Authentication Security Mechanism

PRINCIPLES
MECHANISM
AUTHENTICATION FACTORS
REMOTE AUTHENTICATION

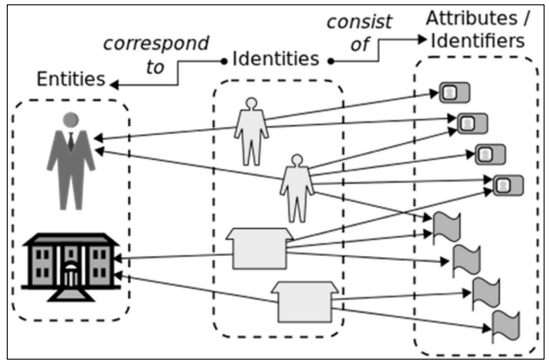
APM@FEUP

Entities and Identities

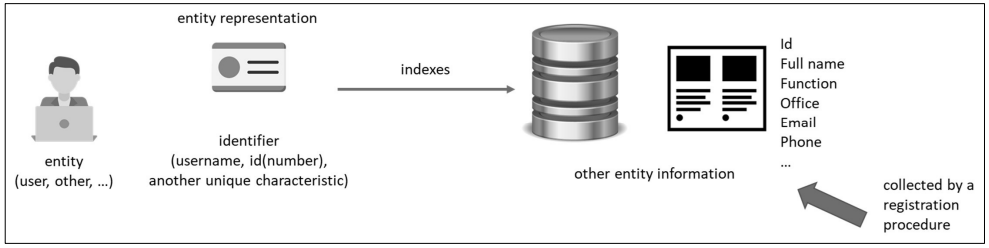
Entity or **principal** in a system

Any actor that needs a distinction between different instances

Identities are unique representations of an entity



Identity system implementation



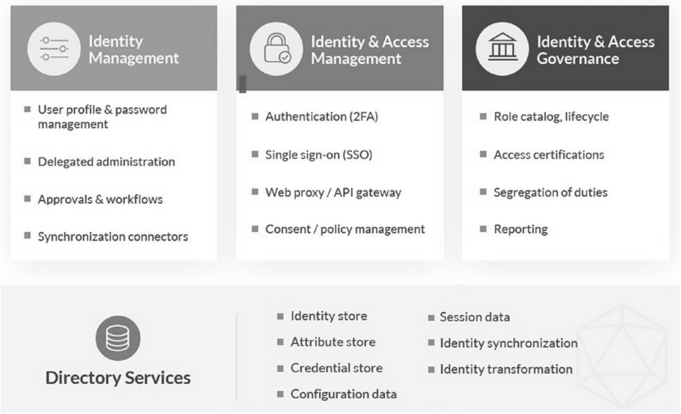
Protecting access

- **In many organizations today**
  - **Specialized protection of resources (files, databases, …), services and APIs, applications, and application specific functionalities, is needed**
  - **For that, many times, user (or entity) identity and access control is also implemented as a set of independent services**

IdM – Identity Management
(information about users)

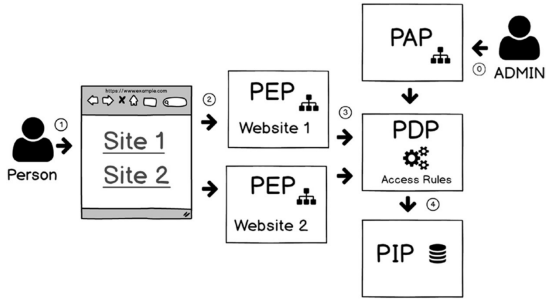
IAM – Identity and Access Management
(authentication, authorization, consumes IdM information)

IAG – Identity and Access Governance
(administration, logging, detection)



The PDP-PEP pattern

- **In operation, the protection system in an organization, has usually the following architecture**



PEP – Policy enforcement point
(application of access policies)

PDP – Policy decision point
(policies: who can access what and how, in what conditions)

PIP – Policy information persistence
(user and policy persistence in some custom or standard database – like LDAP, Directory, SAML, …)

PAP – Policy administration point
(app for protection administration)

A user ① accessing resources or applications (usually using a browser) goes to some web server or service ②. The server can act as, or delegate, to a decisor of access (the PEP). The PEP queries the PDP ③ for access rules to some protected resource or functionality. The PDP can authenticate the user and consult the corresponding access policies. The information is on the PIP ④. The PAP application ⓪ allows an administrator to create, modify, delete access rules (who has access to what, and how), and the user information relative to its identity and authentication information.

