Authentication, Non-repudiation and Data Integrity:

Technical and Law Perspective

Mirko Zichichi
mirko.zichichi@upm.es

some slides created by
prof. Stefano Ferretti (UNIURB)

https://mirkozichichi.me 17/03/2022

Mirko Zichichi

PhD candidate in **Law, Science, and Technology Joint Doctorate - Rights of Internet of Everything**

RIoE
Rights of Internet of Everything

Ontology Engineering Group (OEG),
Universidad Politécnica de Madrid

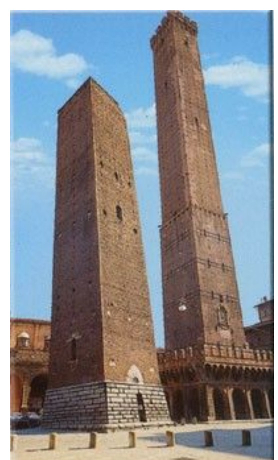
e-mail: **mirko.zichichi@upm.es**

Dipartimento di Scienze Giuridiche
Università di Bologna

e-mail: mirko.zichichi2@unibo.it

Dipartimento di Giurisprudenza
Università di Torino

email: mirko.zichichi@unito.it



Symmetric and Asymmetric **Cryptography**

Cybersecurity

The protection of resources from unauthorized access, use, alteration, or destruction

- Physical: protection of physical devices through alarms, burglar alarms, security doors, safes, etc.

- Logical: protection of information through non-physical resources (cryptography, electronic signature...)

Cybersecurity: Terminology

- **Confidentiality**
  Prevent unauthorized disclosure of data, ensure authenticity of source
- **Integrity**
  Prevent unauthorized changes to data
- **Authentication**
  Verify the identity of the other party (who am I communicating with?)
- **Availability**
  Prevent delays in data dissemination or removal
  E.g.: Denial of Service (DoS) attacks, Ransomware
- **Non-repudiation**
  Prevent the other party from denying its action

