

## User authentication

*Information Security (bmevihim102)*

Dr. Levente Buttyán  
associate professor

BME Híradástechnikai Tanszék  
Lab of Cryptography and System Security (CrySys)  
buttyan@hit.bme.hu, buttyan@crysys.hu



## Outline

- introduction
- password based authentication and password cracking
- authentication with HW tokens
- biometric identification
- CAPTCHAs
- summary



## Introduction

- authentication = process of proving a claimed identity
- basic security function; usually a prerequisite of making access control decisions
  - e.g., login to a computer, connecting to a network, etc.
- three basic approaches
  - passwords and alike (what you know)
  - hardware tokens (what you have)
  - biometrics (what you are)



## Objectives of this lecture

- understand the basic operating principles of
  - password based authentication schemes
  - hardware token (smart card) based authentication
  - biometric authentication
- get a deeper insight into password cracking methods
- get a deeper insight into the field of fingerprint verification
- CAPTCHAs
  - understand the difference between user authentication mechanisms and CAPTCHAs
  - understand how CAPTCHAs work and how they can be defeated

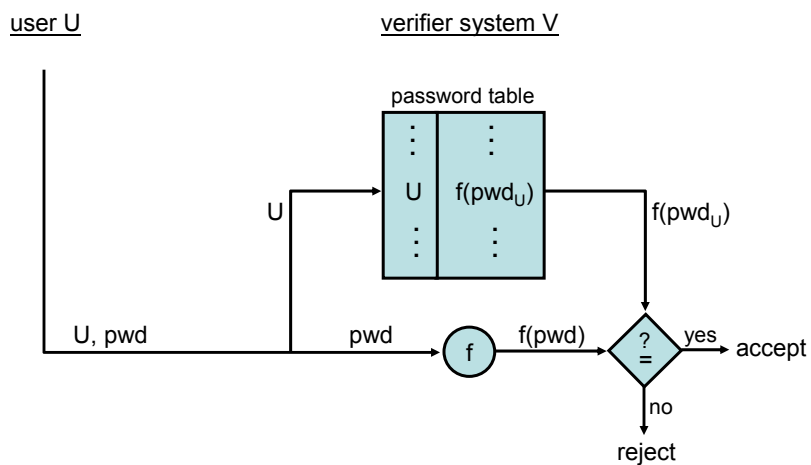


## Lecture outline

- introduction
- password based authentication and password cracking
- authentication with HW tokens
- biometric identification
- CAPTCHAs
- summary

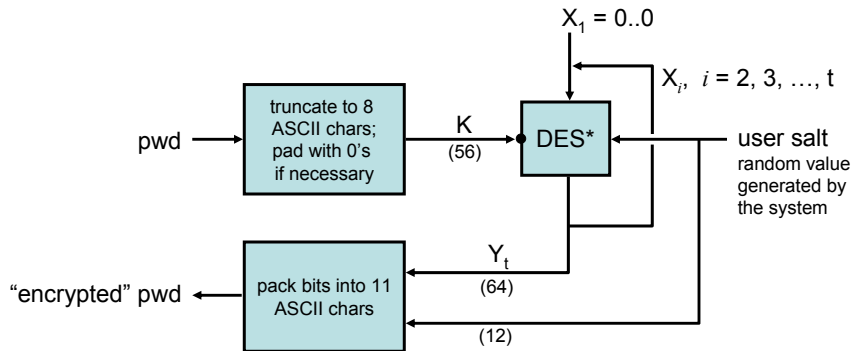


## General model





## Example – Unix passwords

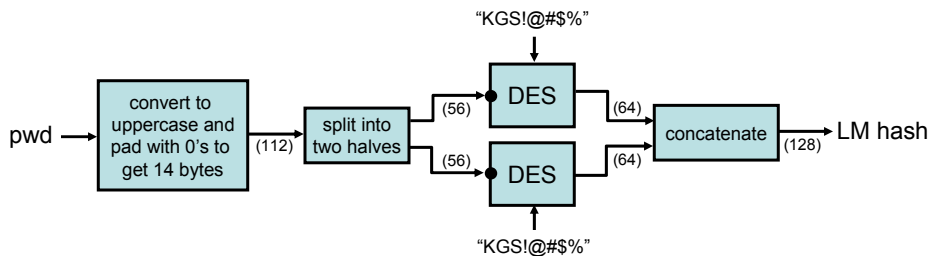


- design principles:
  - multiple iterations make exhaustive search slower
  - user salt makes pre-computation attacks impractical
  - modified DES prevents the use of off-the-shelf DES chips



## Example – Windows passwords

- Windows stores user passwords in two formats
  - NT hash = MD4 hash of the password
  - LM hash (LAN Manager hash)
    - still supported for backward compatibility
    - from Vista, this is switched off by default, but can be enabled



- weaknesses:
  - two halves can be cracked separately ( $95^7 \sim 2^{46} \ll 95^{14} \sim 2^{92}$ )
  - conversion to uppercase reduces the size of the pwd space ( $69^7 \sim 2^{43}$ )



## Pros and cons of passwords

- advantages
  - simple and intuitive → easy to understand by average users
  - cheap to implement
- disadvantages
  - password must be memorized by the user
    - users tend to choose guessable passwords
    - users tend to use the same password on multiple systems
  - passwords or password hashes must be stored by the verifier
    - password files can be stolen and analyzed off-line
    - brute force and dictionary attacks are possible
  - password can be obtained on its way from the user to the verifier
    - key stroke logging, shoulder surfing
    - eavesdropping (→ encrypted transport between remote parties is essential)
    - replay attacks
  - passwords are easy to reveal and share
    - social engineering attacks are possible



