

# Authentication

Segurança Informática em Redes e Sistemas  
2024/25

David R. Matos, Ricardo Chaves

Ack: Miguel Pardal, Miguel P. Correia, Carlos Ribeiro

# Roadmap

- Authentication
- Authentication with passwords
- Biometric authentication

# Roadmap

- **Authentication**
- Authentication with passwords
- Biometric authentication

# Authentication: Problems

- Services must only be used by authorized users
  - To do **authorization**, first services need to **authenticate the users**
- Authentication must be trustworthy
  - It must be proved that the connecting party is **who it claims to be**
- Authentication must be secure
  - The authentication protocol must **not** be used by a third party to **impersonate** legitimate users
- Authentication should be simple to use
  - So that people are able to deal with it without great **cost**
- Users can be careless
  - And choose poor passwords, for example

# Authentication protocols: Goals

- Entity authentication
  - People, services, servers, machines, etc.
- Facilitate protocol usage
  - Provide mechanisms to simplify use of personal secrets (keys, passwords)
  - Interconnection with key distribution protocols
- Assure protocol correctness/accuracy in hostile environments
  - Accuracy in the proof of authentication
  - Confidentiality of the used secrets as proof of authentication
- Prevent impersonation from attackers
  - Prevent attacks to the messages used in the on-line protocols
  - Prevent off-line attacks with dictionaries

# Machine Authentication

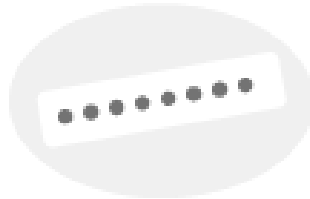
- **Bad option: by name or address**
  - Name, DNS, IP, MAC, or other
  - Unreliable, but used by several old services
    - e.g., NFS, TCP wrappers
  - Some validations are possible
    - e.g., translation DNS → IP → DNS
- **With secrets (keys)**
  - Shared key between machine pairs
  - Asymmetric key pair per machine
  - Methods used in:
    - Secure network protocols, e.g., IPSEC
    - Secure communication protocol that interact with a daemon representing a machine, for tunneling applications' communication, e.g., SSH

# Services Authentication

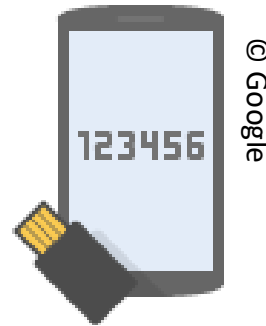
- **Machine authentication**
  - Allows to univocally authenticate a set of services running in that machine
- **Service authentication** uses similar approaches to machine authentication but only for a specific service running on the machine
  - e.g., Web Server
- **Shared secret key**
  - Sharing a long-term secret with the user
  - e.g., remote login with password
- **Asymmetric key pair per service**
  - Allows to differentiate services in a more defined way
  - e.g., HTTPS servers
    - with private key corresponding to public key in digital certificate

# Personal Authentication

Using something that  
**you know**



Using something that  
**you have**



© Google



Using something that  
**you are**



# Personal authentication methods

- Using something that **you know**
  - Typically, a shared secret
  - Password, PIN, etc.
- Using something that **you have**
  - Magnetic card, Smart Card, key generators, etc.
- Using a **physical characteristic**
  - Evaluation of a static unique personal feature:
    - Fingerprint, iris, retina, facial features, voice, etc.
- Using a **behavioral characteristic**
  - Evaluating unique dynamic features of a person
    - Writing patterns on a keyboard
    - Patterns of a signature

Biometric features

# Multi-Factor Authentication

- Use of several authentication methods combined
  - Designated **Double Factor Authentication** or **2FA** when two methods are used
- Can assign different methods to different tasks
  - As users perform more and more sensitive tasks, must authenticate in more and more ways (presumably, more stringently)
  - Example in home banking: password, 3 values from matrix, code sent by SMS

## **Resolução do Conselho de Ministros n.º 41/2018**

*Diário da República, 1.ª série—N.º 62—28 de março de 2018*

Recomenda-se que para novos sistemas seja sempre usado como padrão de autenticação o 2FA.