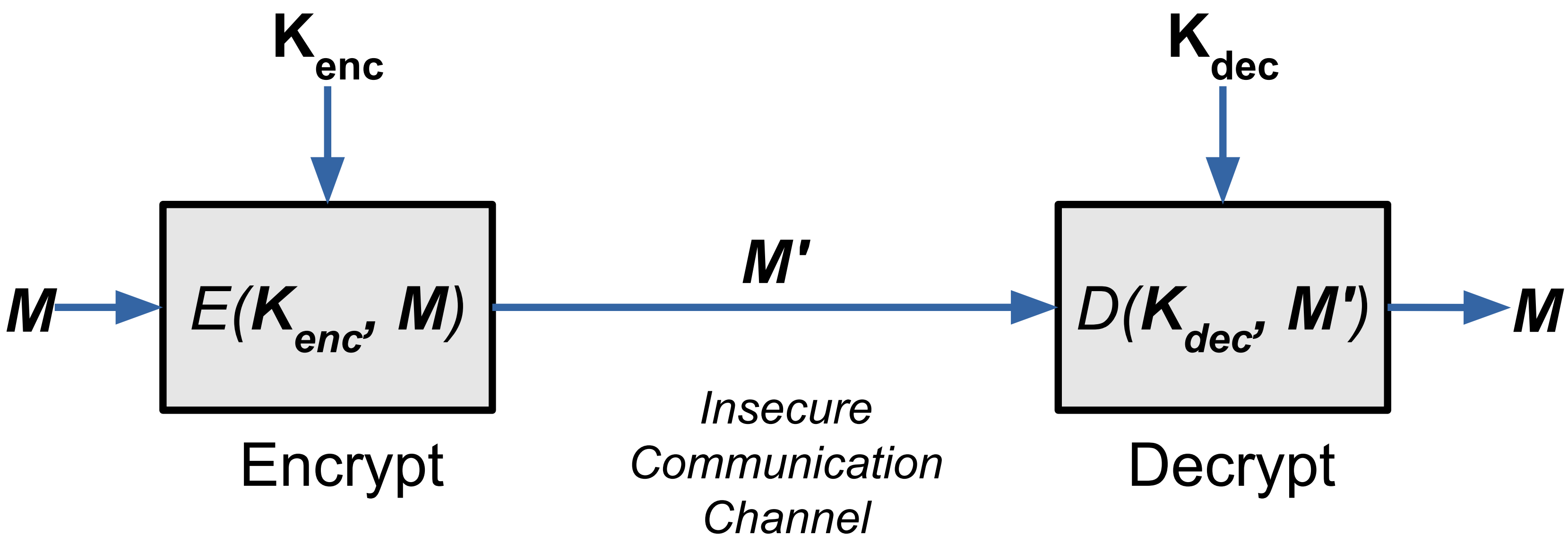
Cryptography

- A discipline that studies the techniques for encrypting a message in such a way that only the legitimate recipient is able to read it.

- Requirements:
  - Encrypt/decrypt messages must be reasonably efficient
  - Must be *"difficult"* for unauthorized parties to interpret an encrypted message

Basics

- **Encryption algorithm**:
  transforms a *"plain text"* message M into an encrypted message M'
- **Keys**: necessary for the encryption K_{enc} and decryption K_{dec}
- **Decryption algorithm**:
  transforms an encrypted message M' into a *"plain text"* message M



Notation

- $E(\boldsymbol{K}_{enc}, M) = M'$ <- encryption function
- $\boldsymbol{K}_{enc}$ <- encryption key
- M <- plaintext message
- M' <- encrypted message

- $D(\boldsymbol{K}_{dec}, M') = M$ <- decryption function
- $\boldsymbol{K}_{dec}$ <- decryption key

Cryptographic systems

- **Symmetric** -> Secret key

  $K_{enc} = K_{dec}$

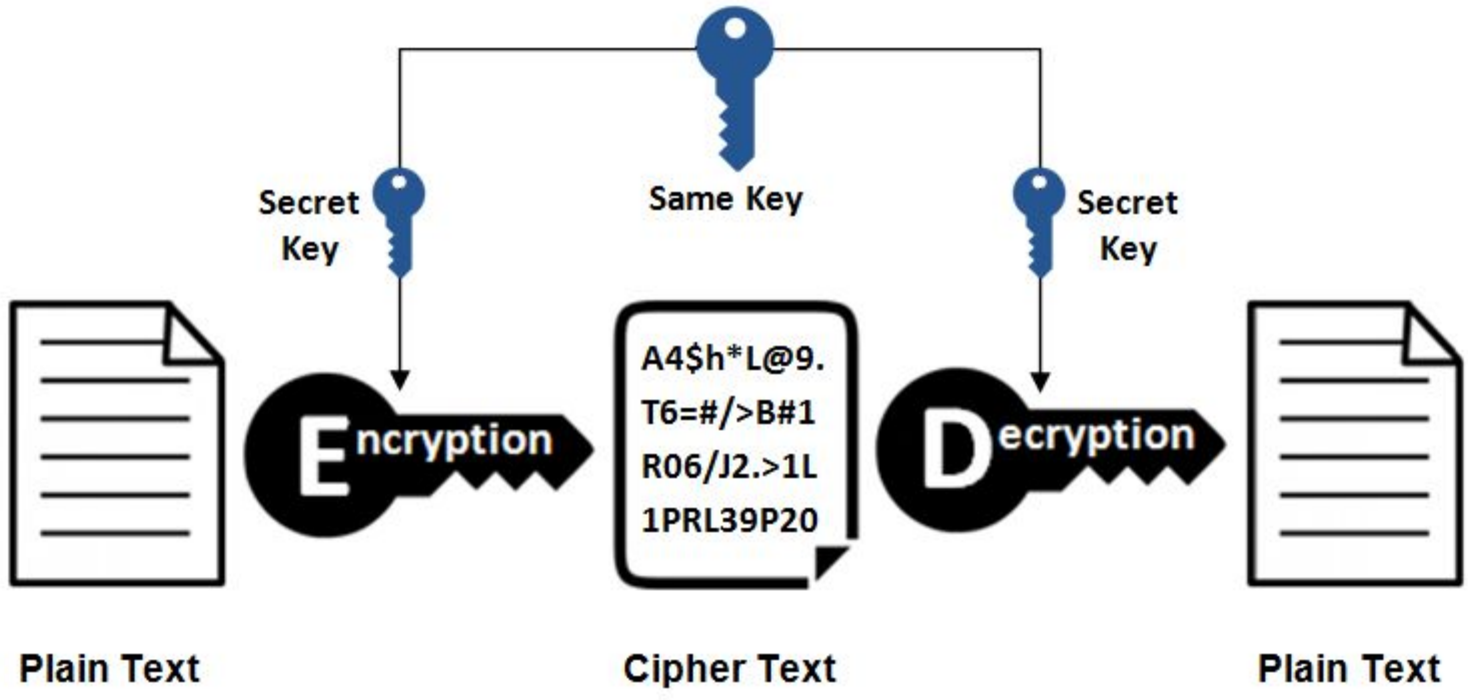
  encryption and decryption keys are equal

