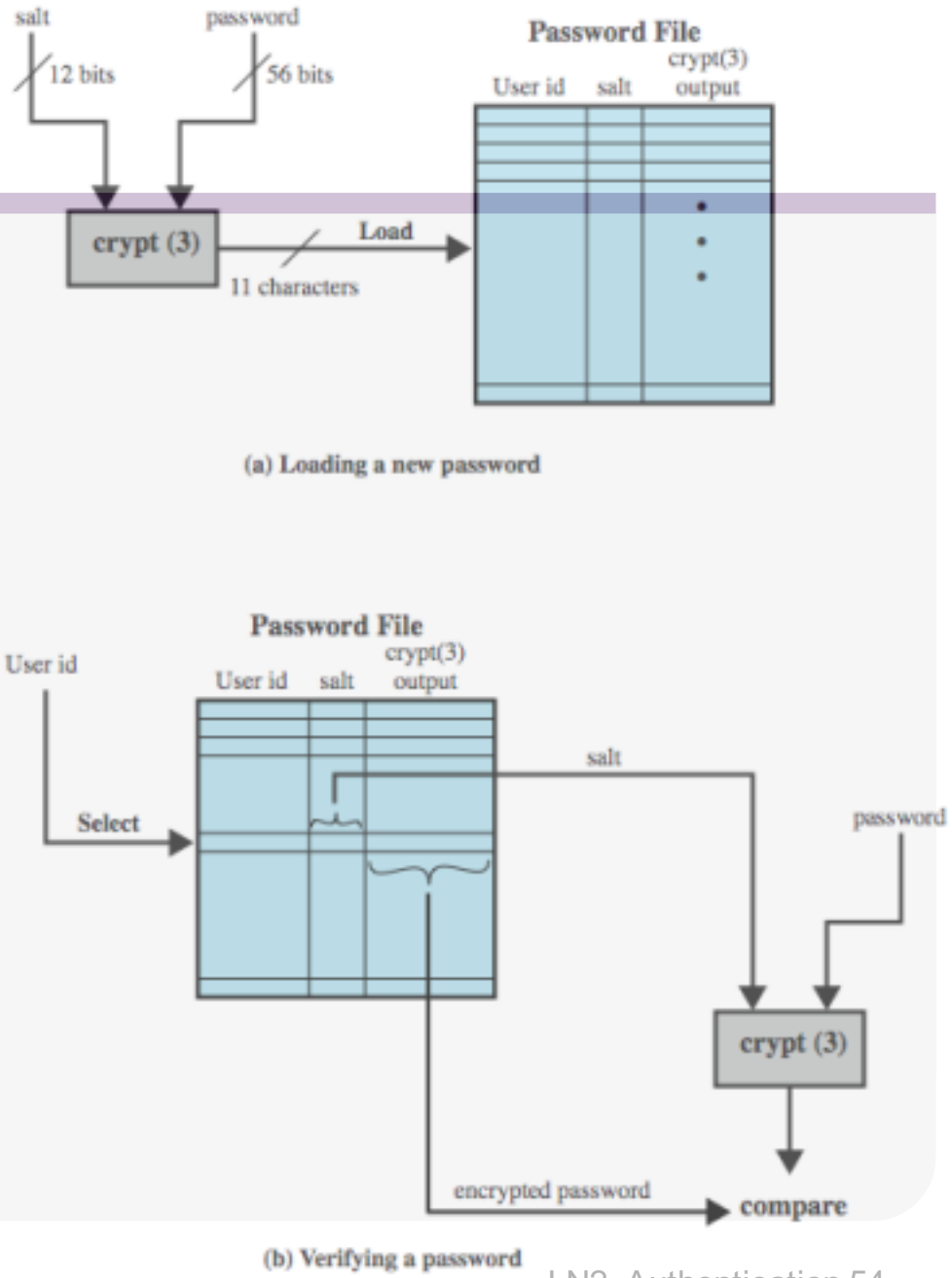
  -r-------- 1 root root 9225 Mar 19 13:49 shadow

- **Typical standard UNIX password is 8 characters length (64 bits).**

Password Cracking

- **If the system does not have shadow password, the /etc/passwd can be copied and the attack can be done offline.**
- **Main technique is to generate hashed/encrypted version of potential password and compare it with the hashed/encrypted password entry in /etc/passwd.**
- **Methods of attack:**
  - Brute Force
    - > Try all possible combinations of alphanumeric/digits/symbols.
  - Dictionary Attack