# Roadmap

- Authentication
- **Authentication with passwords**
- Biometric authentication

# Username/passwords

- Usernames/passwords can take different forms today
  - Email address
  - Phone number



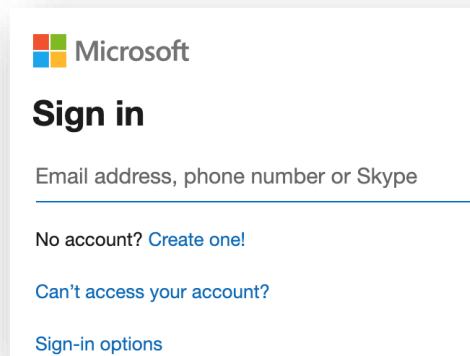
Google

Iniciar sessão

Continuar para o Gmail

Email ou telemóvel

[Esqueceu-se do email?](#)



Microsoft

**Sign in**

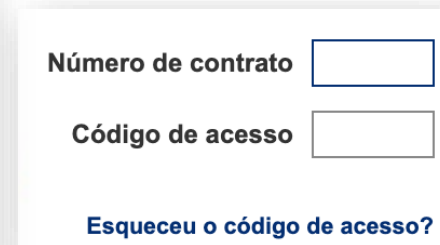
Email address, phone number or Skype

---

No account? [Create one!](#)

[Can't access your account?](#)

[Sign-in options](#)



Número de contrato

Código de acesso

[Esqueceu o código de acesso?](#)

# Authentication of people: with memorized passwords

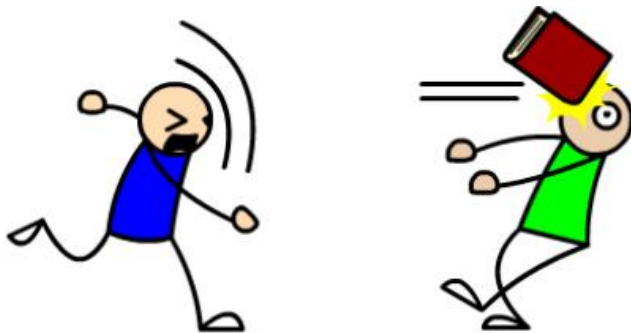
- Operation
  - The user memorizes a **password** and can provide it when prompted
  - Password **validation** relies on a **secret** stored on behalf of the user
    - The password value should never be stored
    - Instead, the personal secret is protected by being transformed by a **one-way function**, for example, using a **cryptographic hash function** or a **key derivation function** (*next*)
- **Advantages** of passwords
  - Simplicity
  - Can be used in any device with text input
- **Problems** of passwords
  - Poor selection by users
  - Remote transmission through unsecure channels

# Key derivation function (KDF)

- KDF transform a **variable-length text password** into a **fixed-length binary cryptographic key**
  - String -> byte[ ]
  - KDFs often use **non-secret parameter** – called **salt** – in addition to the **secret password** to produce diverse hash values across machines (*more on this later*)
  - KDF may ensure that derived keys have other desirable properties
    - e.g., avoiding “weak keys” in some specific encryption systems
- Common uses of KDFs are:
  - **Password hashing** – the one we are interested here
  - **Generating secret keys from passwords**  
e.g., to cipher documents

# Password attacks

- **Mislead a user** to reveal password
  - Pretexting, baiting, phishing and other “social engineering” attacks
- **Try to guess the password**
  - Brute-force – systematically test all possible combinations
  - **Dictionary attack** – test likely values first
    - Values that are easy for humans to memorize, like words in dictionary, dates, etc.



# Top 50 breached passwords

#	Windows Password	Number of Times Detected	#	Windows Password	Number of Times Detected
1	123456	23,174,662	26	myspace1	735,980
2	123456789	7,671,364	27	121212	732,832
3	qwerty	3,810,555	28	homelesspa	727,480
4	password	3,645,804	29	123qwe	711,669
5	111111	3,093,220	30	a123456	679,353
6	12345678	2,889,079	31	123abc	637,906
7	abc123	2,834,058	32	1q2w3e4r	631,071
8	1234567	2,484,157	33	qwe123	630,653
9	password1	2,401,761	34	7777777	623,994
10	12345	2,333,232	35	qwerty123	592,110
11	1234567890	2,224,432	36	target123	587,949
12	123123	2,194,818	37	tinkle	585,933
13	000000	1,942,768	38	987654321	585,426
14	iloveyou	1,593,388	39	qwerty1	581,151
15	1234	1,256,907	40	222222	579,444
16	1q2w3e4r5t	1,141,300	41	zxcvbnm	575,310
17	qwertyuiop	1,081,655	42	1g2w3e4r	573,735
18	123	1,023,001	43	gwerly	573,292
19	monkey	980,209	44	zag12wsx	572,800
20	dragon	968,625	45	gwerly123	572,625
21	123456a	968,369	46	555555	551,013
22	654321	932,752	47	fuou	549,663
23	123321	911,514	48	112233	534,650
24	666666	876,983	49	asdfghjkl	527,764
25	1qaz2wsx	756,613	50	1q2w3e	498,741