- **Asymmetric** -> (Public key, Private Key)

  $K_{enc} \neq K_{dec}$

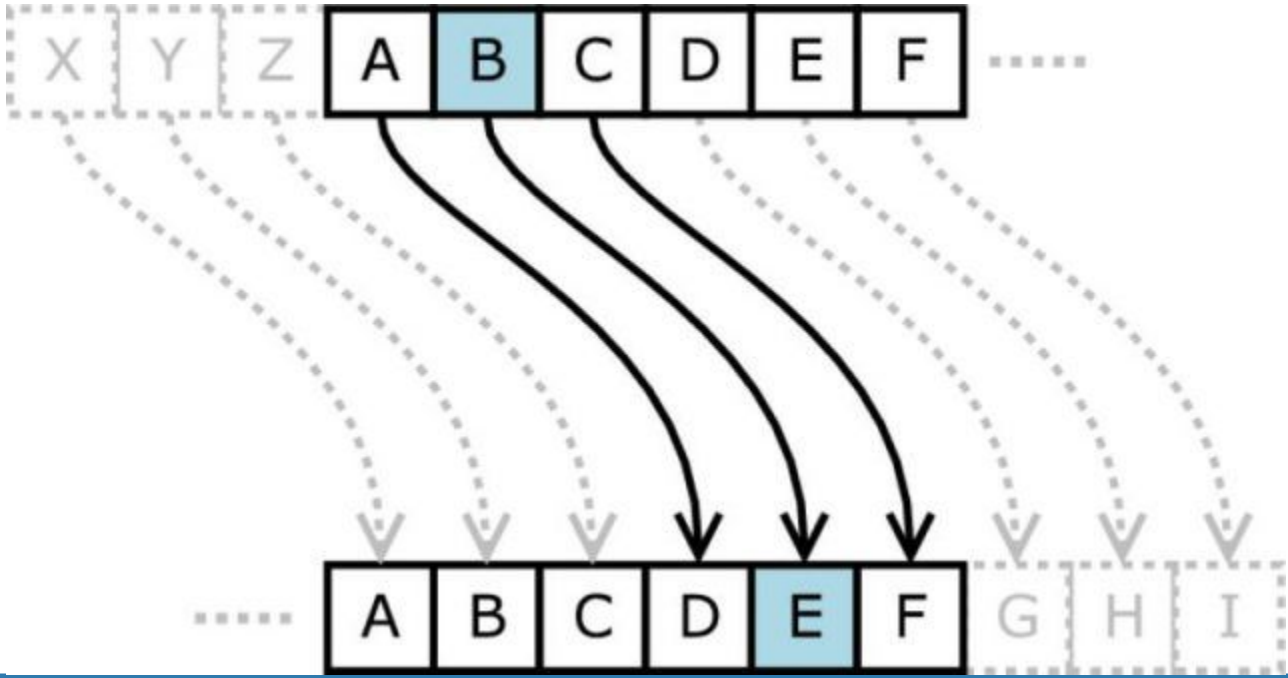
  encryption and decryption keys are different

