Authentication Mechanisms and Protocols

SIO

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

André Zúquete

Authentication (Authn)

Proof that an entity has an attribute it claims to have

– Hi, I’m Joe
– Prove it!
– Here is my proof, calculated with Joe’s credentials that I’ve agreed with you
– Proof accepted/not accepted

– Hi, I’m over 18
– Prove it!
– Here is a claim issued by a competent authority, which I can also prove that I’m the owner
– Proof and claim accepted/not accepted

Authn: Proof Types

- Something we know
  - A secret memorized (or written down…) by Joe
- Something we have
  - An object/token solely held by Joe
- Something we are
  - Joe's Biometry

Multi-factor authentication

- Use of several, different proof types
- 2FA = Two Factor Authentication

Risk-based MFA

- Variable MFA
- Higher attack risk, more factors or less risky factors
- Lower attack risk, less or easier factors

Authn : Goals

- Authenticate interactors
  - People, services, servers, hosts, networks, etc.
- Enable the enforcement of authorization policies and mechanisms
  - Authorization ≠ authentication
  - Authorization requires authentication
- Facilitate the exploitation of other security-related protocols
  - e.g. key distribution for secure communication

Authn : Requirements

- Trustworthiness
  - How good is it in proving the identity of an entity?
  - How difficult is it to be deceived?
  - Level of Assurance (LoA)

- Secrecy
  - No disclosure of secret credentials used by legit entities

LoA by NIST 800-63

LoA	DESCRIPTION	TECHNICAL REQUIREMENTS		
		IDENTITY PROOFING REQUIREMENTS	TOKEN (SECRET) REQUIREMENTS	AUTHENTICATION PROTECTION MECHANISMS REQUIREMENTS
1	Little or no confidence exists in the asserted identity; usually self-asserted; essentially a persistent identifier	Requires no identity proofing	Allows any type of token including a simple PIN	Little effort to protect session from off-line attacks or eavesdropper is required.
2	Confidence exists that the asserted identity is accurate; used frequently for self service applications	Requires some identity proofing	Allows single-factor authentication. Passwords are the norm at this level.	On-line guessing, replay and eavesdropping attacks are prevented using FIPS 140-2 approved cryptographic techniques.
3	High confidence in the asserted identity's accuracy; used to access restricted data	Requires stringent identity proofing	**Multi-factor authentication**, typically a password or biometric factor used in combination with a 1) software token, 2) hardware token, or 3) one-time password device token	On-line guessing, replay, eavesdropper, impersonation and man-in-the-middle attack are prevented. Cryptography must be validated at FIPS 140-2 Level 1 overall with Level 2 validation for physical security.
4	Very high confidence in the asserted identity's accuracy; used to access highly restricted data.	Requires in-person registration	**Multi-factor authentication** with a hardware crypto token.	On-line guessing, replay, eavesdropper, impersonation, man-in-the-middle, and session hijacking attacks are prevented. Cryptography in the hardware token must be validated at FIPS 140-2 level 2 overall, with level 3 validation for physical security.

Authn : Requirements

- Robustness
  - Prevent attacks to the protocol data exchanges
  - Prevent on-line DoS attack scenarios
  - Prevent off-line dictionary attacks
- Simplicity
  - It should be as simple as possible to prevent entities from choosing dangerous shortcuts
- Deal with vulnerabilities introduced by people
  - They have a natural tendency to facilitate or to take shortcuts
  - Deal with phishing!

Authn: Entities and deployment model

Entities

- People
- Hosts
- Networks
- Services/servers

Deployment model

- Along the time
  - Only when the interaction starts
  - Continuously along the interaction
- Directionality
  - Unidirectional
  - Bidirectional (mutual)

Authn interactions: Basic approaches

- Direct approach
  1. Provide credentials
  2. Wait for verdict

  – Advantage: no computations by the presenter
  – Disadvantage: credentials can be exposed to malicious validators

- Challenge-response approach
  1. Get challenge
  2. Provide a response computed from the challenge and the credentials
  3. Wait for verdict

  – Advantage: credentials are not exposed to malicious validators
  – Disadvantage: requires computations by the presenter

Authn of subjects: Direct approach w/ known password

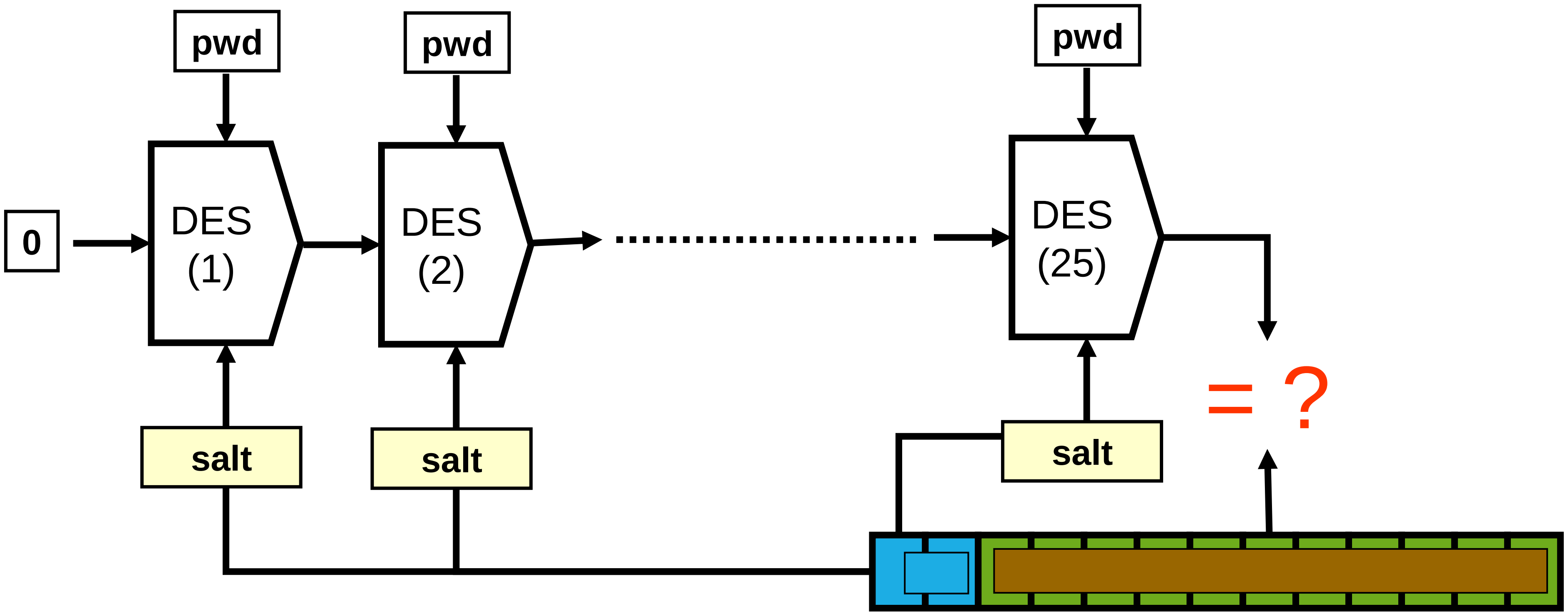
- A password is checked against a value previously stored
  - For a claimed identity (username)