## Attacks

- principle: attacker tries a list of candidate passwords one after the other
  - guessing: tries commonly used passwords and uses knowledge about the victim (obtained through social engineering or social networks)
    - blank, “admin”, “qwerty”, “1234”, ...
    - user name, children, pet, date of birth, ...
    - variations and combinations such as reversing a word or appending digits
    - unchanged default passwords (list of default passwords is available on the Web)
  - dictionary attack: tries words from a dictionary (and their variations)
    - around 40% of user-chosen passwords are readily guessable by sophisticated cracking programs armed with dictionaries and the user's personal information
    - password cracking programs are also available on the Web
  - brute force: tries all possible inputs
- carefully chosen passwords can prevent guessing and dictionary attacks
- brute force attacks can be prevented by choosing a large password space (allowing long passwords)
- efficiency of brute force attacks can be increased by pre-computations (time-memory trade-off)



## Time-memory trade-off

- task: count the number of 1's in a 32-bit integer  $x$
- solution 1:

```
t = 0
for i = 0 to 31
    t = t + (x >> i) & 1
next i
```

- 32 iterations (operations), 1 unit of memory (to store  $t$ )

- solution 2:

```
// pre-compute and store weight[y]
// for each 8-bit y
t = 0
for i = 0 to 4
    t = t + weight[(x >> (i*8)) & 0xFF]
next i
```

- 4 iterations, 256+1 units of memory (to store the array  $weight$  and  $t$ )
- + one-time effort needed for the pre-computation, but that is amortized if the task is needed to be executed many times

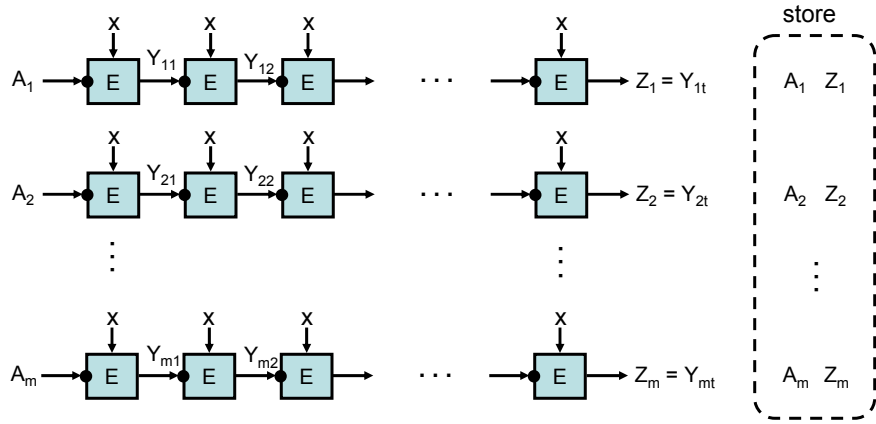


## Application in cryptography

- speeding up brute force attack against block ciphers
- problem:
  - assume a chosen plaintext attack model
    - attacker chooses  $X$  and receives  $Y = E_K(X)$
    - he wants to find the key  $K$
  - naïve exhaustive key search  $\rightarrow \sim 2^{k-1}$  computation, no memory
  - pre-compute and store  $E_K(X)$  for all  $K \rightarrow 1$  table look-up,  $2^k$  memory
  - is there anything in between?

# A crypto time-memory trade-off

- pre-computation

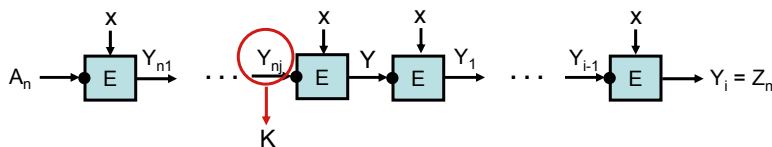


simplifying assumption:  $mt \sim 2^k$  and  $\{Y_{ij}\}$  covers large part of the entire key space

# A crypto time-memory trade-off

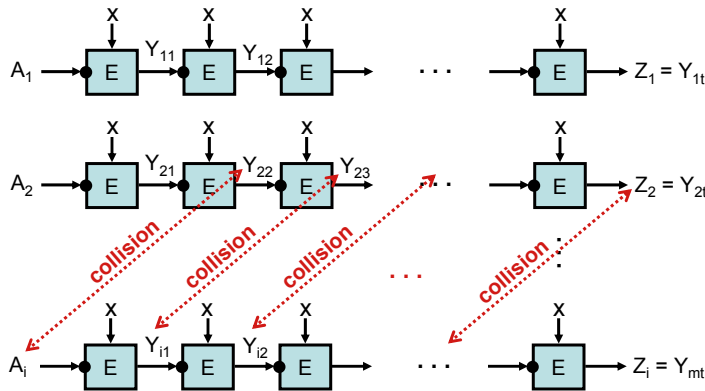
- attack

- obtain  $Y = E_K(X)$
- compute  $Y_1 = E_Y(X)$ ,  $Y_2 = E_{Y_1}(X)$ , ... until  $Y_i = E_{Y_{i-1}}(X) = Z_n$  for some  $i$  and  $n$  (or  $i$  becomes larger than  $t$ )
- compute  $Y_{n1} = E_{A_n}(X)$ ,  $Y_{n2} = E_{Y_{n1}}(X)$ , ... until  $Y = E_{Y_{nj}}(X)$  for some  $j$
- output  $K = Y_{nj}$
- all this requires  $t$  computations of  $E$  and will succeed if  $Y$  is in  $\{Y_{ij}\}$



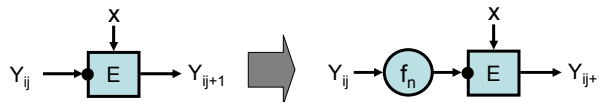
## In reality ...

