==========Security==========

Confidentially: Prevent the divulgation of non-authorized information.

Integrity: Ensure that the information/data received is exactly as the data sent from an authorized entity; Ensures **authenticity** (the involved entity is, in fact, the one she claims to be); Prevents that entities refuse to generate information: **non-repudiation.**

Availability: Information is accessible and usable on demand by an authorized entity – prevent denial of service (DoS).

==========Cryptography==========

Cryptographic Mechanisms:

Primitives: mathematical operations used as building blocks in the realization of schemes; e.g. DES, AES (key size: 128, 256, …), RSA (1024, 2048, …), ECC, etc;

Schemes: combination of primitives and additional methods for the realization of cryptographic tasks such as cipher and digital signature; e.g. DES-CBC-PKCS5Padding, RSA-OAEP, etc;

Protocols: sequences of operations performed by one or more entities, involving schemes and primitives; e.g. TLS, TLS_RSA_WITH_AES_128_CBC_SHA, etc.

	Symmetric	Asymmetric
Confidentiality	Symmetric Cipher	Asymmetric Cipher
Authenticity	MAC	Digital Signature

Symmetric Schemes: Process of protection and deprotection uses the same key; Keys are usually used during a short period of time; Keys are established after a negotiation process between who encrypts and who decrypts;

Symmetric Cipher: `Keys: D(k)(E(k)(m)) = m`; Primitives: DES, AES, Blowfish; Does not guarantee data integrity. DES-CBC-PKCS5Padding.

Mac: `Keys: V(k)(T(k)(m),m) = true`; Tag has fixed dimension (e.g. 160, 256, 512 bits); Code detectors and error correctors do not serve for MAC schemes; Examples: HMAC-SHA1, HMAC-SHA256, etc; Does not ensure non-repudiation.

Block Cipher: block size n (64, 128 bits); key size (56, 128, 256 bits)

Operation Modes: plain text patterns should not be evident in the ciphered text; The efficiency of the used method should be higher than the efficiency of the cipher primitive; Ciphered text dimension should be approximately equal to plain text dimension; The decipher should be capable of error detection and correction; **Random access** - ability to decipher and change only part of the ciphered text.

ECB (Electronic Code Book): random access - independent blocks, all blocks ciphered with the same key;

CBC (Cipher Block Chaining): error propagation - dependent blocks, IV (stored and transmitted in plain; unique, unpredictable)

CTR (Counter) Mode: stream operation mode; random access - independent blocks

Padding: a way to take data that may or may not be a multiple of the block size for a cipher and extend it out so that it is; must be unpredictable (not just 0s or 1s); e.g. PKCS# 7, CMS, SSL, etc.

Authenticated Cipher: confidentiality and simultaneously authenticity; Encrypt-then-MAC (tag indicates changes in ciphered text); MAC-then-Encrypt (tag generated over the msg); e.g. GCM (Galois/Counter Mode), OCB, CCM.

Hash Functions: easy to get `H(x)` given x; hard to get `x'`, given x, such that `x'!=x` and `H(x')=H(x)` - second pre-image; hard to get `(x,x')`, such that `x!=x'` and `H(x)=H(x')` – collision; hash of `m` serves as `m`'s digital signature. Example: MD5 (n=128), SHA-1 (n=160); based on boolean and arithmetic ops over small words (16, 32, 64 bits); data integrity, MAC, digital signature, pwds,…; HMAC uses hash functions – deterministic.

Asymmetric Schemes: Process of protection and deprotection uses different keys; Keys are usually used during a long period of time; There are public keys and private keys; higher cost than symmetric.

Asymmetric Cipher: `KeyPairs: D(kd)(E(ke)(m)) = m`; Does not guarantee data integrity; limitations in the dimension of the ciphered info; RSA primitive – prime numbers factorization. Used in hybrid schemes to enc the symmetric key. RSA, PKCS#1 v1.5, OAEP.

Digital Signature: `KeyPairs: V(kv)(S(ks)(m), m) = true`; Signature with private key; verification with public key; public key diffused through a certificate; without the `ks`, its computationally infeasible: selective forgery – given m, find s such `V(kv)(s, m) = true`; existential forgery - find the pair `(m, s)` such that `V(kv)(s, m) = true;` fixed size: 160, 1024, 2048 bits; RSA, PKCS#1 v1.5 or PSS, hash function; Ensures non-repudiation.