Source: <https://haveibeenpwned.com/Passwords>

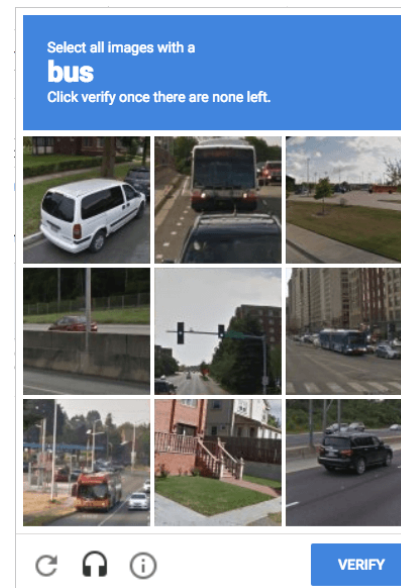
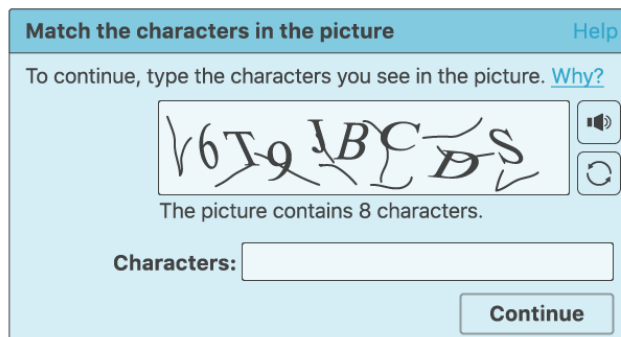


# Mitigations

- Against “social engineering”
  - Provide user education and training
- Against brute-force
  - Increase number of combinations
    - Ask for longer passwords, e.g., minimum 10 characters
    - Force use of diverse characters: letters, capital letters, numbers, symbols
  - Refresh passwords periodically
    - E.g., every 6 months
    - Make sure new password is very different from old password
- Against dictionary attacks
  - Reject popular/trivial passwords
    - And variations with prefixes, suffixes, replacements of letters by numbers, and other ways that users try to use weak passwords

# Online password attacks

- The attacker tries to guess the password, **one attempt at a time**, through the user interface
- Mitigation: limit the number of attempts
  - **Throttling**, e.g., limit to N attempts (per minute)
  - “*Human verification*” with challenges like CAPTCHA



# Offline password attacks

- The attacker exfiltrated the **password file** and can try **many guesses** rapidly
  - Throttling and CAPTCHA are circumvented
- Mitigation: store a **transformation** of the password, not its actual value
  - Use of cryptographic hash
  - Use of salt
  - Key stretching / strengthening



# Salt



- **Salt** – random data used as an additional input to a one-way function that hashes a password
  - Useful if attacker gains access to the saved password hash because they help defending from attacks that use:
- **Rainbow tables** (precomputed hash databases)
  - Tables that containing passwords and their hashes
    - Checking the hashes is fast: compare stolen hashes with the table
    - Without the need of calculating the hashes
  - Rainbow tables are expensive to compute
  - Without salt, attacker can build table once, then use it always
    - MD5 of "password" is always  
*5f4dcc3b5aa765d61d8327deb882cf99*
    - SHA1 of "password" is always  
*5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8*
  - With a salt, attacker has to compute a rainbow table for each salt value

# Key stretching / strengthening

- **Derived key = KDF(Key, Salt, Iterations)**
- Parameters:
  - **Key** – original key or **password** (password in the present case)
  - **Salt** – random non-secret number
  - **Iterations** – number of executions of a sub-function
    - The difficulty of a brute force attack increases with the number of iterations
    - Practical limit is the unwillingness of users to tolerate perceptible delay
- Modern password-based KDF: **PBKDF2** (RFC 2898)
  - Uses cryptographic hash function (e.g., SHA-2), salt (e.g., 64 bits or more) and a high iteration count (often tens or hundreds of thousands)
  - NIST requires at least 128 bits of random salt and a NIST-approved cryptographic function, such as the SHA or AES
- 2013: Password Hashing Competition
  - Choose a new standard algorithm for password hashing: Argon2