- there may be collisions in the table, which produce further collisions
- the larger the table, the higher the probability that such collisions occur



## A time-memory trade-off by Hellman

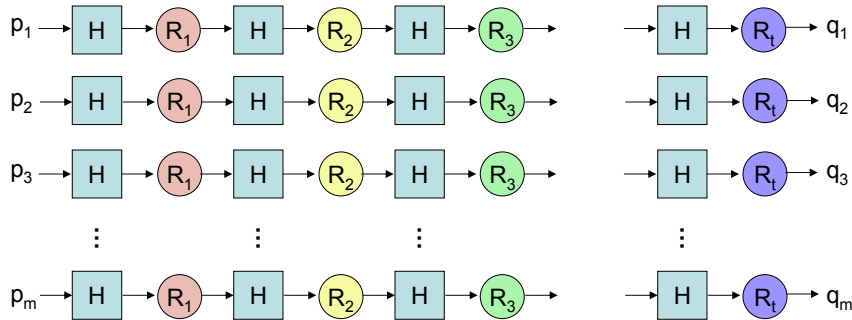
- use smaller tables, but construct  $r$  of them
- when computing table  $n$ , replace  $E_k(X)$  with  $E_{f_n(k)}(X)$ , where  $f_n$  is a simple function (e.g., a random bit permutation)



- why is this good?
  - smaller tables  $\rightarrow$  fewer collisions within a single table
  - function  $f_n \rightarrow$  if an  $Y^{(i)}$  in table  $i$  collides with an  $Y^{(j)}$  in table  $j$ , then  $f_i(Y^{(i)})$  will be different from  $f_j(Y^{(j)})$ , and hence the outputs of  $E$  will not collide ( $E_{f_i(Y^{(i)})}(X) \neq E_{f_j(Y^{(j)})}(X)$ )
- a good parameter selection:
  - $m = t = r = 2^{k/3}$
  - memory requirement:  $r \times m = 2^{k/3} \times 2^{k/3} = 2^{2k/3}$
  - computations:  $r \times t = 2^{k/3} \times 2^{k/3} = 2^{2k/3}$
  - pre-computation:  $r \times m \times t = 2^{k/3} \times 2^{k/3} \times 2^{k/3} = 2^k$
  - success probability  $> 1/2$



## Rainbow tables



- $R_i$ 's are reduction functions that map a hash output to a valid password
- given a password hash  $h$ , do the following:
  - compute  $R_i(h)$  and check for a matching  $q_n$ , if this fails ...
  - then compute  $R_i(H(R_{i-1}(h)))$  and check for a matching  $q_n$ , if this fails ...
  - ...
- if a matching  $q_n$  was found, then compute  $h_0 = H(p_n)$ ,  $h_1 = H(R_1(h_0))$ , ... until  $h_i = H(R_i(h_{i-1})) = h$ , and output  $R_i(h_{i-1})$



## Demo

- John the Ripper
  - password cracking with dictionary and brute force
  - Unix, Windows, ...
  - <http://www.openwall.com/john/>
- Ophcrack
  - cracking Windows passwords using rainbow tables
  - <http://ophcrack.sourceforge.net/>



## Countermeasures

- use of a good hash function
  - NT hash on Windows
  - md5-crypt, bcrypt (based on Blowfish) on Unix
- use of salting
  - adding a random string to the password before hashing it
  - this randomizes the hashing process and makes pre-computation attacks ineffective
  - use long enough salts (24 bits or more)
- prevention of accessing the password file
  - on modern Unix systems, the hashed passwords are stored in /etc/shadow which is accessible only to programs running with enhanced privileges



## Selection of good passwords

- DO NOT
  - choose a password based upon personal data like your name, your username, or other information that one could easily discover about you from such sources as searching the internet
  - choose a password that is a word (English or otherwise), proper name, name of a TV shows, or anything else that one would expect a clever person to put in a "dictionary" of passwords
  - choose a password that is a simple transformation of a word, such as putting a punctuation mark at the beginning or end of a word, converting the letter "l" to the digit "1", writing a word backwards, etc.
  - choose passwords less than 8 characters long and that that are made up solely of numbers or letters. Instead, use letters of different cases, mixtures of digits and letters, and/or non-alphanumeric characters.
- randomly generated passwords satisfy these rules, but they are hard to remember for ordinary people



## Good password selection example

- Make up a sentence you can easily remember. Some examples:
  - I have two kids: Jack and Jill.
  - I like to eat Dave & Andy's ice cream.
  - No, the capital of Wisconsin isn't Cheeseopolis!
- Now take the first letter of every word in the sentence, and include the punctuation. You can throw in extra punctuation, or turn numbers into digits for variety. The above sentences would become:
  - lh2k:JaJ.
  - lIteD&A'ic.
  - N,tcoWi'C!