- Personal stored value:
  - Transformed by a unidirectional function
  - Windows: digest function
  - UNIX: DES hash + salt
  - Linux: MD5 + salt
    - hash is configurable

Optimal scenario

- Complex, slow password transformations
- PBKDF2, Script with high complexity

Authn of subjects: Direct approach w/ known password



$\text{DES hash} = \text{DES}_{pwd}^{25}(0)$

$\text{DES}_k^n(x) = \text{DES}_k(\text{DES}_k^{n-1}(x))$

Permutation of 12 subkeys' bit pairs with salt (12 bits)

Authn of subjects: Direct approach w/ known password

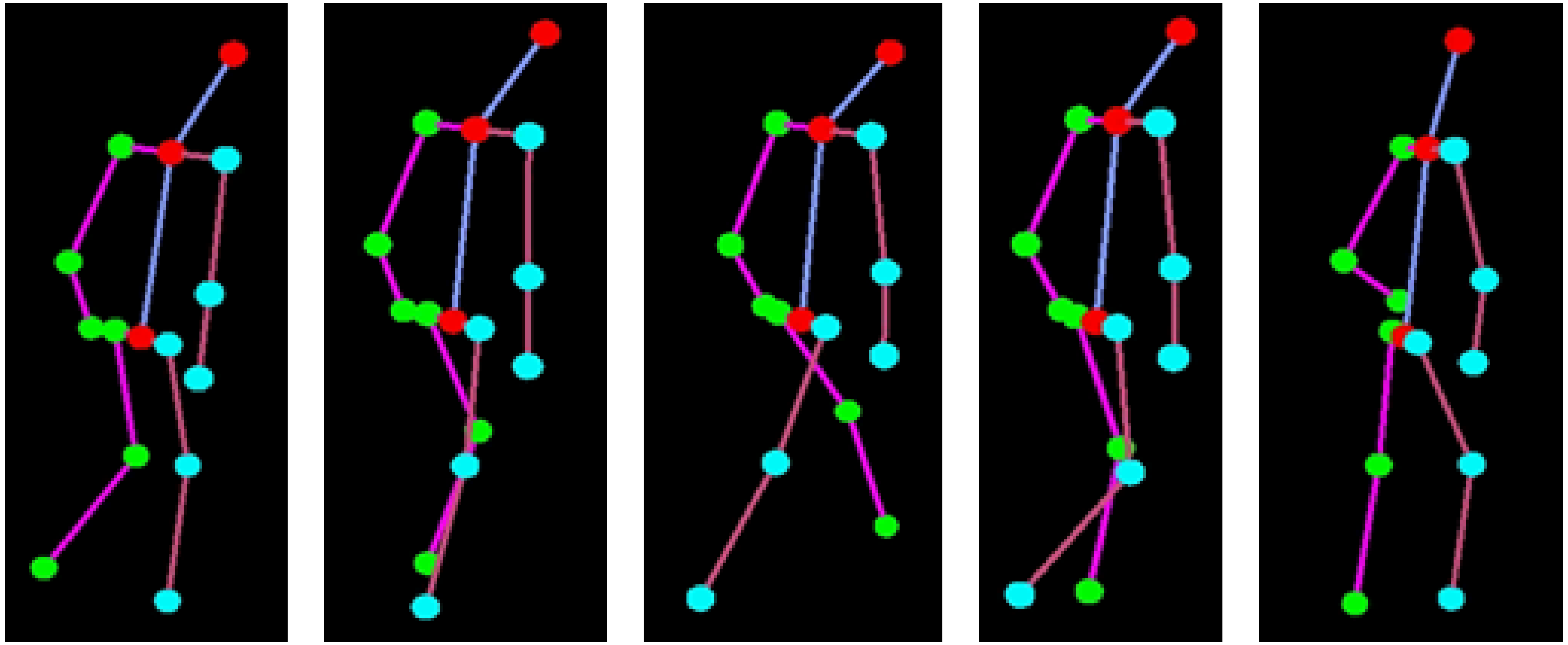
- Advantage
  - Simplicity!
- Problems
  - Usage of weak passwords
    - Enable dictionary attacks
  - Transmission of passwords along insecure communication channels
    - Eavesdroppers can easily learn the password
    - e.g. Unix remote services, PAP

Top Ten 2017 from Splashdata

1. 123456
2. Password
3. 12345678
4. qwerty
5. 12345
6. 123456789
7. letmein
8. 1234567
9. football
10. iloveyou

Authn of people: Direct approach with biometrics

- People get authenticated using body measures
  - Biometric samples
  - Fingerprint, iris, face geometry, voice timber, manual writing, vein matching, etc.
- Measures are compared with personal records
  - Biometric references (or template)
  - Registered in the system with a previous enrolment procedure
- Identification vs authentication
  - Identification: 1-to-many check for a match
  - Authentication: 1-to-1 check for a match



Authn of people: Direct approach with biometrics

- Advantages
  - People do not need to use memory, or carry something
  - Just be their self
- People cannot choose weak passwords
  - In fact, they don’t choose anything
- Authentication credentials cannot be transferred to others
  - One cannot delegate its own authentication

Authentication of people: Direct approach with biometrics

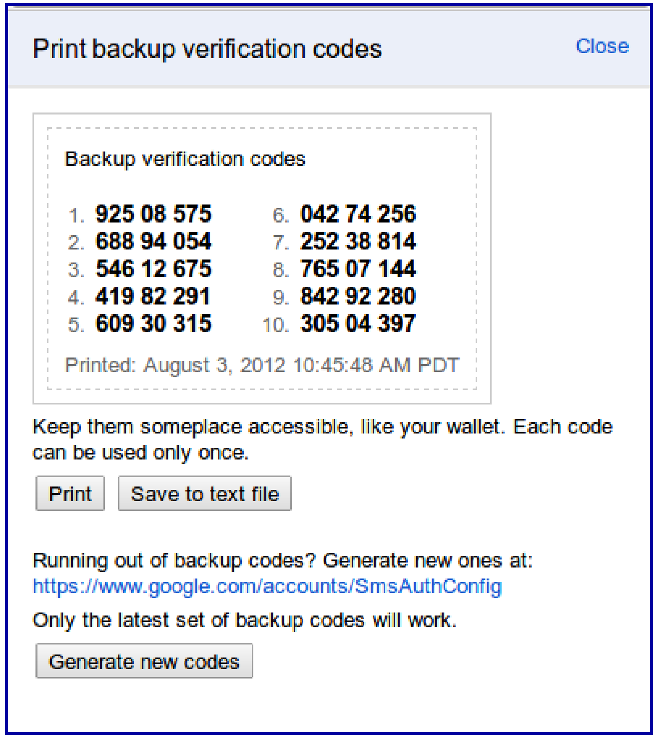
- Problems
  - Biometric methods are still incipient
    - In many cases it can be fooled with ease (Face Recognition, Fingerprint)
  - People cannot change credentials
    - If the credentials or templates are stolen
  - Credentials cannot be transferred between individuals
    - If it is required in extraordinary scenarios
  - Can pose risks to individuals
    - Physical integrity can be compromised by an attacker in order to acquire biometric data
  - It is not easy to be implemented in remote systems
    - It is mandatory to have secure and trusted biometric acquisition devices
  - Biometrics can reveal other personal secrets
    - Diseases