# Historical examples: Unix Crypt

- **Passwords** with 8 7-bit ASCII characters
  - **Key** =  $8 \times 7 = 56$  bits (password)
- 25 **iterations** of modified DES algorithm: **DES<sub>Key</sub>(0)**
  - Modified DES to avoid hardware attacks that existed at the time
- **Salt** of 12 bits
  - Additional permutation in DES' interactions depending on the salt
  - Stored in cleartext next to the password
- **Encoding** as printable string to be stored in the `/etc/passwd` file
  - 64 bits resulting from cipher of "0" + 2 bits = 66 bits (2 bits for padding)
  - Encoded as 11 base64 digits

too small today

# Historical examples: Windows NT

- Two hashes for compatibility reasons: NT and LanMan
- NT hash **NT-Hash** = MD4(pwd) (MD4 is an old hash function; insecure)
  - Weak: no salt, no iterations
- LanMan (LM) hash
  - PWD=CutOrPad(Uppercase(password), 14) – 14 characters
  - L = Left(PWD); R = Right(PWD) – 7 characters each
  - **LM-Hash** = DES<sub>L</sub>("KGS!@#%") | DES<sub>R</sub>("KGS!@#%") – 16-byte hash
  - Optionally **Hash** = DES<sub>ID</sub>(Hash)
- NTv2 and LMv2 (since Windows NT 4.0 SP4, 1998)
  - SC = 8-byte server challenge, random
  - CC = 8-byte client challenge, random
  - CC\* = (X, time, CC, domain name) (X is a constant)
  - v2-Hash = HMAC-MD5(NT-Hash, user name, domain name)
  - **LMv2** = HMAC-MD5(v2-Hash, SC, CC)
  - **NTv2** = HMAC-MD5(v2-Hash, SC, CC\*)

# LanMan Security Weaknesses

- Passwords are limited to a maximum of only 14 characters
  - Theoretical maximum keyspace of  $95^{14} \approx 2^{92}$
  - But passwords longer than 7 characters are divided into two pieces and each piece is hashed separately:  $95^7 \approx 2^{46}$
  - Password lowercase letters are changed into uppercase, which further reduces the key space for each half:  $69^7 \approx 2^{43}$
  - In addition, any password that is shorter than 8 characters will result in the hashing of 7 null bytes ( $L = 0$ ), yielding a constant value
- LM hash does not use salt
- Ophcrack tool (2003) had rainbow table that targets the weaknesses of LM
  - Can crack all alphanumeric LM hashes in a few seconds



# Roadmap

- Authentication
- Authentication with passwords
  - **One-time passwords**
- Biometric authentication

# Authentication of people: one-time passwords (OTP)

- Operation
  - Validation of a **one-time password** (i.e., single use) regarding a secret stored for the user
  - A one-time password can only be used once
    - Therefore, it is pointless for an attacker to capture it
  - Which one-time password to use can be known by several means
    - Challenge-response
    - Synchronization
      - Generation every 30 to 60 seconds
      - Requiring a synchronization server
      - Typically, also has a secret

# Assessment of one-time passwords

- Advantages

- Security in remote authentications over insecure networks
- Security if used in insecure terminal (e.g., with a key logger running)

- Problems

- In some implementations the users need to use some device/application to generate the one-time passwords
  - Dedicated equipment (smartcard, etc.)
  - Computation algorithms and functions
  - Not much appreciated by the users
- Memorization of one-time passwords by the attackers
  - Off-line attacks with dictionaries using collected keys
  - Dictionary attacks can be prevented if non-textual seed-passwords are used

# Roadmap

- Authentication
- Authentication with passwords
  - **Challenge-response**
- Biometric authentication

# Authentication of people: challenge-response

- Operation
  - **Authenticator** provides a value: **challenge**
  - **User** being authenticated transforms the challenge using the **secret** that he shares with the authenticator
  - **User** sends the result to the authenticator: **response**
  - **Authenticator** validates the response / transformation
- Advantages
  - Security for authentication over unsecure networks
- Problems
  - The authenticator needs to have access to shared secrets with all users
  - Attackers may store the challenge-response pairs exchanged
  - Offline attacks with dictionaries to the collected pairs

# Roadmap

- Authentication
- Authentication with passwords
- **Biometric authentication**

# Authentication of people: biometric authentication

- Operation
  - User validated through biometrical data: fingerprint, physiognomy,...
  - This info is compared with records for each user recognized by the system
- Advantages
  - Solves the problem of choosing a good password
  - Eliminates the need for the users' memory
- Problems
  - Sometimes can be deceived
  - Do not allow the transference of authentication between subjects
  - Do not support change of authentication data (it is fixed)
  - Complicates secure remote authentication