## Measuring password strength

- strength of randomly chosen passwords against brute force attack can be calculated with precision:

$$H = L * \log_2 N$$

where N is the number of possible symbols and L is the length of the password (in symbols), and the unit of H is a bit

- H is essentially the **entropy** of a randomly chosen password
- entropy per symbol for different symbol sets:
  - decimal digits (0-9):  $\log 10 = 3.322$  bits
  - case sensitive alphanumeric chars (a-z, A-Z, 0-9):  $\log 62 = 5.954$  bits
  - printable ASCII chars:  $\log 94 = 6.555$  bits
- minimum password length for a desired strength:
  - 40 bits ~ 13 decimal digits or 7 ASCII chars
  - 64 bits ~ 20 decimal digits or 10 ASCII chars
  - 80 bits ~ 25 decimal digits or 13 ASCII chars



## Measuring password strength

- strength of user selected passwords cannot be computed precisely
- rule of thumb (by NIST):
  - estimated entropy of characters in user selected passwords (using the English alphabet):
    - char 1 → 4 bits
    - char 2-8 → 2 bits (per char)
    - char 9-20 → 1.5 bits (per char)
    - char 21 and above → 1 bit (per char)
  - example:
    - entropy of an 8 char user selected password → 18 bits
    - entropy of a 14 char user selected password → 27 bits
  - this can be improved by a password policy which requires at least one digit, one non-alpha-numeric character, mixed case, and also disallows common words findable in a dictionary
    - such an 8 character password will have an estimated entropy of 30 bits

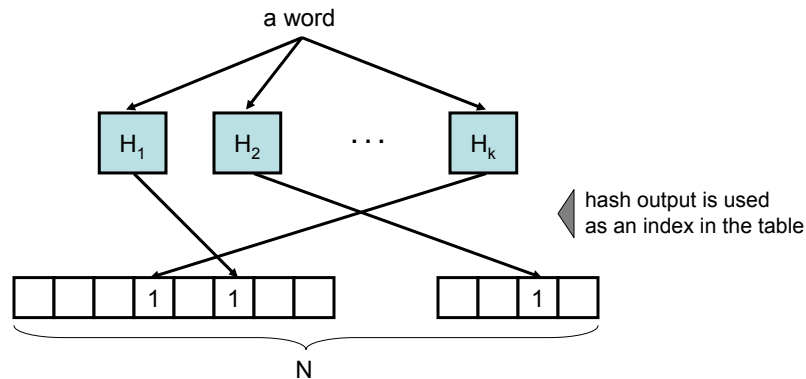


## Pro-active password checking

- algorithm that tells if a given password is strong enough
- tries to build a compact model of an entire dictionary and then decides if a candidate password matches the model (weak password) or not (strong password)
- an example: **Markov model**
  - compute a transition probability matrix  $[p_{ij}]$  such that
$$p_{ij} = f(ij) / f(i^*)$$
where  $f(ij)$  is the number of occurrences of the bigram  $ij$  in the dictionary and  $f(i^*)$  is the total number of bigrams in the dictionary beginning with  $i$
  - take the bigrams in the candidate password and use standard statistical hypothesis testing to decide if those bigrams are likely to be generated according to the transition probability distributions of the Markov model; if so, then the password is weak
  - higher order Markov models (probability of making a transition to a given character depends on multiple previous characters) give better results

## Pro-active password checking

- another example: **Bloom filter**
  - words from a dictionary of size  $D$  (in words) are mapped into a hash table of size  $N$  (in bits); resulting table represents the dictionary
  - if the hashes of the candidate password all point to an entry which is set, then the password is considered weak
  - words from the dictionary are always rejected, but false positives are possible



## Lecture outline

- introduction
- password based authentication and password cracking
- authentication with HW tokens
- biometric identification
- CAPTCHAs
- summary

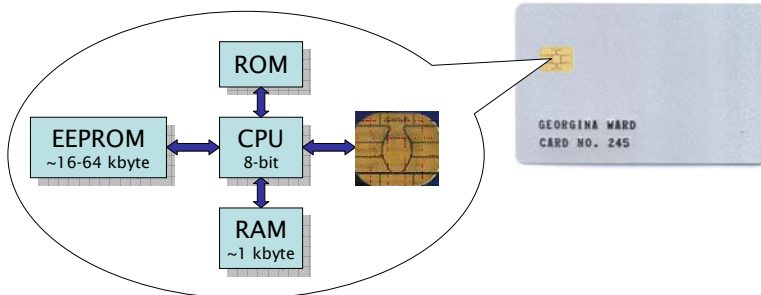
## One-time password generators

- paper based list of OTPs
  - used in practice by a Swiss bank a decade ago
- sending a OTP via SMS
  - out-of-band channel is assumed to be secure
  - used by many banks today
- off-line OTP generators (e.g., RSA SecureID)
  - displays a new OTP periodically (e.g., every minute)
  - OTP is generated from a user specific secret and the time
  - token must stay in synch with the verifier system
- usually combined with traditional passwords or PINs (two factor authentication)



## Smart cards

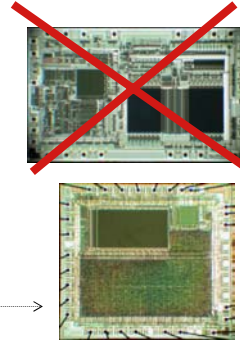
- a smart card is microcomputer embedded in a plastic card (credit card size or smaller)



- many smart cards support cryptographic operations
  - custom hardware for block ciphers and modular arithmetics