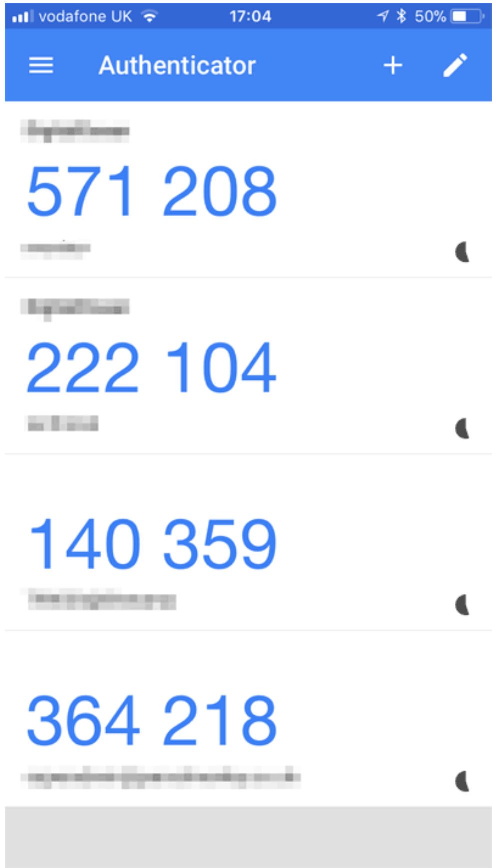
Authn of subjects: Direct approach w/ one-time passwords

- One-Time Passwords = Secrets that can be used only once
  - Pre-distributed directly, or the result of a generator function

- Example: Bank codes, Google Backup Codes

https://www.montepio.pt/SitePublico/pt_PT/particulares/montepio24/cartao-matriz.page?altcode=10006P





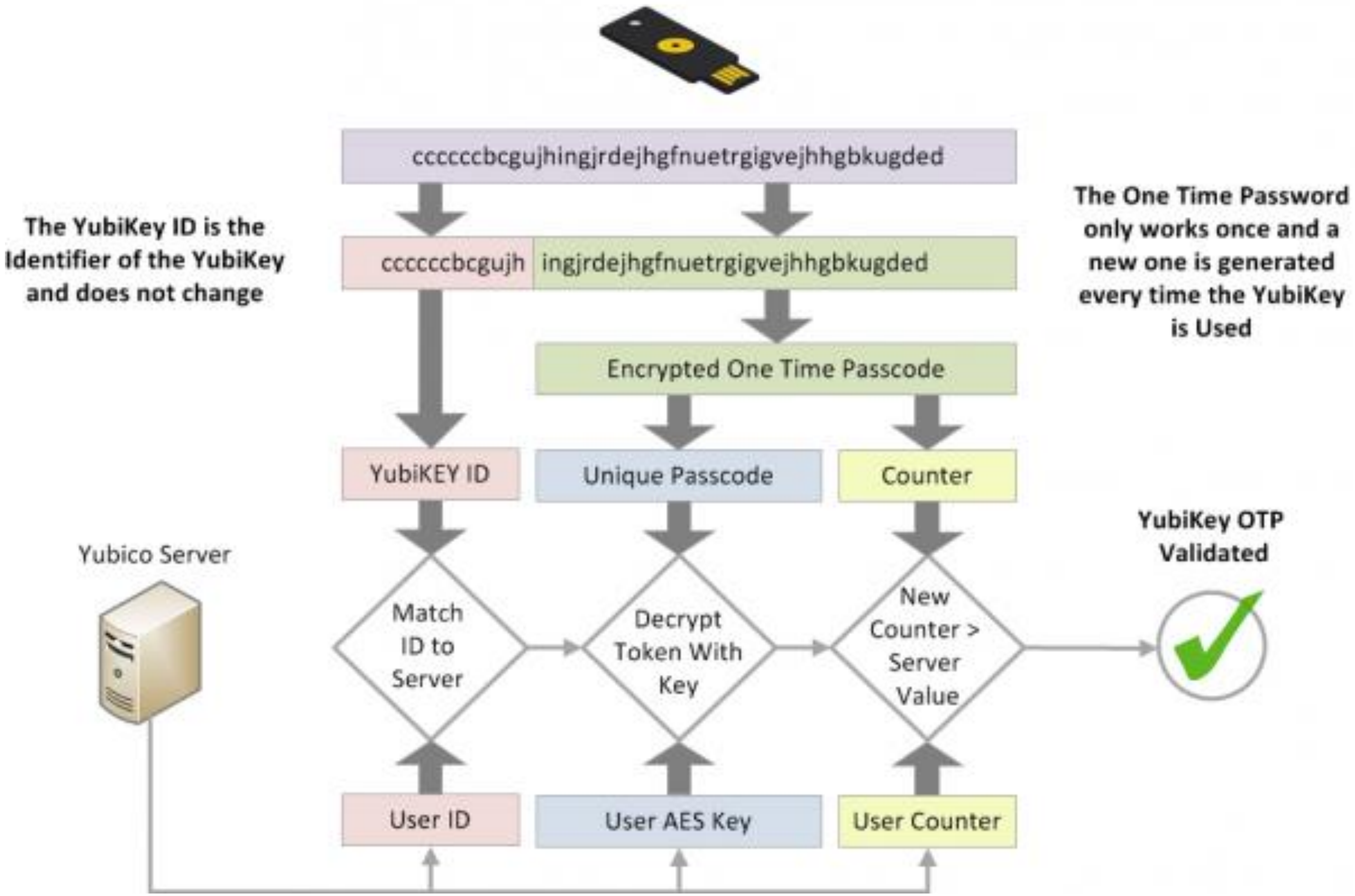
Authn of subjects: Direct approach w/ one-time passwords

- Advantages
  - Can be eavesdropped, allowing its use in channels without encryption
  - Can be chosen by the authenticator, which may enforce a given complexity
  - Can depend on a shared password
- Problems
  - Interacting entities need to know which password to use on each occasion
    - Implies some form of synchronization (e.g., index, coordinates)
  - Individuals may require additional resources to store/generate the passwords
    - Sheet of paper, application, additional device, etc.