Use of Hashed Passwords



UNIX Implementation

- **original scheme**
  - 8 character password form 56-bit key
  - 12-bit salt used to modify DES encryption into a one-way hash function
  - 0 value repeatedly encrypted 25 times
  - output translated to 11 character sequence
- **now regarded as woefully insecure**
  - e.g. supercomputer, 50 million tests, 80 min
- **sometimes still used for compatibility**

Salt

- Adding randomness to the encrypted password.
- For example, time of the day can be used as a salt.
- The salt is converted into a two-character string (12 bits in fact) and is stored in the */etc/passwd* file along with the encrypted "password"
- The encryption of the string of 64 bits of zeros is done by the string which is the concatenation of the salt and the user supplied password string!
- Having salt means that the same password can encrypt in 4096 different ways! and this makes it much harder for the attacker to build a reverse dictionary for translated encrypted passwords.

Using Better Passwords

- **clearly have problems with passwords**
- **goal to eliminate guessable passwords**
- **whilst still easy for user to remember**
- **techniques:**
  - user education
  - computer-generated passwords
  - reactive password checking
  - proactive password checking

Remote User Authentication

- **authentication over network, the Internet or a communications link is more complex**
  - problems of eavesdropping, capturing a password, replaying an authentication sequence
- **generally use challenge-response**
  - user sends identity
  - host responds with random number
  - user computes f(r,h(P)) and sends back
  - host compares value from user with own computed value, if match user authenticated
- **protects against a number of attacks**

Password Protocol

Client	Transmission	Host
U, user	$U \rightarrow$	
	$\leftarrow \{r, h(), f()\}$	random number h(), f(), functions
P' password r', return of r	$f(r', h(P')) \rightarrow$	
	$\leftarrow$ yes/no	if $f(r', h(P')) = f(r, h(P(U)))$ then yes else no

Example of a challenge-response protocol

- user transmits identity to remote host
- host generates a random number (nonce)
- nonce is returned to the user
- host stores a hash code of the password
- function in which the password hash is one of the arguments
- use of a random number helps defend against an adversary capturing the user's transmission

Token Protocol

- user transmits identity to the remote host
- host returns a random number and identifiers
- token either stores a static passcode or generates a one-time random passcode
- user activates passcode by entering a password
- password is shared between the user and token and does not involve the remote host

Client	Transmission	Host
U, user	$U \rightarrow$	
	$\leftarrow \{ r, \text{h}(), \text{f}() \}$	r, random number h(), f(), functions
$P' \rightarrow W'$ password to passcode via token r', return of r	$\text{f}(r', \text{h}(W')) \rightarrow$	
	$\leftarrow$ yes/no	if $\text{f}(r', \text{h}(W')) = \text{f}(r, \text{h}(W(U)))$ then yes else no

(b) Protocol for a token

Example of a token protocol

Biometric Protocol

Client	Transmission	Host
U, user	$U \rightarrow$	
	$\leftarrow \{ r, \mathrm{E}() \}$	r, random number E(), function
$B' \rightarrow BT'$ biometric D' biometric device r', return of r	$\mathrm{E}(r', D', BT') \rightarrow$	$\mathrm{E}^{-1}\mathrm{E}(r', P', BT') =$ (r', P', BT')
	$\leftarrow$ yes/no	if $r' = r$ and $D' = D$ and $BT' = BT(U)$ then yes else no