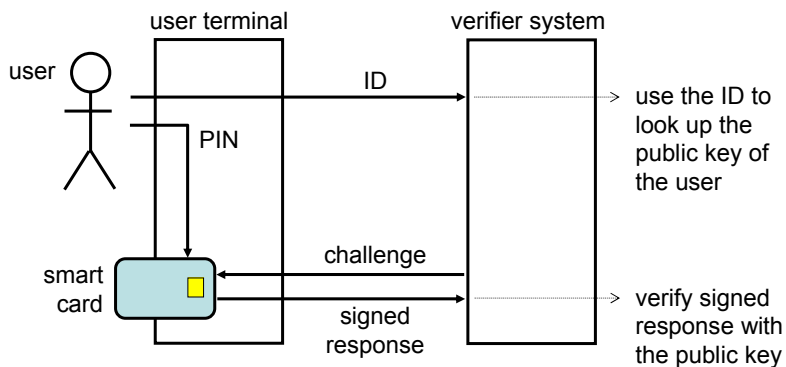
# Security of smart cards

- access control
  - smart cards have a single interface
  - access to sensitive data through the interface is controlled by the smart card (OS)
  - authorization of access is based on PIN codes
  - after a certain number of unsuccessful attempts, the card blocks itself
  
- physical protection and tamper resistance
  - internal voltage sensors to protect against under- and over-voltages used in power glitch attacks
  - clock frequency sensors to prevent attackers slowing down the clock frequency for static analysis and also from raising it for clock-glitch attacks
  - top-layer metal sensor meshes
  - internal bus hardware encryption to make data analysis more difficult
  - randomized ASIC-like logic design (glue logic)



# Authentication with smart cards

- smart cards can store cryptographic keys and run cryptographic algorithms → they can perform any crypto based (strong) entity authentication protocol on behalf of their owner



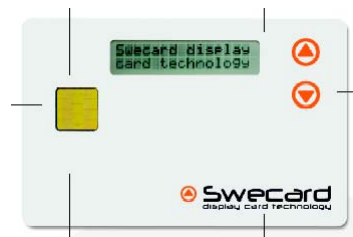


## A potential problem

- smart cards have no user interface → PIN must be entered through the user terminal
- a malicious terminal can use the PIN to request a signature from the smart card on any message
- what if the terminal is not trusted by the user?
  - examples for untrusted terminals:
    - a terminal installed at a public place (e.g., a PC in a hotel or airport lounge, Internet cafe, ...)
    - a terminal operated by an untrusted principal (e.g., an ATM or a payment terminal of an unknown merchant in a foreign country)
    - the user's own PC running Windows



## Smart cards with display

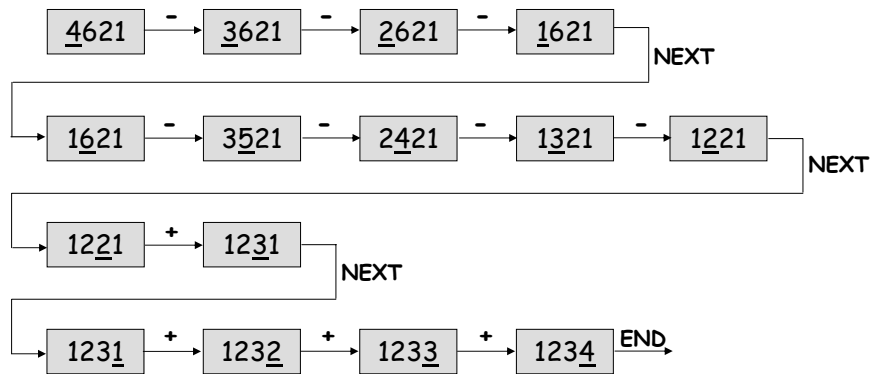


- a confidential output can be converted into a confidential input (PIN remains hidden from the terminal)
  - the smart card displays a random sequence of digits
  - the user enters a sequence of increment, decrement, and next digit commands via the terminal, and alters the original value until the smart card displays the desired input value
  - this is essentially a one-time pad!



## Example

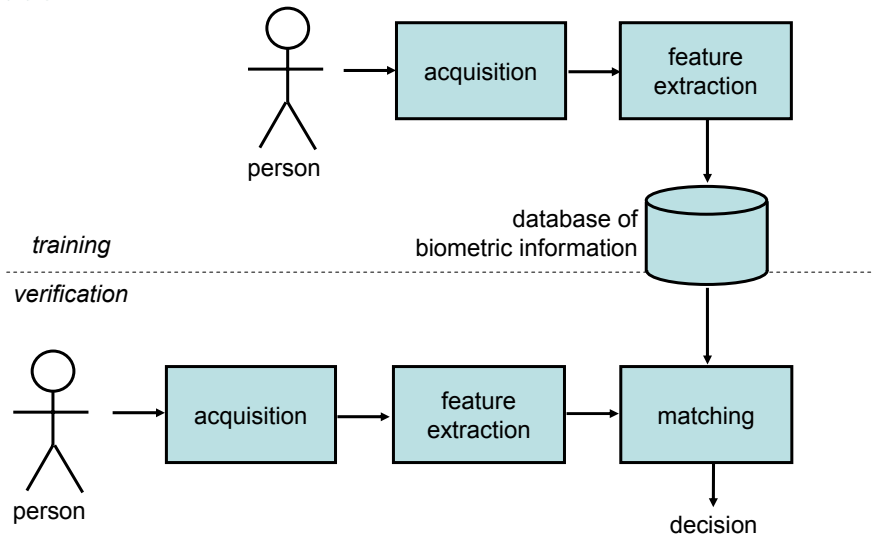
- assume you want to enter your PIN: 1234



## Lecture outline

- introduction
- password based authentication and password cracking
- authentication with HW tokens
- biometric identification**
- CAPTCHAs
- summary