Yubikey

- Personal Authentication Device
  - USB, Bluetooth and/or NFC

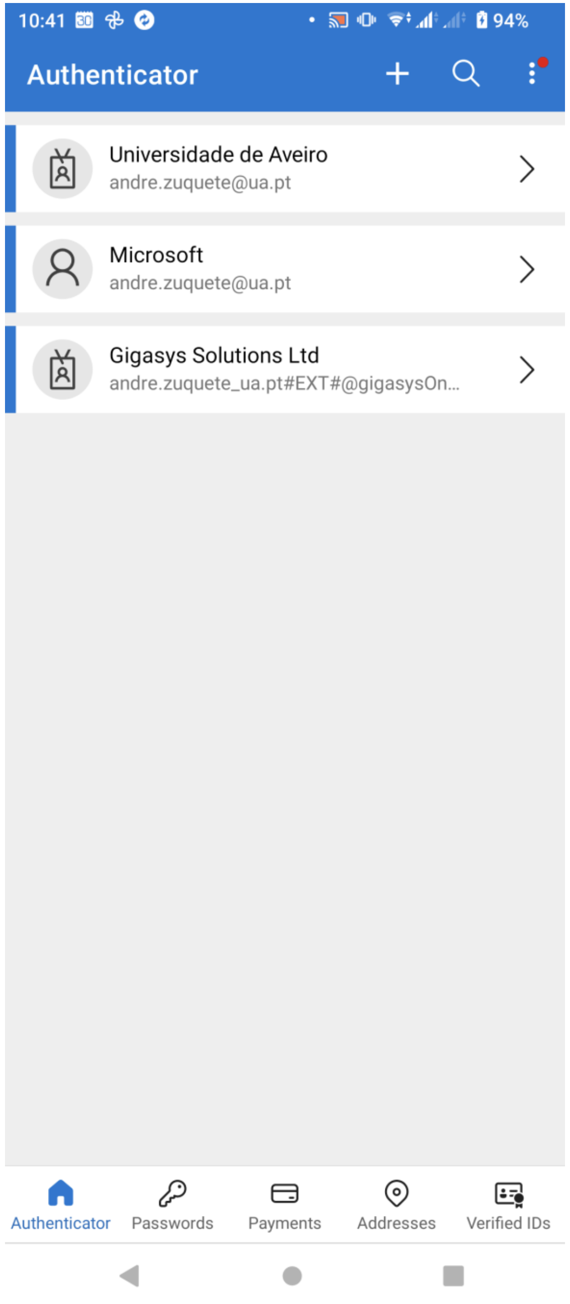
- Activation generates a 44 characters key
  - Emulates a USB keyboard (besides an own API)
  - Supports HOTP (events) or TOPT (Temporal)
  - If a challenge is provided, user must touch the button to obtain a result
  - Several algorithms, including AES 256

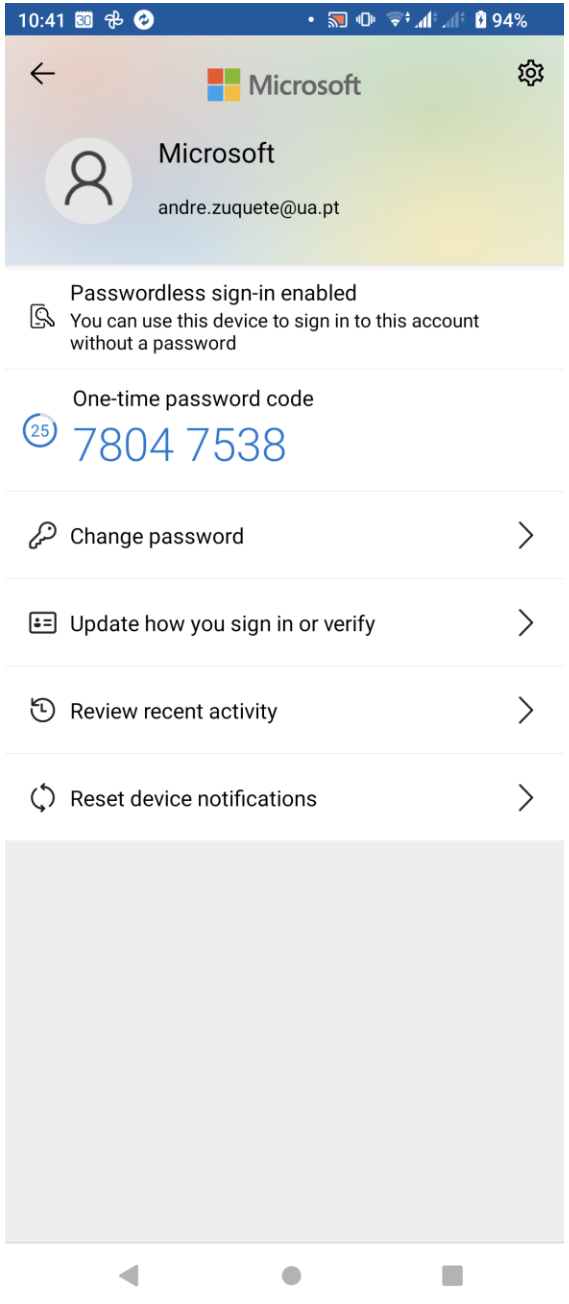

cccccbcgujhingjrdejhgfnuetrgigvejhhgbkugded
The YubiKey ID is the Identifier of the YubiKey and does not change
cccccbcgujh
ingjrdejhgfnuetrgigvejhhgbkugded
The One Time Password only works once and a new one is generated every time the YubiKey is Used
Encrypted One Time Passcode
YubiKEY ID
Unique Passcode
Counter
YubiKey OTP Validated
Yubico Server
Match ID to Server
Decrypt Token With Key
New Counter > Server Value
User ID
User AES Key
User Counter

HOTP / TOTP

- HOTP (HMAC-based One-Time Password)
  - Counter-based HMAC
  - Result is converted to human-readable text
  - Counter's desynchronization is an issue

- TOTP (Time-based one Time Password)
  - HOTP using timestamps instead of counters
  - Time synchronization is fundamental





Challenge-Response approach

- The authenticator provides a challenge
  - A nonce (value not once used)
  - Usually random
  - Can be a counter
- The authenticated entity transforms the challenge
  - The transformation method is shared with the authenticator
  - The result is sent to the authenticator
- The authenticator verifies the result
- Calculates a result using the same method and challenge
  - Or produces a value from the result and evaluates if it is equal to the challenge, or to some related value

Challenge-Response approach

- Advantages
  - Authentication credentials are not exposed
  - An eavesdropper will see the challenge and the result
    - but has no knowledge about the transformation

- Problems
  - Authenticated entities must have the capability of calculating results to challenges
    - Hardware token or software application
  - The authenticator may need to keep shared secrets (in clear text)
    - Secrets can be stolen
    - Individuals may reuse secrets in other systems, enabling lateral attacks
  - May be possible to calculate all results to a single (or all) challenge(s)
    - Can revel the secret used
  - May be vulnerable to dictionary attacks
    - Authenticator should NEVER issue the same challenge to the same user

Authn of Subjects: Challenge-Response with Smartcards

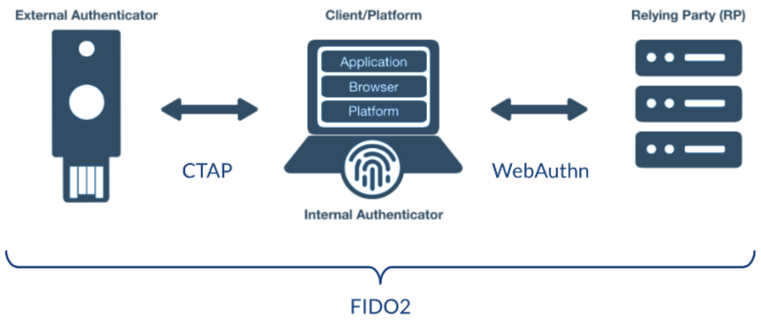
- Authentication Credentials
  - Having the smartcard (e.g., the Citizen Card)
  - The private key stored inside the smartcard
  - The PIN code to access the key
- The authenticator knows
  - The user public key
- Robust against:
  - Dictionary attacks
  - Offline attacks to the database
  - Insecure channels