# Identification vs. Authentication

- **Identification**

- Search for an entity of a community given an identifying feature
  - Example: searching for person in a criminal record given his fingerprint
- *Open* search

- **Authentication** (or verification)

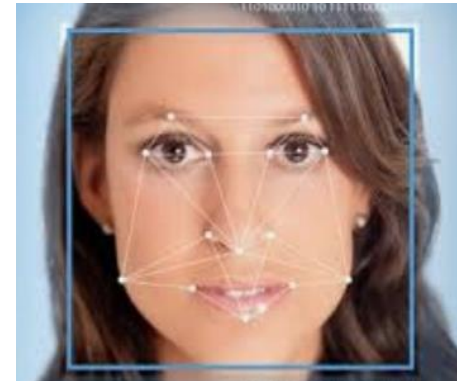
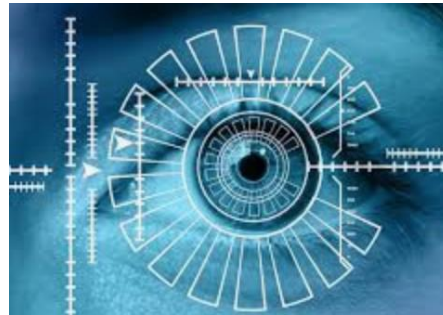
- Verifying that an entity is who it claims to be (the id is given)
  - Example: verify in an access control device if the fingerprint presented by a person matches the one registered for that same person
- *Closed* search





# Examples of biometric technologies

- Fingerprints
- Iris recognition
- Retina scan
- Facial recognition
- Palm vein biometrics
- Voice recognition



# Working principle of biometric authentication

- **Registration**

- Acquisition of information about the person
  - Biometric data and other information
- Construction of the validation information: the **template**
  - Typically from the template it is not possible to reconstruct the biometric data

- **Authentication**

- By providing the biometric information along with the claimed identity
- Checking the obtained biometric data with the stored template

- Authentication in this context is unidirectional

- A person proves to be who he claims
- The authentication system does not prove its authenticity
  - It either accepts the individual authentication or not

# Biometric authentication: desirable properties

- Universality
  - Ability to be applied to all individuals
- Unicity
  - Ability to distinguish all the individuals
- Stability
  - Ability to operate continuously without problems during lifetime of the individuals
- Correctness
  - Ability to acquire and use validation data capable of distinguishing all individuals
  - related to Unicity, but Correctness is “practical”, Unicity is “theoretical”
- Convenience
  - Ability not to cause discomfort or repulsion
- Acceptance
  - Ability not to cause rejection due to loss of privacy or ethical-social issues

Exercise: how do fingerprinting and face recognition stand in each property?

# Biometric authentication:

## Advantages & disadvantages (1/2)

- Biometric characteristics cannot be lost, forgotten, or lost
  - But also cannot be modified if needed
  - Biometric authentication may abuse **privacy** principles
- Biometric authentication requires physical presence
  - This complicates certain actions:
    - Delegation of responsibilities
    - Remote authentication
  - May cause discomfort or rejection
    - e.g. exposing the retina to lasers
- Dictionary attacks are not possible
  - But typically a shared secret key cannot be derived from the authenticator – validation is **probabilistic**

# Biometric authentication:

## Advantages & disadvantages (2/2)

- The likelihood of 2 individuals with the same biometric features existing is low
  - But the biometric **acquisition mechanisms** and the **template matching** processes may not be able to distinguish the two
- Allow for maintenance cost reduction in password-based systems
  - However, if they fail, alternative authentication mechanisms are required
    - Typically passwords
  - Which is even worst:
    - The initial goal of eliminating the need for password is not achieved!
    - The system becomes even more complex
    - People must remember passwords that are rarely used

# Technical aspects: Correction

- Making no errors; 2 types of errors:
  - **False Acceptance / False Positive (FP)**
    - Accepting incorrect biometric data
  - **False Rejection / False Negative (FN)**
    - Not accepting true biometric data
- **Accuracy:**  $(TP + TN) / (TP + TN + FP + FN)$ 
  - The greater the accuracy the greater the distinction between individuals
  - Can be critical for identification
  - Typically, not so critical for authentication
- Error rate / accuracy adjustment
  - Allows to select acceptable error rates considering the operational environment

# Adjustment of error rates

- The reduction of one error rate increases the other
  - The reduction of the **False Rejection Rate (FRR)** is associated to an increase of the **False Acceptance Rate (FAR)** and vice-versa
- The error rate depends on the stored validation information (template)