## General model



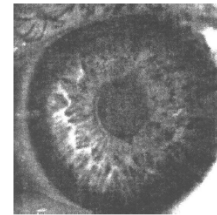
## Biometrics

- any human physiological or behavioral characteristic could be a biometrics provided it has the following properties:
  - **universality** – every person should have the characteristic
  - **uniqueness** – no two persons should be the same in terms of the characteristic
  - **permanence** – the characteristic should be invariant with time
  - **collectability** – the characteristic can be measured quantitatively
  - **circumvention** – it should be difficult to fool the system by fraudulent techniques
- other desirable properties:
  - **performance** – refers to the achievable identification accuracy, the resource requirements to achieve an acceptable identification accuracy, and the working or environmental factors that affect the identification accuracy
  - **acceptability** – indicates to what extent people are willing to accept the biometric system



## Examples

- fingerprint
  - flow-like patterns present on human fingers
  - their formation depends on the initial conditions of the embryonic development and they are believed to be unique to each person (and each finger)
  - one of the most mature biometric technologies (long history)
- iris
  - visual texture of the human iris is determined by the chaotic morphogenetic processes during embryonic development and is posited to be unique for each person and each eye
  - a position-invariant constant length feature vector is derived from an annular part of the iris image based on its texture
  - iris verification based on the feature vector is extremely fast, and the identification error rate is believed to be extremely small
  - capturing an iris image involves cooperation from the user, both to register the image of iris in the central imaging area and to ensure that the iris is at a predetermined distance from the focal plane of the camera



## Examples

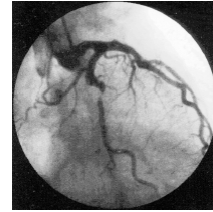
- face
  - there are two primary approaches to identification based on face recognition:
    - eigenfaces:
      - the universe of face image domain is represented using a set of orthonormal eigenfaces (~basis vectors), which are derived from the covariance analysis of the face image population
      - two faces are considered to be identical if they are sufficiently "close" in the eigenface feature space
    - face attributes:
      - facial attributes like nose, eyes, etc. are extracted from the face image and the invariance of geometric properties among them are used for recognizing features
  - advantage: acquiring face images is non-intrusive
  - disadvantages: problem of aging, facial disguise, facial expressions, ...
- ear
  - shape of the ear and its structure are distinctive, but not expected to be unique to each individual



## Examples

### retina scan

- the retinal vasculature is rich in structure and is supposed to be a characteristic of each individual and each eye
- the image capture requires a person to peep into an eye-piece and focus on a specific spot in the visual field → requires a conscious effort on the part of the user
- retinal vasculature can reveal some medical conditions, e.g., hypertension, which is another factor standing in the way of public acceptance



### hand geometry

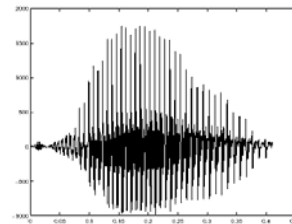
- in recent years, hand geometry based solutions have captured almost half of the physical access control market
- the representational requirements of the hand are very small (9 bytes) which is an attractive feature for bandwidth and memory limited systems
- the hand geometry is not unique and cannot be scaled up for systems requiring identification of an individual from a large population of identities



## Examples

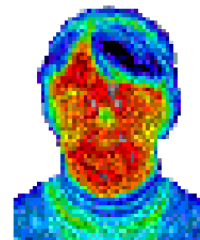
### voice

- voice is characteristic for an individual, however, it is not expected to be sufficiently unique
- moreover, voice signals are typically degraded in quality by the microphone and communication channel
- voice is affected by a person's health (e.g., cold), stress, emotions
- problem of mimicking others
- features extracted from multiple bands may be either time-domain or frequency domain features
- speaker identification can be text-dependent, text-independent, or language independent (with increasing complexity)



### facial thermogram

- infrared sensors can acquire an image indicating the heat emanating from different parts of the body
- absolute values are not completely invariant to the identity of an individual → need normalization
- infrared sensors are prohibitively expensive
- could be used for covert identification solutions





## Examples

- keystroke dynamics
  - each person types on a keyboard in a characteristic way, yet this behavioral biometrics is not expected to be unique to each individual
  - can be based on
    - time durations between keystrokes
    - dwell times - how long a person holds down a key
  - this is a behavioral biometric → one may expect to observe large variations of typing patterns of the same individual
- signature and acoustic emission
  - the way a person signs her name is known to be a characteristic of that individual
  - however, signatures evolve over time and are influenced by physical and emotional conditions of the signatories
  - signatures of some people vary a lot: even the successive impressions of their signature are significantly different
  - professional forgers can reproduce signatures; although, human experts can discriminate genuine signatures from forged ones, modeling the invariance in the signatures and automating signature recognition process are challenging
  - a related technology is authentication of an identity based on the characteristics of the acoustic emissions emitted during a signature scribble



## Comparison of approaches

Biometrics	Uniqueness	Permanence	Collectability	Circumvention	Performance
Fingerprint	High	High	Medium	High	High
Iris	High	High	Medium	High	High
Face	Low	Medium	High	Low	Low
Ear	Medium	High	Medium	Medium	Medium
Retina	High	Medium	Low	High	High
Hand geo.	Medium	Medium	High	Medium	Medium
Voice	Low	Low	Medium	Low	Low
Thermo	High	Low	High	High	Medium
Key stroke	Low	Low	Medium	Medium	Low
Signature	Low	Low	High	Low	Low



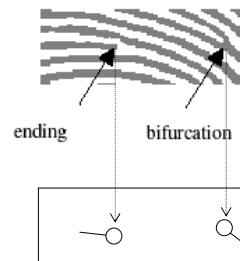
# Fingerprints

- fingerprints have been used for forensic purposes for more than a century (mainly in criminal applications)
- technological advances in personal computers and optical scanners make fingerprint capture practical in non-criminal applications
- fingerprints are unique and state-of-the-art fingerprint matching algorithms are quite reliable
- two facets of the fingerprint matching problem:
  - one-to-one matching or **fingerprint verification**: comparison of a *claimant* fingerprint against an *enrollee* fingerprint
  - one-to-many matching or identification: a fingerprint (of unknown ownership) is matched against a database of known fingerprints