Authentication Definition

- Can be defined as the "binding of an identity to an entity"
  - An identity is a representation or 'name' of some entity
  - entities in computer systems (also called principals) can be users (of an operating system, or application), can be computer nodes on a network, or even can be programs (applications) executing on the system
- Authentication is a fundamental security building block
  - Is the basis of access control and accountability, and a trusted proof mechanism for an identity
- Is distinct from message authentication
  - message authentication has to do with the integrity of messages sent between two parties
  - user authentication establishes (or allows trust about) the user identity

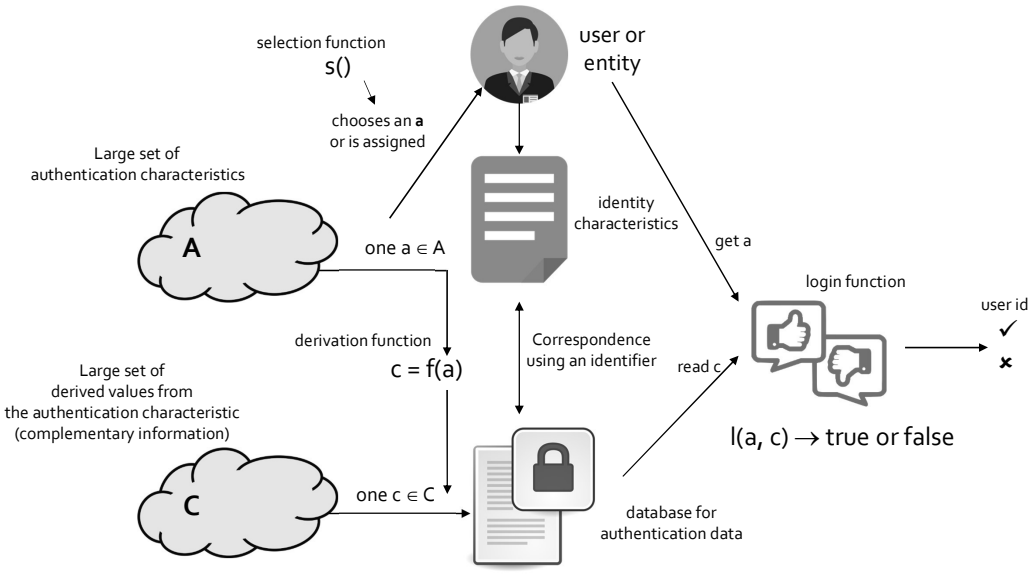
Process

- For authentication, some steps are needed
  - 1. Registration
    - The information about the entities to be authenticated should be collected and stored first
    - In this information there should be an identifier representing the identity of the entity
    - Other information (location, function, contact, …)
    - Complementary information associated with the entity (passphrase, password, public key, biometrics, …)
    - The registration process should be done in a way to prevent imposture
  - 2. Identification
    - System interactions (e.g., login), allowing the user to specify his identifier
  - 3. Verification
    - The system (an inner and protected component of the OS) verifies the previous claim
    - Asks to supply, in some form, some of the recorded complementary information

Elements of the Authentication Mechanism

- The Authentication mechanism has several components
  - A set A of possible information for proving the identity (authentication information)
    - The #A should be very large
    - An entity have an association to an element a ∈ A (by choosing or assignment)
  - A set C of complementary information, which is stored, and used for validating the elements of A
    - Usually, each element of C (c ∈ C) is derived from an element of A
  - A set of one or more functions F (complementation functions) used to generate a c from a
    - That is, a f ∈ F is a function of A → C (f(a) → c)
      - It can be a cryptographic encryption, or more commonly, a cryptographic hash
  - One or more authentication functions L to verify identity
    - A function l ∈ L is a function of A×C → {true, false} (f(a, c) → true || false)
  - A set of selection functions S, allowing an entity to change its authentication information (a ∈ A)
    - A function s ∈ S should allow an entity to choose another a, or to get a new association to another a; in the process the value c = f(a) is stored

Local authentication mechanism



Factors of User Authentication

- The verification process uses one or more of usually four characteristics' types (factors) associated with users
  - Something the user knows
    - Can be passphrases, passwords, PINs, etc.
  - Something the user possesses physically
    - A key, a token, a smartcard, a smartphone, capable of interact with the system
  - Something the user is (distinctive)
    - Also called static biometrics
    - fingerprint, face, retina, iris, etc.
  - Something the user does (distinctively)
    - Also called dynamic biometrics
    - voice, typing, signature (handwritten), etc.
- The four factors can be used alone or combined
  - 2FA (two factor authentication, for two different characteristics' types)
  - MFA (multi-factor authentication, usually for more than two)
- All have issues