==========JCA==========

Design Principles: Algorithm independence and expandability (it is possible to use the same API for several algorithms, and to add new algorithms); Implementation independence and interoperability (multiple implementations of the same algorithm; interoperability between implementations – sign with one implementation and verify with another)

Architecture: Cryptographic Service Provider (CSP) - cryptographic mechanisms implementation; Engine Classes - abstract definition of a cryptographic mechanism; Specification Classes - representations of cryptographic objects (keys & algorithm parameters)

Cipher: `update` (continues incremental op), `doFinal` (termina op; add padding), `wrap`, `unwrap`

MAC: `update`, `doFinal` (finaliza e retorna tag)

Signature: `initSign`, `initVerify`, `update`, `sign` (finaliza e retorna signature), `verify` (finaliza e retorna validade)

KeyStore: stores keys and certificates

==========Certificates==========

Certificates X.509: subject (and subject public key), issuer (CA), content, attributes (validity, name, …), issuer signature (performed with the CA private key); only store the public key, the private key is associated with the cert in a secure storage (like CC) (private key formats: PEM format, PKCS#12 / PFX). Authenticated association between subject and public key.

Certification Path: set of certificates that allows the validation of a certificate; Private keys formats: PEM e PKCS#12 / PFX;

Extensions: normalized way to add information not considered in the base standard; Composition: ID, value, critical flag (if true, cannot be ignored); Profile: set of extensions e.g., PKIX (Public Key Infrastructure for the Internet)

==========SSL/TLS==========

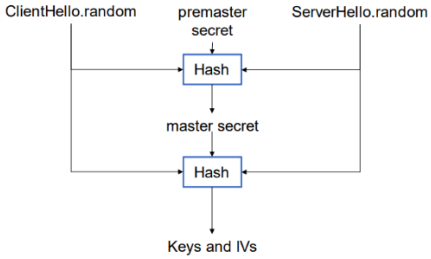
Goal: secure channel with confidentiality, integrity and authentication; Between the APP Layer and the Transport Layer; Requires a reliable transport layer, such as TCP;

Handshake Protocol: establishes the secure channel; responsible for negotiation of the operating params, authentication of the endpoints and establishment of a secure key; Auth optional on both ends; RSA for key transports and DH for key agreement. Message modification is detected with the **`Finished`** message, which guarantees that both endpoints receive the same messages – HMAC of the master secret and the handshake messages.

Perfect forward secrecy is the property of the handshake that guarantees that, if the private key is compromised, it is not possible to decrypt previous master secrets, and, consequently, it is not possible to decrypt record protocol messages.

Master Secret: the change of keys with RSA implies that the browser uses the server's public key to encrypt the pre master secret, and the server decrypts the pre master secret using its private key.

Record Protocol: transmits the encrypted data; fragments, compresses, authenticates (MAC) and encrypts the data; same TCP conn, with 2 independent streams of data (keys, IVs and seq numbers diff for each stream – client write & server write); msg repetition detected by seq number; msg reflection detected by the MAC, through separated keys for each direction; accorded schemes depend on the cipher suit agreed (negotiated by the handshake; defines the hash function used by the HMAC (e.g. SHA), the symmetric scheme (e.g. 3DES_EDE_CBC or RC4_128), and the key establishment scheme (RSA or DH)).



==========Password Based Auth==========

Dictionary Attacks: guess the password by trying all the words in a dictionary; Protection against this attack: increase password incertitude, control the access of the verification info, increase the processing time of the auth/validation function, limit the access.

Type 1: Pre-Computed Attacks: auth fun is equal to all the users; pre-compute all the options; attacker knows the validation info; Protection with **salt:** specific to each user.

Type 2: attacker knows the validation fun; Limit the access of the validation function: back-off (increase time between failed attempts), connection termination, blocking (block after a certain num of failed attempts), jailing (access with limited functionality).

Increase Request Cost: computational challenge has to be in the client side, before the request is sent to the serve – CAPTCHA.

==========Web Auth==========

HTTP Cookies: maintains info about the client state; used via HTTP headers; contains info in the form of a key-value pair, range of URLs that the cookie is valid for and an expiration date; used to create sessions, automated login, navigation history and register client prefs;

Auth Process: Phase 1 – verify the use's credentials and obtain an authenticator (token); Phase 2 – presentation of the authenticator to the server automatically by the user-agent. Attacker Goals: existential falsification (obtain a valid authenticator), selective falsification (obtain a valid authenticator for a specific user), obtain the key used to generate the authenticators.