# Fingerprint features

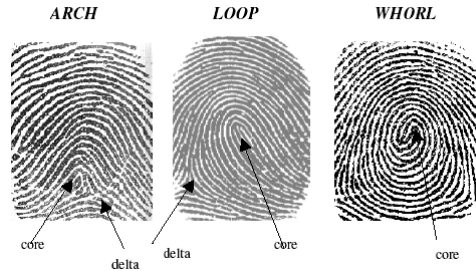
- ridges and valleys
  - ridges: the lines that flow in various patterns across the fingerprint
  - valleys: the spaces between ridges
  - when matching two fingerprints, essentially the patterns of their ridges are compared
- minutiae
  - ending: termination of a ridge
  - bifurcation: split of a ridge from a single path to two paths (Y-junction)
  - minutia representation:
    - type (ending, bifurcation)
    - position (x, y)
    - direction ( $\phi$ )





## Fingerprint features

- global patterns
  - more macroscopic pattern of the flow of ridges
  - three basic types: arch, loop, whorl



- core and delta
  - core: center area of a fingerprint pattern
  - delta: a singular point from which three patterns deviate



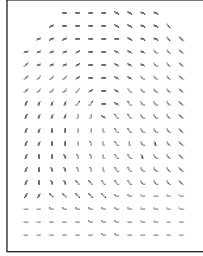
## Fingerprint processing steps

- image capture
  - obtaining the fingerprint image
- noise reduction and image enhancement
  - uses inherent redundancy of parallel ridges
  - based on the local orientation of the ridges around each pixel, ridges oriented in the same direction as those in the same locality are enhanced, and anything oriented differently are decreased
  - this eliminates noise that may be joining adjacent ridges (flowing perpendicular to the local flow)
- feature extraction
  - **adaptive thresholding**: binarization of the image from gray-scale to black and white
  - **thinning**: reducing the widths of the ridges down to a single pixel
  - **minutia detection**: quite straightforward after thinning

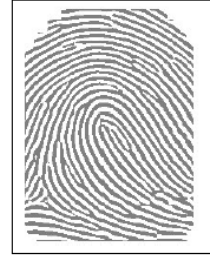
## Fingerprint processing steps



original image



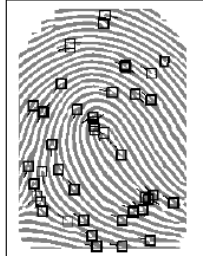
orientation



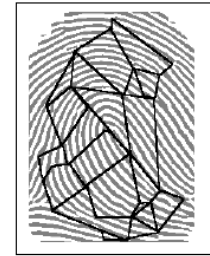
binarization



thinning



minutia detection



minutia graph

## Fingerprint matching

- the most common is minutiae based matching
  - uses local minutia structures to quickly find a coarse alignment between two fingerprints and then consolidate the local matching results at a global level
- it consists of four steps:
  - compute pairwise similarity between minutiae
    - use minutia descriptors that are invariant to rotation and transposition
    - e.g., structure of neighborhood, distances between minutia and its neighbors, their relative positions, but also own attributes of each minutia (e.g., type)
  - alignment of the two fingerprints according to the most similar minutia pairs
  - establishment of minutia correspondence
    - minutiae that are close enough both in location and direction are deemed to be matching
  - computing a global similarity score between the two fingerprints based on the number of matching minutia pairs
    - if this similarity score is beyond a threshold, then the fingerprints are said to match
- template matching can often be visualized as graph matching (i.e., comparing the shapes of graphs joining fingerprint minutiae)



## Some extensions

- the number of neighborhood pairs to be compared is often reduced by aligning the claimant and the enrollee fingerprints using the core and the delta
- efficiency of matching can also be improved by sorting the list of minutiae
  - e.g., a linearly sorted list of minutiae can be compiled by scanning the fingerprint from a selected center point outward by a predetermined scanning trajectory such as a spiral
- another approach to find matching minutiae is to compare their k-nearest neighbor “stars” (edges emanating to the k nearest neighbor)



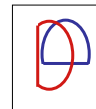
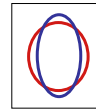
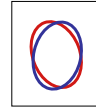
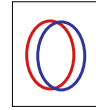
## Other matching approaches

- image correlation
  - instead of using minutiae, some systems perform matches on the basis of the overall ridge patterns of the fingerprints
  - standard correlation techniques can be used to compute the correlation between two images in the spatial domain or in the spatial frequency domain (after computing the FFT of the images)
  - if correlation is above a threshold, then the two fingerprints match
- phase matching
- skeleton matching



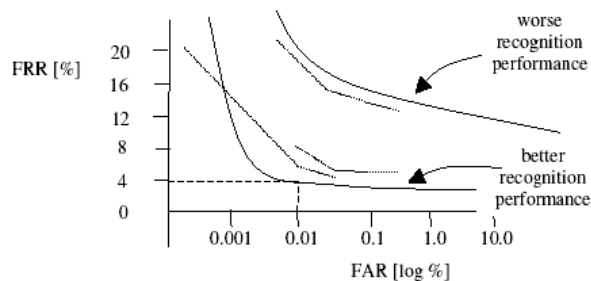
## Performance measures

- verification errors are unavoidable
  - fingerprints will likely have different locations within their respective bounding boxes
  - fingerprints may have different orientations
  - because of skin elasticity, even if fingerprints match in location and orientation, all sub-regions may not align
  - there is the inevitable problem of noise
    - discontinuities in ridges will be different depending on finger dryness
    - the portion of the fingerprint captured in each image will be different
    - ...
- two types of errors:
  - false rejection (FR) or Type I error
  - false acceptance (FA) or Type II error