Password Authentication

- A password should be a unique string known by the entity
- Still the most widely used authentication method
  - User provides username/login id
  - System asks for password (some $a \in A$)
  - System compares password with that previously saved for the supplied identifier (reading c, associated with id, and applying f(a))
    - This operation is l(a, c)
- After positive authentication
  - Verification that the authenticated user is authorized to access the system
    - Some restriction (policies) can exist on access hours and places (terminals), password ageing, …
  - Determines the authenticated user privileges
  - Uses the user identity for access control of system resources
    - Create processes, execute programs, access files, query databases, …

Example: the Authentication Elements

- A user authenticates by an eight-character password, stored in a database table, indexed by a user id
  - In this case the set A is composed by all possible strings of 8 characters, usually restricted to printable Latin characters (say about 96 different ones)
    - There are 96^8 possibilities ($= 7.2 \times 10^{15}$), but if it is allowed a user to choose it, they are not equiprobable
    - A user password will be one of them ($a \in A$)
  - In this case C = A
  - Also, f() is the identity function (I()), that is, f(a) = a
  - The function l(), (login) just verifies if the supplied a is equal to the stored one, indexed by the supplied user id
  - A function s() should allow store a new a in the database table, indexed by the initially supplied user id (in the login process, and after authentication)

Password Vulnerabilities and Countermeasures

- Vulnerabilities
  - offline dictionary attacks
  - specific account attack (for a specific user, from his characteristics …)
  - popular passwords attack (against a wide range of users)
  - workstation hijacking
  - exploiting user mistakes and social engineering
  - exploiting multiple password use
  - electronic monitoring
- Countermeasures
  - protect password file
  - intrusion detection (hour, place, access pattern, errors, …)
  - account lockout mechanism
  - password policies
  - automatic logout
  - encrypted communications
  - training and enforcement of policies

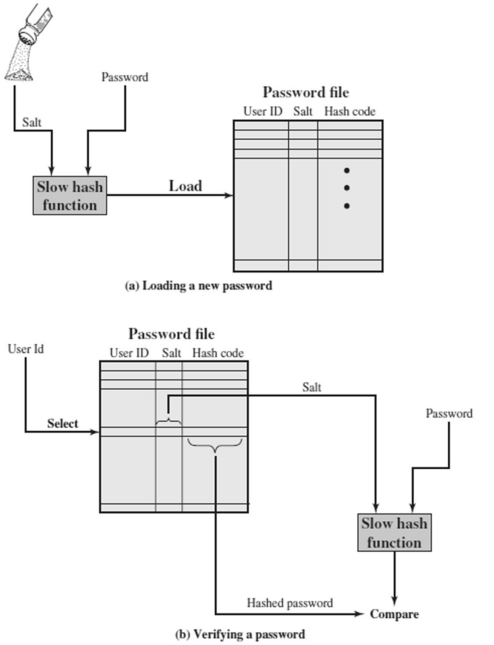
Local Linux Password System

hash code – passwords stored as a cryptographic hash calculated value (nowadays some variation of SHA-512)

slow hash – cryptographic hash applied many times (e.g., 1000 or 5000 times)

salt – random value or string with some size characteristics

files scattered in the system, with protected access (passwd, shadow, …)

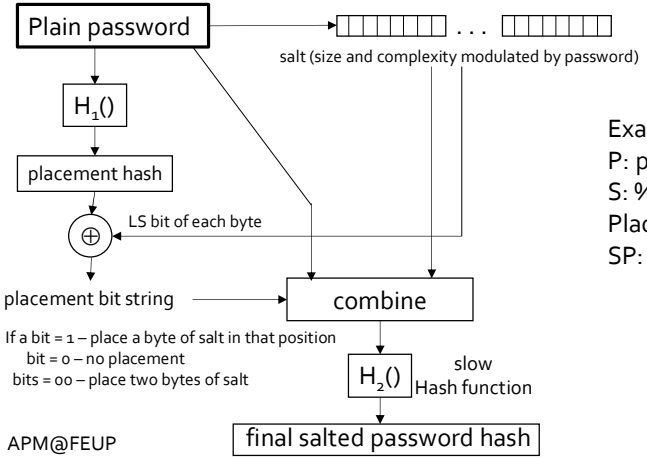


What is a Salt ?