Auth via Cookies: Session ID stored in the cookie – infeasible to create a valid session ID; Cookie protected by a MAC (if confidentiality is required, the cookie should be encrypted); Logout – invalidate the session ID, putting the cookie in a blacklist; Protection – transport via SSL (`secure` flag) and client-side protection (`httpOnly` flag).

Cross-Site Request Forgery (CSRF): attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated; `SameSite=Strict` option ensures that cookies are not used in cross-site requests.

Cross-Site Scripting (XSS): a type of injection attack that occurs when an attacker injects malicious code into a web page viewed by other users. The malicious code can then be executed by the victim's web browser and can steal user's data or session cookies.

HTTP Auth: basic (credentials in plain text) or digest (credentials in hash): 1. client accesses protected resource; 2. Srv responds with 401 and WWW-Authenticate header; 3. Client sends Authorization header.

==========OAuth 2.0==========

Provide secure, conditioned, access to a resource, which the resource owner authorizes temporary access to a predetermined set of resources. **Roles:** Resource Owner (end-user); Client (application); Resource Server; Authorization Server

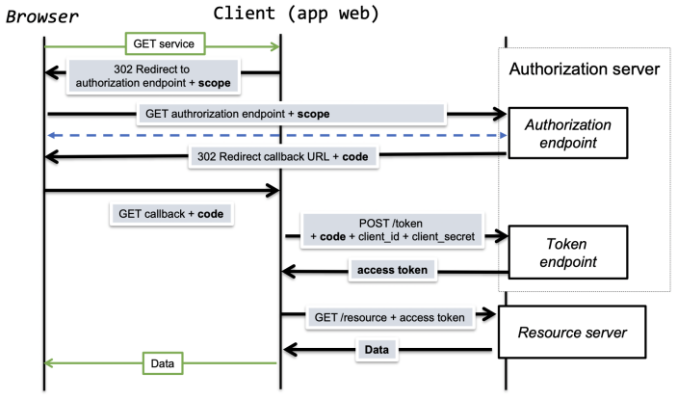
Client: when the client registration is performed with the authorization server, the client is assigned a `client_id` and a `client_secret`, which is used to authenticate the client at the authorization server.

Access Token: string representing the access credentials for a given resource; used by the client to access the protected resources hosted by the resource server, on behalf of the resource owner; opaque to the client, representing specific scopes and access duration; emitted by the auth server and validated by the resource server (both need to know the structure of the token and what represents); resource server does not need to have auth mechanisms.

Grant Flows (obtain the access token)**:** Client Credentials; Resource Owner Password Credentials (client uses the user password – only when the client is trusted); Authorization Code Grant; Implicit (delivered directly to the client by the user agent).

Refresh Token used to obtain a new access token.

Scope: represents the type of access/authorization being requested.; expressed as a list of space-delimited, case-sensitive strings. **Callback/Redirect URI**: After completing its interaction with the resource owner, the authorization server directs the resource owner's user-agent back to the client. The authorization server redirects the user-agent to the client's redirection endpoint previously established with the authorization server during the client registration process or when making the authorization request. **State:** An opaque value used by the client to maintain state between the request and callback. The authorization server includes this value when redirecting the user-agent back to the client. The parameter SHOULD be used for preventing cross-site request forgery.



Front channel: Communication channel between the client and the auth server, via user-agent redirection; em caso de erro, via redirect `client_secret` never travels through the front channel. **Back channel:** client <-> token endpoint.

==========OpenID Connect==========

Identity layer on top of the OAuth 2.0. Enables clients to verify the identity of the End-User based on the authentication performed by an Auth Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner. End-User, Relying Party (client app) and OpenID Provider (auth server). **ID Token**: set of claims about the user. **UserInfo** endpoint contains info of an authenticated user.

==========Access Control==========

Architecture: Subject (user/process) that wants to access an object; Object; Policy Enforcement Point (PEP); Policy Decision Point (PDP). Policy is a set of rules that define access control.

Role-Based Access Control (RBAC): The access control is based on the role of the subject in the system and the rules that are defined for that role. **$RBAC_0$** (Users, Roles, Perms, Sessions); **$RBAC_1$** – RBAC0 + role hierarchy; **$RBAC_2$** – RBAC0 + constraints (rules that restrict the perms of the roles); **$RBAC_3$** – $RBAC_1$ + $RBAC_2$; **UA** (users-roles) **PA** (roles-perms)