Symmetric Encryption



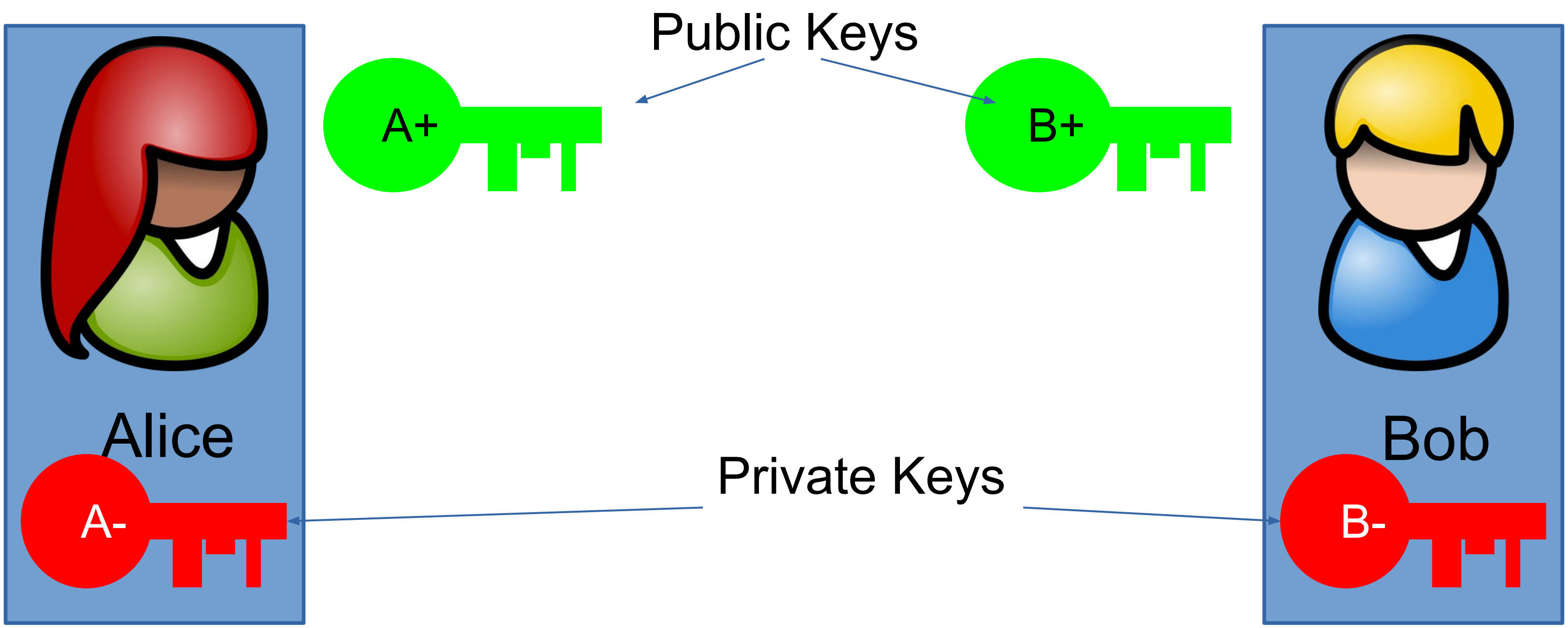
Symmetric key cryptosystems

- One of the first examples is the "Caesar's cipher".
  - The key K is an integer number
  - Each letter of the alphabet is replaced by the one that follows it by K positions
  - https://cryptii.com/pipes/caesar-cipher