- Random value to be combined with a password
- Prevents duplicate passwords from being visible in the password files
  - they produce different hashes with different salt values
- Increase the difficulty of offline dictionary attacks
  - If the salt is not known to the attacker
  - The attacker tries to find a password with the same hash as the one stored in the password file
- Not possible to know if the user has the same password in several different systems
  - The salt modifies the stored hash value that corresponds to the password

Modern (Dynamic) Salt Use

- Normally salts are randomly generated with a fixed size and algorithm, and pre- or post-pended to the plain password
- More recently the size (10 to 32 characters) and complexity depend on a size and complexity evaluation of the password
  - The salt is then added according to a placement algorithm



Example:
P: password
S: %@&03U+
Placement: 10101010
SP: p%as@sw&ord3U+

Anatomy of a Local Password Attack

- Goal
  - Find an a ∈ A such that some f(a) = c, associated with an identity
  - c and f() must be known to the attacker
- Direct attack
  - Find f() by researching the operating system or application
  - Find c, getting access to file or database where is stored
- Indirect approach
  - Make system trying l(a), for some entity, and see the result (true or false)
    - Many systems have limitations on the number of failed trials if l(a) is tried on login

Password Cracking Attacks

Dictionary attacks
- Develop a large dictionary of possible passwords and try each against the password file
- Each password must be hashed using each salt value and then compared to stored hash values

Rainbow table attacks
- Pre-compute tables of hash values for all salts
- A mammoth table of hash values
- Can be countered by using a sufficiently large salt value and a sufficiently large hash length

Password crackers exploit the fact that people choose easily guessable passwords
- Shorter password lengths are also easier to crack
- Potential passwords are generated with common variations

John the Ripper
- Open-source password cracker first developed in in 1996
- Uses a combination of brute-force and dictionary techniques
- Generates variations of dictionary passwords

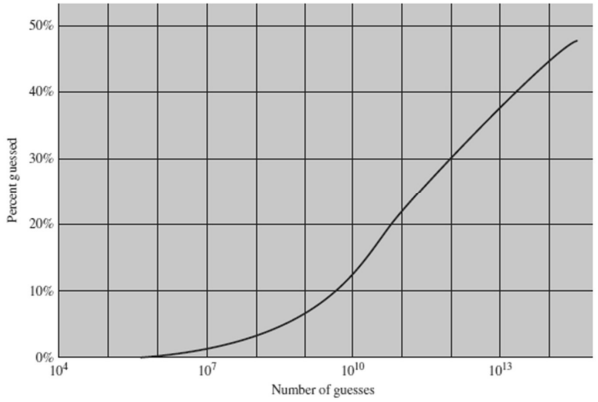
Dictionary and Rainbow Attacks

- **Dictionary attacks**
  - try each word and obvious variants in a large dictionary against hash in password file
    - facilitated if the salt is also known
    - need to know the hashing algorithm or applied variation
    - Can take a large amount of time
- **Rainbow table attack**
  - Increases speed of attack
  - Uses a large dictionary of possible passwords
  - for each password in dictionary
    - precompute a table of hash values for all possible salts
    - results in a huge table of hash values (generated from a dictionary and small hashes) of more than 10 billion entries was able to crack 99.9% of small alphanumeric passwords in 14 s, some years ago …
    - since then, salts and hashes increased in size, and password policies were made more difficult
    - A big enough rainbow table can take months (or even years) to generate

Another Case Study

- Some years ago (2013) 25000 passwords picked by students at a university, with a complex password policy, were analyzed and tried to crack
- over 10% recovered after 10^{10} guesses (dictionary and variations)

Mazurek, M., et al.
"Measuring Password Guessability for an Entire University."
Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security,
November 2013



Using Better Passwords

- Care must be used when picking or assigning passwords
- The goal is to eliminate guessable passwords
  - Preferably still easy to remember by users
  - Taking note of passwords can be OK if the user takes some precautions
    - Using a password storage application (with a master password for encryption)
- Techniques for password picking
  - <u>User education</u>
    - Making the user aware of the perils of guessable passwords
  - <u>Computer-generated passwords</u>
    - Completely random can be very difficult to enter and remember
    - NIST FIPS 181 defines an algorithm to generate pronounceable concatenation of syllables (needs to increase the size for same #A)
  - <u>Reactive password checking</u>
    - Periodically checks weaknesses in password guessing (running its own password cracker)
  - <u>Proactive password checking (at the time of selection) or Complex Password Policy</u>
    - Enforcement of password policy rules (rejected at the moment, if fails checking)
    - use a Bloom filter (technique to quickly check if a candidate is in a large dictionary) (OPUS checker)