## Performance measures

- performance of a fingerprint verification systems can be characterized by its ROC-curve
  - ROC: Receiver Operating Curve (historical name)
  - the ROC-curve plots FA rate (FAR) versus FR rate (FRR) for a system
  - the typical ROC-curve has a shape whose "elbow" points toward (0,0) and whose asymptotes are the positive x-and y-axes
  - the sharper the elbow and (equivalently) the closer is the ROC-curve to the x- and y-axes, the better is the performance





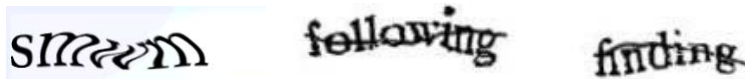
## Lecture outline

- introduction
- password based authentication and password cracking
- authentication with HW tokens
- biometric identification
- **CAPTCHAs**
- summary



## CAPTCHA

- Completely Automated Public Turing test to tell Computers and Humans Apart
- the objective is not to verify the identity of the claimant, but to decide if the claimant is a human user or a computer program (bot)
- typically based on open problems in AI: problems that are hard for a computer program to solve but easy for a human
  - e.g., decoding images of distorted text





## Applications

- preventing comment spam in blogs
  - programs should be prevented to submit bogus comments
- protecting website registration
  - programs should be prevented to create a large number of registration requests for free web accounts
- protecting e-mail addresses from spammers
  - e-mail addresses on web sites should be readable only for humans
- online polls
  - programs should be prevented to cast votes



## Applications (cont'd)

- search engine bots
  - it is sometimes desirable to keep web pages unindexed to prevent others from finding them easily; hence search engine bots should be kept out
- preventing the spread of worms and spam
  - “I will only accept an email from you if you are a human“
- preventing on-line brute force attacks on passwords
  - blocking an account after a given number of unsuccessful login attempts also locks out the legitimate user
  - instead, programs should be prevented to make any login attempts



## Defeating CAPTCHAs

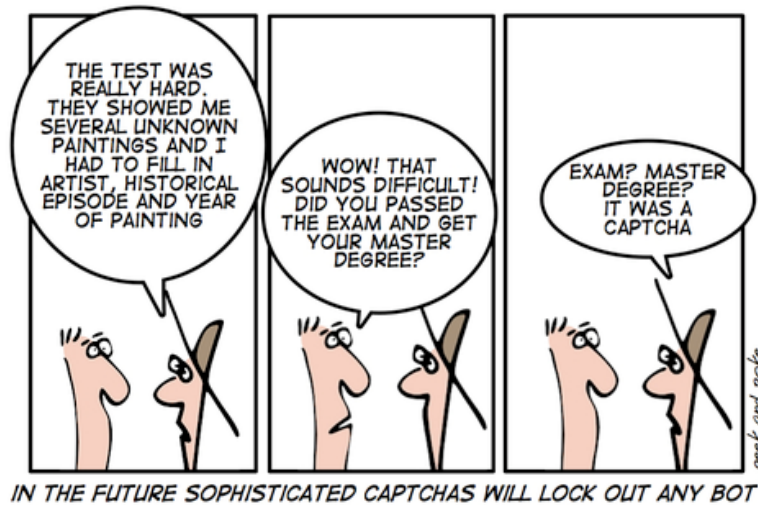
- exploiting design flaws and bugs in the implementation
  - e.g., re-using a session ID of a known CAPTCHA image
  - e.g., hash of the solution is passed to the client for verification
  - e.g., small pool of CAPTCHA images
- improving character recognition software
  - pre-processing: removal of background clutter and noise
  - segmentation: splitting the image into regions which each contain a single character (this is still hard for computers)
  - classification: identifying the character in each region
- using cheap human labor to process the tests
  - automatically relaying puzzles to a group of human operators
  - copying the CAPTCHA images and using them as CAPTCHAs for a high-traffic site owned by the attacker



## reCAPTCHA

- a project that uses CAPTCHA to help digitize books while protecting websites from bots
- operation
  - scanned text is subjected to analysis by two different OCR programs
  - in case the programs disagree, the questionable word is converted into a CAPTCHA, and displayed along with a control word already known
  - the system assumes that if the human types the control word correctly, the questionable word is also correct
  - reCAPTCHA tests are taken by servers from the central site of the reCAPTCHA project through a JavaScript API (with the servers making a callback to reCAPTCHA after a request has been submitted)
- MailHide
  - reCAPTCHA offers html code for e-mail addresses supplied by a client that replaces the address with a link to the reCAPTCHA server
  - the e-mail address is revealed only if a CAPTCHA is solved

## The future of CAPTCHAs



## Summary

- user authentication is a basic security function, which is usually a prerequisite of making access control decisions
- three basic approaches
  - passwords and alike (what you know)
    - simple, intuitive, and cheap, but has many disadvantages
    - guessing, dictionary, and brute force attacks
    - time-memory trade-off techniques
    - password strengthening (guidelines, pro-active checking)
  - hardware tokens (what you have)
    - one-time password generators and smart cards
    - untrusted terminal problem
  - biometrics (what you are)
    - many different techniques with various trade-offs
    - fingerprint verification and its performance measures
- CAPTCHAs: puzzles that are used to distinguish computer programs from human users