Authn of Subjects: Challenge-Response with Smartcards

- Challenge-Response Protocol
  - The authenticator generates a challenge
  - Smartcard owner ciphers the challenge with their private key
    - Stored in the smartcard, protected by the PIN code
    - In alternative, can sign the challenge
  - The authenticator deciphers the result with the public key
    - If the decrypted result matches the challenge, the authentication is successful
    - In alternative, it can verify the signature (which is the same process)

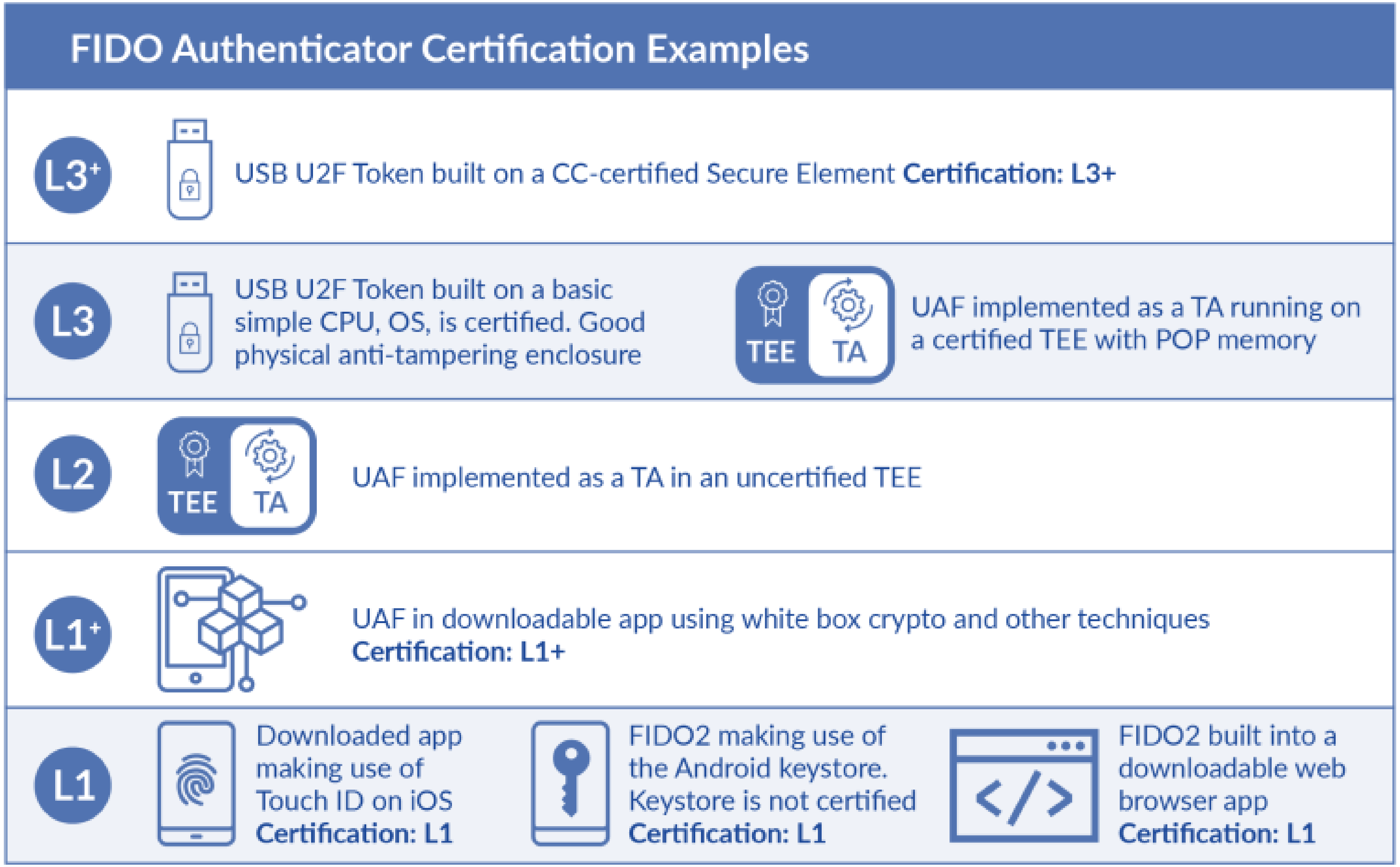
Authn of Subjects: Challenge-Response with other tokens

- FIDO2 tokens (FIDO Alliance)
  - For both mobile and desktop environments
  - Web Authentication (WebAuthn) specification
  - Client-to-Authenticator Protocol (CTAP)
  - Security
    - Credentials never leave the user's device and are never stored on a server
    - No risks of phishing, no password theft (still, tokens can be stolen)
    - No replay attacks
    - Token certification levels
  - Privacy
    - Credentials are unique per website
    - Tracking is not possible (different web sites, different public keys for the same token)
    - Biometric data, when used, never leaves the user's device



https://www.inovex.de/de/blog/fido2-webauthn-in-practice/

FIDO2 certification



Authn of Subjects: Challenge-Response with Shared Secret

- Authentication Credentials
  - Password selected by the individual
- The authenticator knows:
  - Bad approach: the shared password
  - Better approach: A transformation of the shared password
    - The transformation should be unidirectional

Authn of Subjects: Challenge-Response with Shared Secret

- Basic Challenge-Response Protocol
  - The authenticator generates a challenge

  - The individual calculates a transformation of the challenge and the password
    - result = hash(challenge || password)
    - or... result = encrypt(challenge, password)

  - The authenticator reverts the process and checks if the values match
    - result == hash(challenge || password)
    - or .... challenge == decrypt(result, password)

  - Examples with shared passwords: CHAP, MS-CHAP v1/v2, S/Key
  - Examples with shared keys: SIM & USIM (cellular communications)

PAP and CHAP (RFC 1334, 1992, RFC 1994, 1996)

- Protocols user for PPP (Point-to-Point Protocol)
  - Unidirectional authentication
    - The authenticator authenticates users, but users do not authenticate the authenticator

- PAP (PPP Authentication Protocol)
  - Simple presentation of a UID/password pair
  - Insecure transmission (in clear text)

- CHAP (CHallenge-response Authentication Protocol)

  Aut → U : authID, challenge

  U → Aut: authID, MD5(authID, secret, challenge), identity

  Aut → U : authID, OK/not OK

  - The authenticator can request further authentication at any time

Authn of subjects: Challenge-Response with Shared Key

- Uses a cryptographic key instead of a password
  - Robust against dictionary attacks
  - ..but requires a device to store the shared key