Password Anderson Formula

- Anderson formula measures the probability of an attacker guessing a password in a certain interval of time
  - We need to know the time interval (T)
  - In an offline testing of the guess, we also need to know the number of tests we are able to perform per time unit (G)
  - And we need to know the possible number of passwords under consideration, that should be equiprobable (N)
- With these assumptions the probability (P) is calculated as:

$$P \geq \frac{TG}{N}$$

- Example:

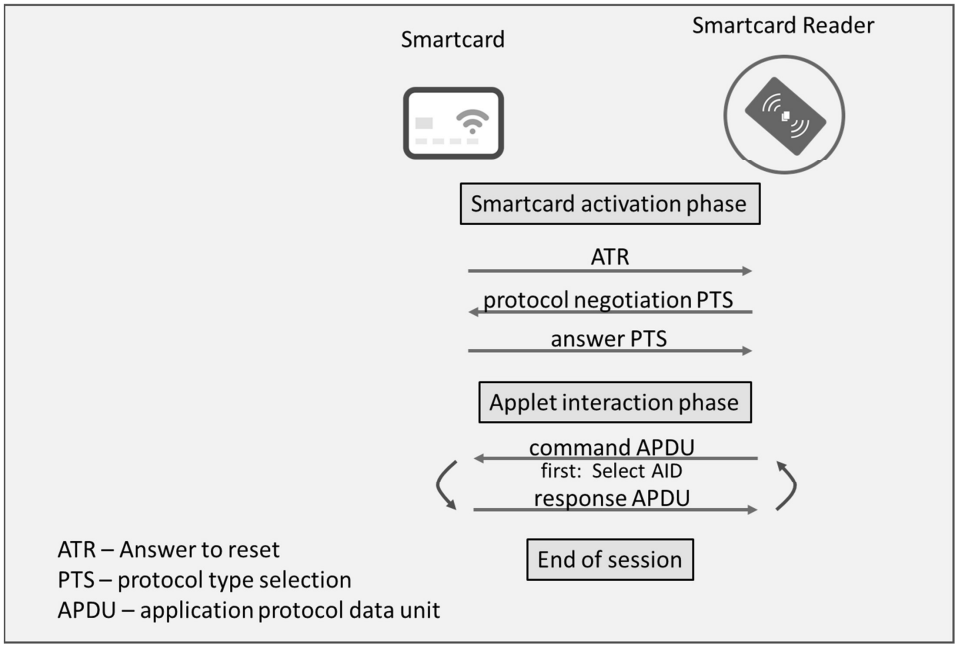
  Let passwords be composed of characters from a 65 alphabet, and suppose we can test 10^6 passwords per second. How long should a password be (L characters) to guarantee a probability at most 1/1000 over 1 year of testing?

  We have $N \geq \frac{TG}{P} = \frac{365 \times 24 \times 3600 \times 10^6}{0.001} = 31.536 \times 10^{15}$, $N = 65^L$, $L \geq 10$

Token Based Authentication

- Object possession to use as authentication
  - Memory objects
  - Smartcards
- Memory objects store but not process data
  - Used after reading for access (e.g., hotel rooms)
  - The access hardware can verify a PIN (or password) also stored in the object
  - Have some drawbacks
    - Can be easy to duplicate
    - Needs special readers
    - A loss can be problematic
    - User dissatisfaction
- Smartcards
  - Has memory, processor, and I/O
    - Can generate a dynamic password (e.g., based on date/time or other parameters)
    - Can use challenge / response
    - Can use a PIN as second factor
- One-time passwords (OTP) and devices

Smartcard Operation



Electronic Identity (eID) System

- Used in passports, citizen cards, drivers license
  - provides a national electronic identity (eID)
    - Its implementation is based on wired or wireless smartcards
- Can provide a stronger proof of identity and signature
- Usual data stored in the card
  - Personal data (name, address, birthplace, birthdate, …)
  - Unique document number
  - Card access codes (PINs)
  - Machine Readable Zone (public info)
  - Private Key
  - Public Key and Certificate(s)
  - Can use challenge / response for proving identity
  - Used also in official digital signatures