(c) Protocol for static biometric

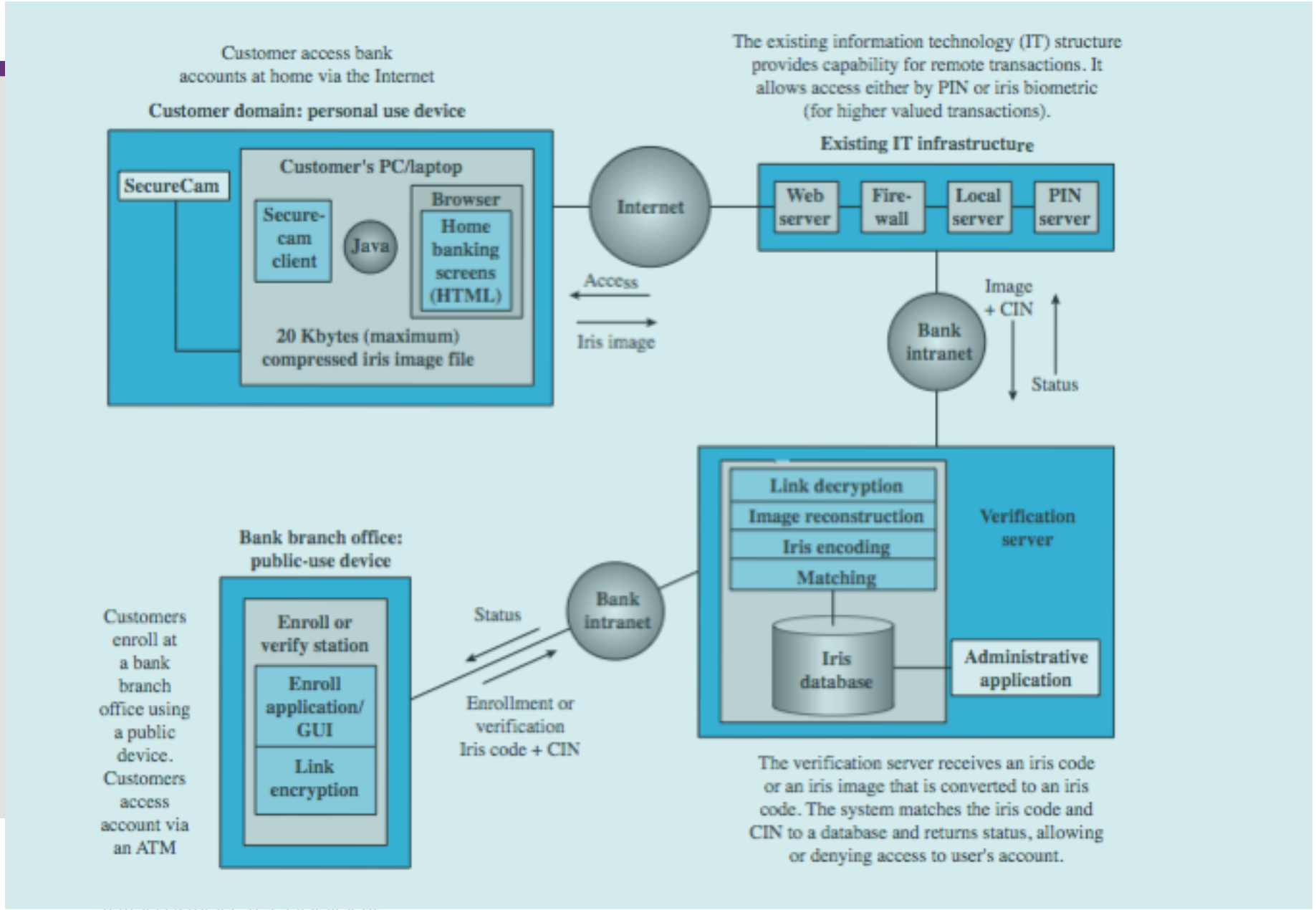
Example of a static biometric protocol

- user transmits an ID to the host
- host responds with a random number and the identifier for an encryption
- client system controls biometric device on user side
- host decrypts incoming message and compares these to locally stored values
- host provides authentication by comparing the incoming device ID to a list of registered devices at the host database

Authentication Security Issues

- **client attacks**
  - adversary attempts to achieve user authentication without access to the remote host
- **host attacks**
  - directed at the user file at the host where passwords, token passcodes, or biometric templates are stored
- **Eavesdropping**
  - attempts to learn the password by physical proximity to user
- **Replay**
  - adversary repeats a previously captured user response
- **trojan horse**
  - an application or physical device masquerades as an authentic application
- **denial-of-service**
  - attempts to disable a user authentication service by flooding the service with numerous authentication attempts

Practical Application: Iris Biometric System



Summary

- **introduced user authentication**
  - using passwords
  - using tokens
  - using biometrics
- **examined user authentication issues**
- **How password is managed in unix**
- **example application and case study**

Further Reading

- **Chapter 3 of the textbook: *Computer Security: Principles and Practice" by William Stallings & Lawrie Brown,* Prentice Hall, 2015**
- **Chapter 8 of the book: "*Practical Unix & Internet Security"* by Simon Garfunkel and Gene Spafford, Edition 2, 2003.**

- **Acknowledgement: part of the materials presented in the slides was developed with the help of Instructor's Manual and other resources made available by the author of the textbook.**