GSM Subscriber authentication

- Uses a secret shared between the HLR and the subscriber phone
  - Uses 128-bit shared key (not an asymmetric key pair)
  - Key is stored in the SIM card
  - SIM card is unlocked by a user PIN
  - SIM card answers challenges using the shared key
- Uses (initially unknown algorithms):
  - A3 for authentication
  - A8 to generate the session key
  - A5 is a stream cipher for communication
- A3 and A8 executed by the SIM, A5 executed by the baseband
  - A3 and A8 can be chosen by the operator

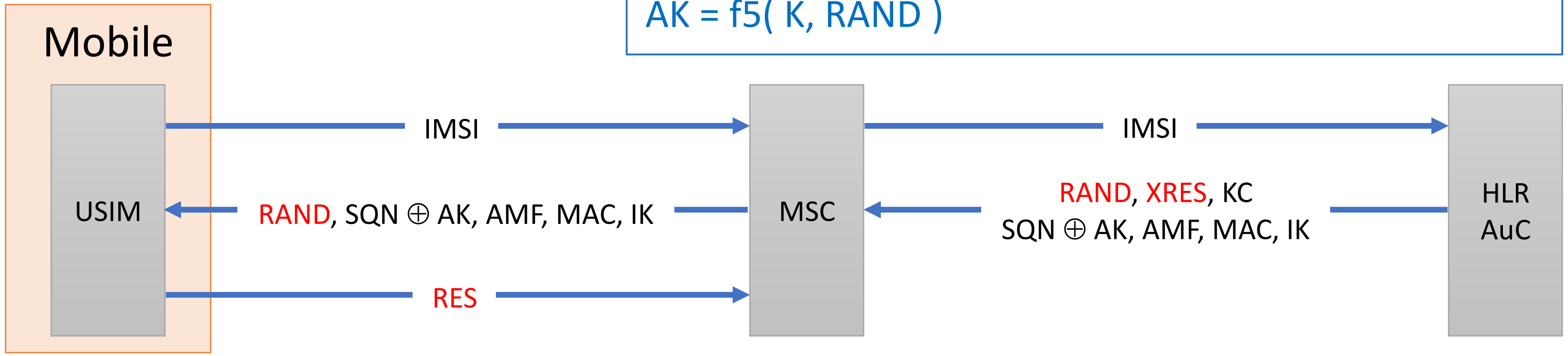
GSM/UMTS Subscriber authentication

- MSC requests authentication data from HLR/AUC
  - Given the USIM IMSI (Int. Mobile Subscriber Identifier)
  - HLR/AuC generates RAND and related data from K
  - K, RAND → XRES, KC, SQN ⊕ AK, AMF, MAC, IK
- MSC propagates some items to mobile
  - RAND, SQN ⊕ AK, AMF, MAC
- USIM validates items and creates response & secrets
  - K, RAND → AK → SQN → MAC
  - SQN higher than its own, SQN++
  - K, RAND → RES, CK, IK
- Mobile sends RES
  - That MSC checks against XRES

K - Subscriber authentication key (128 bits)

RAND - Authentication challenge (128 bits)
SQN - Sequence number (48 bits)
AMF - Authentication Management Field (16 bits)
MAC - Message Authentication Code (64 bits)
(X)RES - (eXpected) subscriber RESponse (32-128 bits)
CK – Cipher Key
IK – Integrity Key (128 bits)
AK – Anonymity Key

MAC = f1(K, SQN || RAND || AMF)
(X)RES = f2(K, RAND)
CK = f3(K, RAND)
IK = f4(K, RAND)
AK = f5(K, RAND)



Authentication of Systems

- By name (DNS) or MAC/IP address
  - Extremely weak, without cryptographic proof
  - Still... it is used by some services
  - e.g., NFS, TCP wrappers

- With cryptographic keys
  - Secret keys, shared between entities that communicate frequently
  - Asymmetric key pairs, one per host
  - Public keys pre-shared with entities that communicate frequently
  - Public keys certified by a third party (a CA)

Authentication of Services

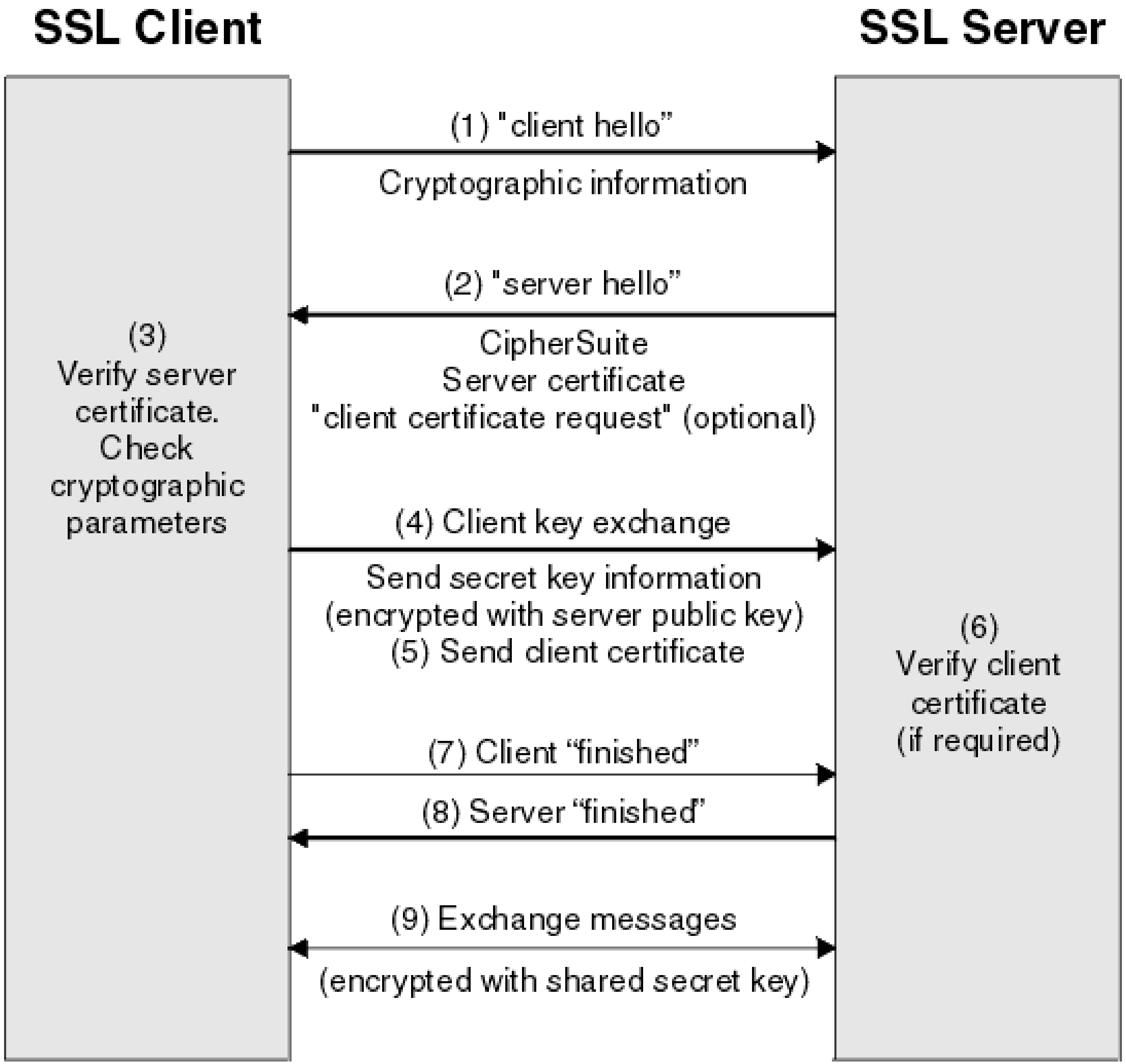
- Authentication of the host
  - All services co-located in the same host are automatically and indirectly authenticated
- Credentials exclusive to each service
- Authentication:
  - Secret keys shared with clients
    - When they require authentication of the clients (e.g. MS-CHAP V2, RFC 2759)
  - Asymmetric key pairs by host/service
    - Certified by others or not