Functions and Data for eID Cards

Function	Purpose	PACE Password	Data	Uses
ePass (mandatory)	Authorized offline inspection systems read the data	CAN or MRZ	Face image; two fingerprint images (optional), MRZ data	Offline biometric identity verification reserved for government access
eID (activation optional	Online applications read the data or acess functions as authorized	eID PIN	Family and given names; artistic name and doctoral degree: date and place of birth; address and community ID; expiration date	Identification; age verification; community ID verification; restricted identification (pseudonym); revocation query
	Offline inspection systems read the data and update the address and community ID	CAN or MRZ		
eSign (certificate optional	A certification authority installs the signature certificate online	eID PIN	Signature key; X.509 certificate	Electronic signature creation
	Citizens make electronic signature with eSign PIN	CAN		

CAN – card access number
MRZ – machine readable zone
PACE – password authenticated connection establishment
PIN – personal identification number

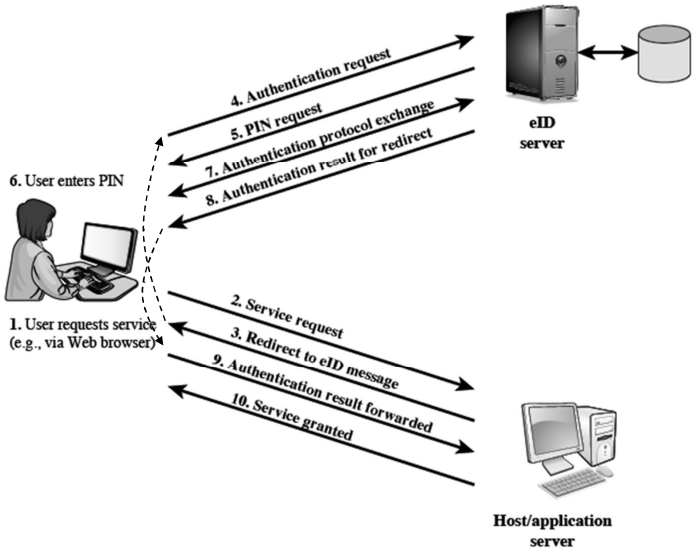
PACE

Ensures that the contactless RF chip in the eID card cannot be read without explicit access control

For online applications, access is established by the user entering the 6-digit PIN (which should only be known to the holder of the card)

For offline applications, either the MRZ printed on the back of the card, or the six-digit card access number (CAN) printed on the front is used

User Authentication with eID

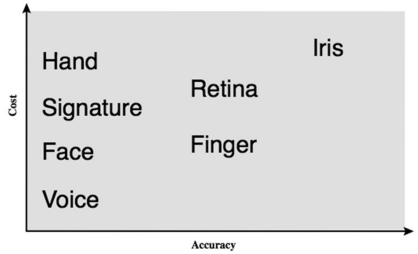


One Time Passwords (OTP)

- **Passwords that are used only once (age policy)**
  - **The entity and the authenticator must have a means of calculating the next same password**
  - **For a human, usually a device in his possession is used, or a software generator**
    - **Needs an initial synchronization with the authenticator**
- **Several methods have been proposed and implemented**
  - **S/Key – Uses a series of hashes, from an initial seed K**
    - **To attack the next password one hash function h() should be inverted**
    - **Only brute-force attacks are known, infeasible if h() has a large result**
    - **OPIE is an implementation device of S/Key**
  - **HOTP – HMAC based OTP Algorithm**
    - **Uses a shared key K, and an 8-byte counter c, in a standard described in RFC 4226**
    - **The counter is incremented each time a new password is generated**
  - **TOTP – Time based OTP Algorithm**
    - **Defines an initial counter time t_0, a time step x, and uses a time variable t**
    - **Described as a standard in RFC 6238**
    - **Time t must be synchronized (within a tolerance), and there is a resynchronization mechanism defined**

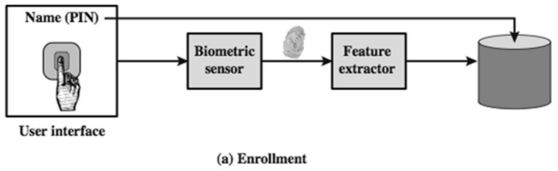
Biometric Authentication

- Based on the user static or dynamic physical characteristics
  - Usual characteristics used in authentication systems
    - Face detection and characterization
    - Fingerprint acquisition and processing
    - Hand geometry and lines
    - Retina pattern acquisition and characterization
    - Iris patterns
    - Voice characterization pronouncing a known text (dynamic)
    - Handwritten signature (dynamic)