TLS (Transport Layer Security, RFC 8446)

- Secure Communication Protocol over TCP/IP
  - Evolved from the SSL V3 (Secure Sockets Layer) standard
  - Manages secure sessions over TCP/IP, individual to each application
  - Initially designed for HTTP traffic
    - Currently used for many other types of traffic

- Security mechanisms
  - Confidentiality and integrity of the communication between entities
    - Key distribution, negotiation of ciphers, digests and other mechanisms

- Authentication of the intervenient entities
  - Servers, services, etc... (normal, but may be disabled)
  - Clients (not so common)
  - Both executed with asymmetric keys (not common) and X.509 certificates (common)


SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3)
Verify server
certificate.
Check
cryptographic
parameters
(4) Client key exchange
Send secret key information
(encrypted with server public key)
(5) Send client certificate
(6)
Verify client
certificate
(if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

TLS Ciphersuites

- If a server supports a single algorithm, it cannot be expected for all clients to also support it
  - More powerful/limited, older/newer

- The ciphersuite concept allows the negotiation of mechanisms between client and server
  - Both send their supported ciphersuites, and select one they both share
  - The server choses

- Exemple: ECDHE-RSA-AES128-GCM-SHA256
  - Key negotiation algorithm: ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)
  - Authentication algorithm: RSA
  - Cipher algorithm and cipher mode: AES-128 Galois/Counter Mode
  - Integrity control algorithm: SHA256

SSH (Secure SHell)

- Manages secure console sessions over TCP/IP
  - Initially designed to replace the Telnet application/protocol
  - Currently used in many other applications
    - Execution of remote commands in a secure manner (rsh / rexec)
    - Secure copy of contents from/to remote hosts (rcp)
    - Secure FTP (sftp)
    - Secure (generic) communication tunnels (carry standard IP packets)

- Security Mechanisms
  - Confidentiality and integrity of the communications
    - Key distribution
  - Authentication of the intervenient entities
    - Server / Hosts
    - Client users
    - Both achieved through several, and differentiated mechanisms

SSH: Authentication mechanisms

- Server: an asymmetric key pair
  - Public keys are distributed during the interaction
    - Not certified!
  - Clients store the public keys from previous interactions
    - Key should be stored in some trusted environment
    - If the key changes the client user is warned
      - e.g., server is reinstalled, key is regenerated, an attacker is hijacking the connection
      - Client can refuse to continue with the authentication process

- Clients: authentication is configurable
  - Default: username and password
  - Other: username + private key
    - The public key MUST be pre-installed in the server
  - Other: integration with PAM for alternative authentication mechanisms

Centralized network authentication

- Used for restricting network access to known clients
  - In cabled networks
  - In wireless networks
  - In VPNs (Virtual Private Networks)

- Usually implemented by a central service
  - AAA server
    - Authentication, Authorization and Accounting
    - e.g. RADIUS and DIAMETER
  - This server defines which network services the user can make use of

freeRADIUS

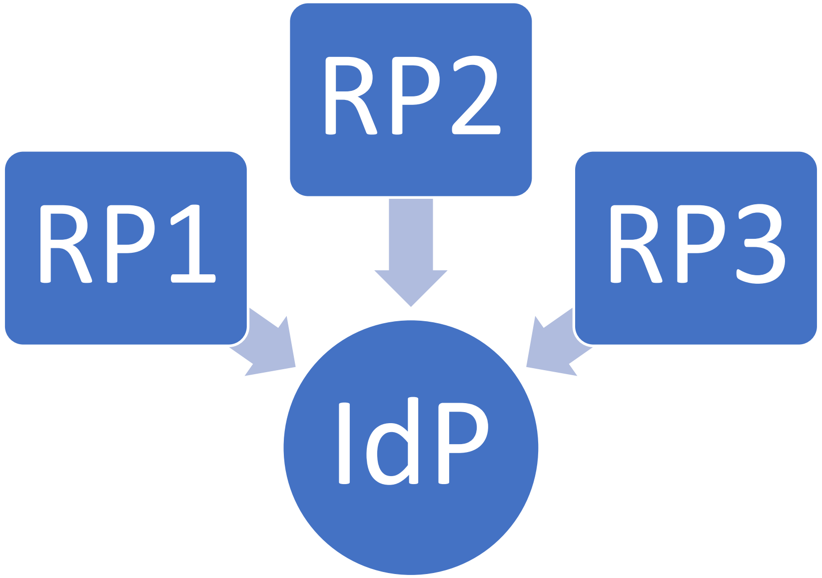
Centralized authentication

- Advantages:
  - Can reuse same credentials over multiple systems/services
  - Single secure repository for credentials
    - More difficult to steal credentials when used in many services
  - Can implement restrictions to services/systems

- Disadvantages:
  - Requires additional servers
  - Single point of failure: without authentication systems, no one will be authenticated
    - Important to also deploy local credentials for admins
  - Introduces delays in the authentication process
  - Privacy issues (tracking because it records every device/user session)

Authentication by an IdP (Identity Provider)

- Unique, centralized authentication for a set of federated services
  - The identity of a client, upon authentication, is given to all federated services
  - The identity attributes given to each service may vary
  - The authenticator is called Identity Provider (IdP)
  - The federated service is called a Relying Party (RP)
  - In some cases, the provided identity attributes are shown to the client



- Examples
  - Authentication at UA
    - Performed by a central, institutional IdP (idp.ua.pt)
    - The identity attributes are securely conveyed to the service accessed by the user
  - Autenticação.gov (www.autenticacao.gov.pt)
    - Performed by a central, national IdP
    - The identity attributes are shown to the user
  - Other:
    - Services used worldwide: Google, Facebook, etc.