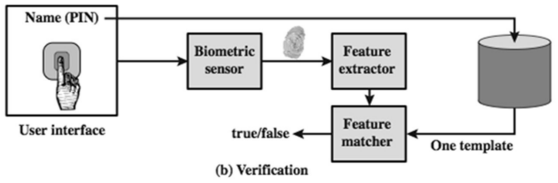


Operation of a Biometric System

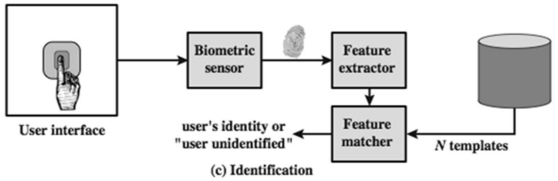
Enrollment is the registration. Biometric features are extracted and stored and associated with an Id.



Verification verifies if a biometric feature corresponds to a stored template associated with an Id. (e.g., a PIN).

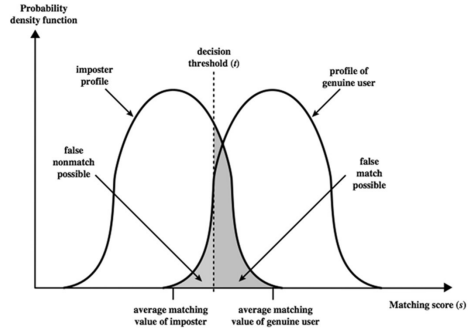


Identification is done with biometric info but no Id's. The system compares with a stored template and if founds one match it supplies the corresponding Id.



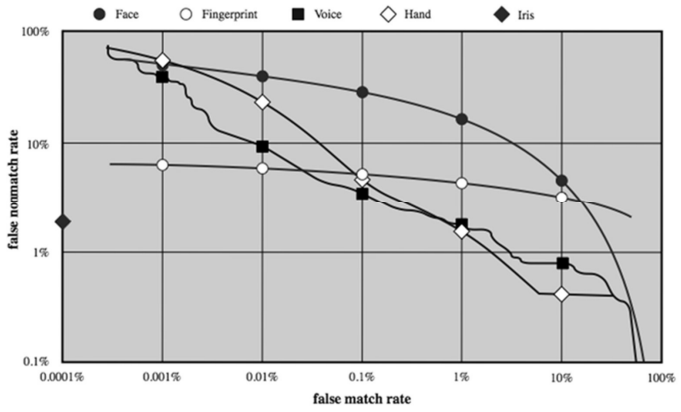
Biometric Accuracy (1)

- The system generates a matching score
  - The score quantifies similarity between the input and the closest stored template
- Concerns
  - Sensor noise produces almost always some deviations
  - Detection accuracy (acquisition, position, processing, …)
- Problems with false matches and false non-matches

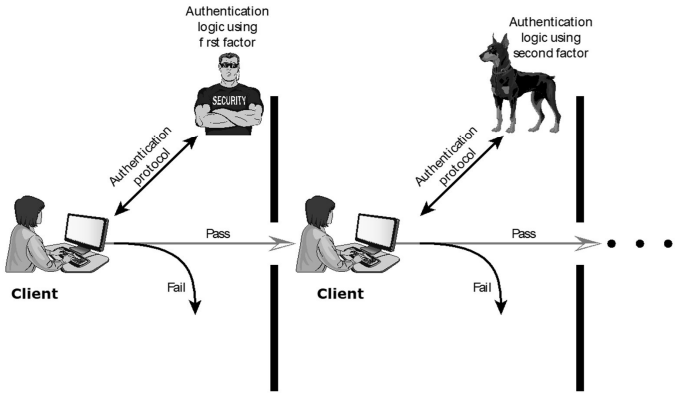


Biometric Accuracy (2)

- Choose a threshold minimizing false positives and negatives
  - Needs large experimentations for combinations of sensors and processing / extracting algorithms



Two-factor and Multi-factor



It's very common a second factor to be based on the possession of another communication channel (side-channel) with another device (smartphone)

Basic Remote User Authentication

- Authentication over a network requires more complexity
  - Should protect against eavesdropping and replay
- The main process should use a challenge / response protocol
  - User sends his identifier (represents the user identity)
  - Authenticator responds with a random message r (also known as a nonce)
  - User computes a value represented as f(r, h(P))
    - h() is an agreed upon cryptographic hash function
    - f() is another agreed upon function that can combine the value r and the hash h(P), where P is the user password
  - User sends the computed value to the host
  - The host computes the same value using r and the stored hash of the password (h(P))
  - The result is positive if there is a match (user authenticated)

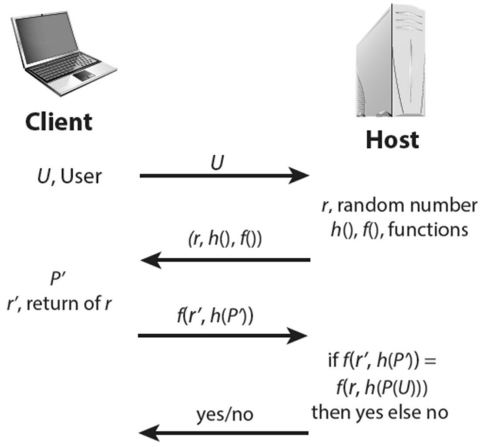
Remote Password Protocol

U – user identifier
r – nonce (random value (unique))
f(), h() – identifiers of the functions or implicitly agreed upon by both parties

f(r', h(P')) – computed by client with the received r (r' ← r) and the user supplied password (P')

f(r, h(P)) – computed by host using the transmitted value r and the stored hash of the user U password

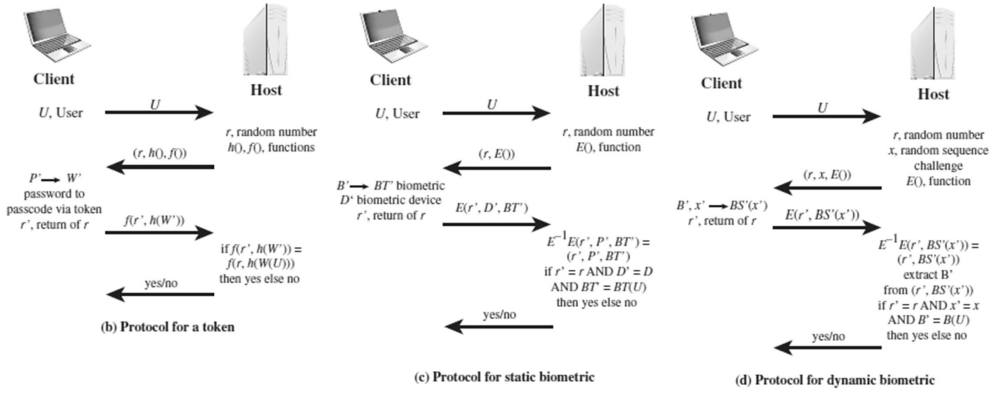
if previous values are equal than yes
else no



based on the old CHAP* standard (RFC 1994, from 1996)
where h(P(U)) is the stored secret, and f() is a hash function (initially MD5)
* Challenge Handshake Authentication Protocol

CHAP can be replaced by more secure protocols like SCRAM (Salted Challenge Response Authentication Mechanism), RFC 5802, 7677, 7804
SCRAM stores on the server a salt, and a hash of a HMAC, using the salted password as a key, depending on the user U. It allows also the host verification by the client

Protocols for Other Authentication Types



W' – passcode from password
h(W(U)) – stored passcode hash derived from the password

BT' – biometric template derived from the acquired biometrics B' at the client side
D' – identifier of the biometric acquisition device
E() is an identified or agreed encryption function (E^{-1}() – decryption)
BT(U) is the stored biometric template belonging to user U

x – random sequence of characters or words
BS'(x') – biometric signal generated from vocalization, typing or writing the sequence x'
B' – biometric characteristics extracted from the signal BS'(x')
B(U) – stored biometric characteristics of user U

Authentication Security Issues (1)

- **Client attacks**
  - attacker attempts to achieve user authentication without access to remote authenticator
    - Masquerade as a legitimate user (guess the password or try many)
  - Countermeasures: strong passwords; limit on the number of wrong attempts
- **Host attacks**
  - Attackers try to get the stored password file in the host
  - Countermeasures: password hashing; increased protection on password database
- **Eavesdropping**
  - attacker attempts to observe the user and transmissions: find written passwords; keylogging; network interception
  - Countermeasures: keep password secret and user memorized; multifactor authentication; quick revocation of compromised passwords

Authentication Security Issues (2)

- **Replay**
  - Attacker tries to repeat a previously captured user response
  - Countermeasures: use of challenge / response; generating 1-time passwords
- **Trojan horse**
  - an application or device masquerades as an authentic application or device
  - Countermeasures: authentication of clients should occur within trusted security environments
- **Denial of service**
  - Attacker attempts to disable the authentication service (e.g., by flooding)
  - Countermeasures: multifactor authentication with a fast verifiable token