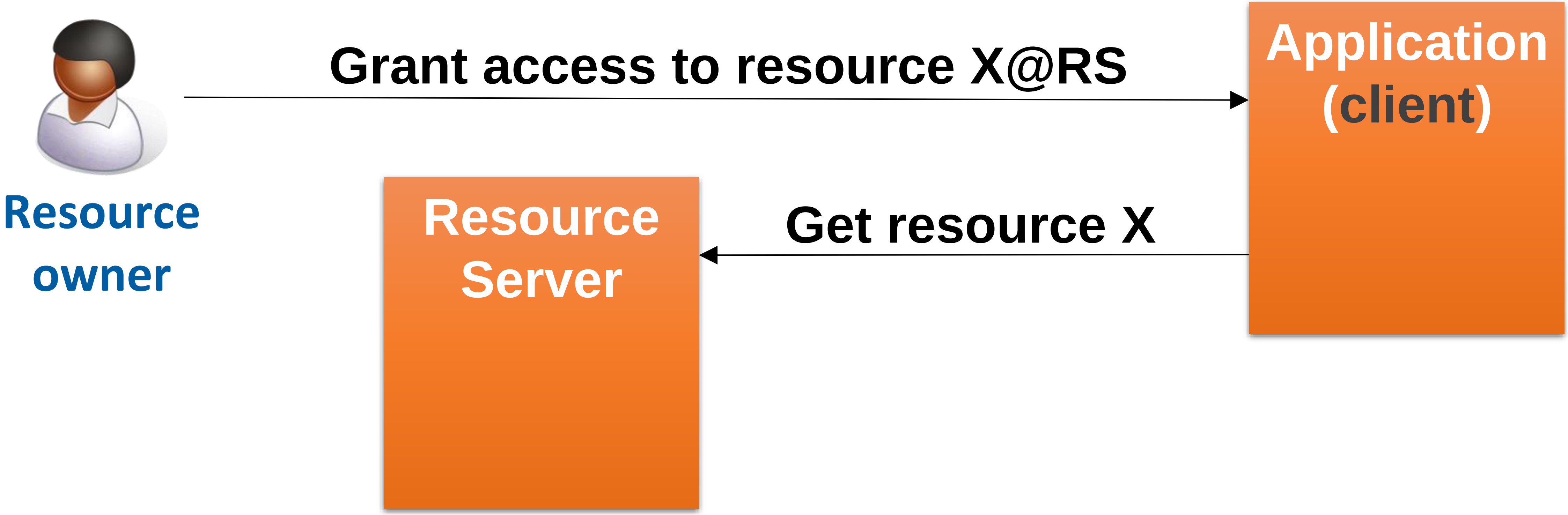
Single Sign-On (SSO)

- A facility usually associated with IdP
  - Both not mandatory nor always appropriate

- SSO exists for simplifying users' life
  - They login just one for accessing several federated services during a given period

OAuth 2.0: delegation (RFC 6749)

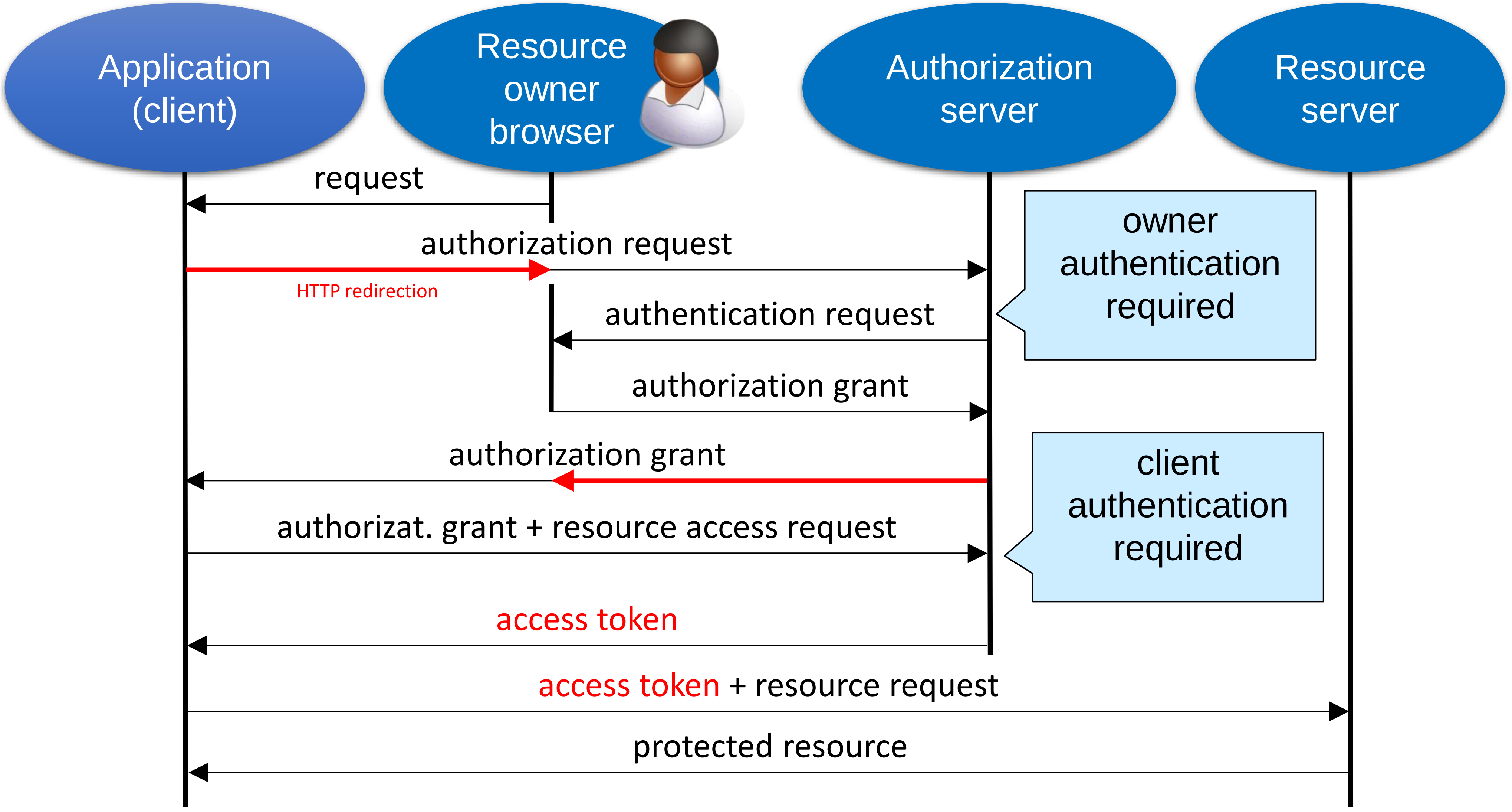
- A framework to allow users to **delegate access** to their resources on their behalf



OAuth 2.0 players

- Resource owner
  - An entity capable of granting access to a **protected resource**
  - **End-user**: a resource owner that is a person
- Client
  - An **application** making requests for protected resources on behalf of the resource owner and with its authorization
- Resource Server
  - The server hosting protected resources
  - Responds to protected resource requests that have an **access token**
- Authorization Server
  - The server issuing access tokens to clients after successfully **authenticating resource owners** and obtaining their **authorization** for the clients to access one of their (users) resources

Protocol flow



OpenID Connect (OIDC)

- An identification layer on top of OAuth 2.0
  - OAuth 2.0 provides the fundamental centralized authentication

- The protected resources are identity attributes
  - Packed in **scopes**
  - The attributes are called (identity) **claims**