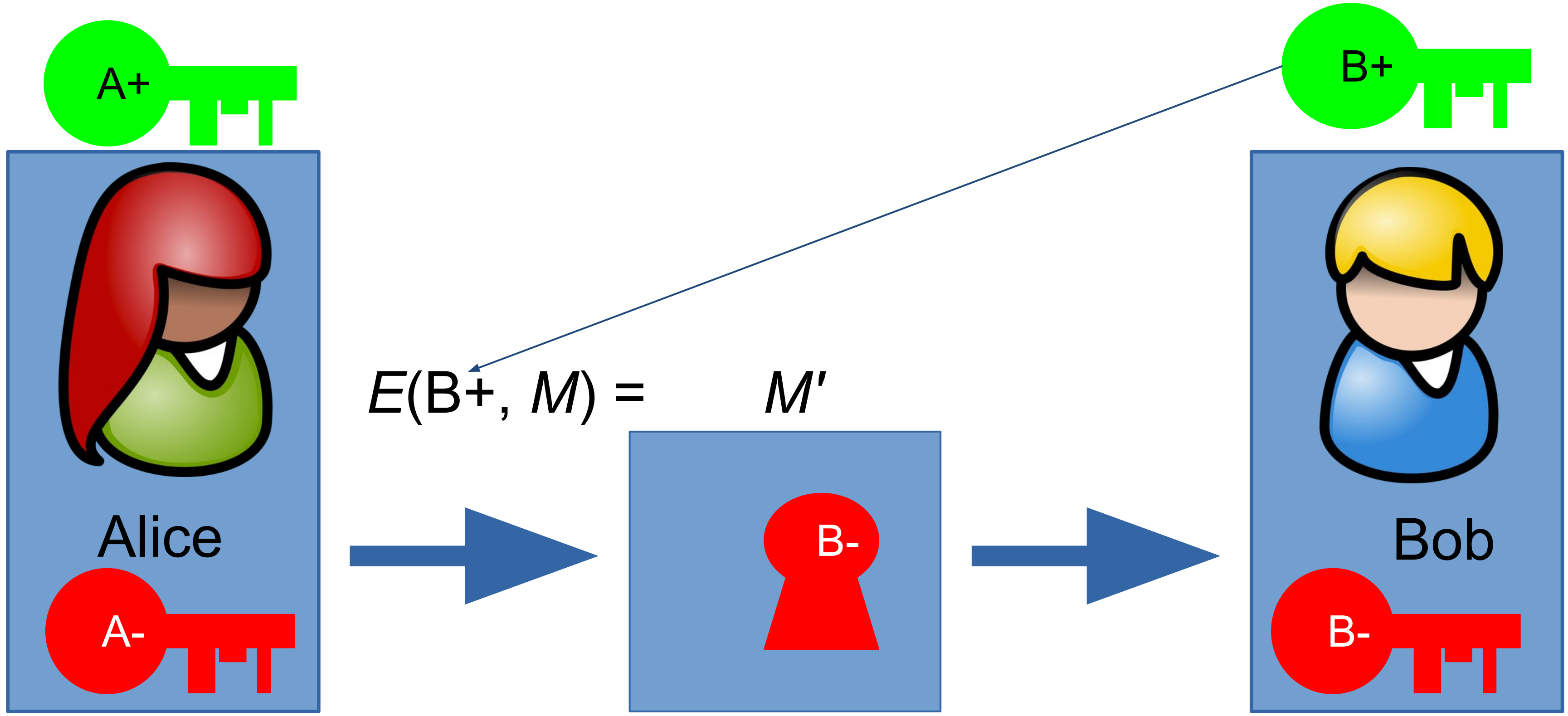


Asymmetric Encryption

- Each user has two keys
  - One is **public**, and is made available to anyone
  - The other is **private**, and the user must guard it jealously and not communicate it to anyone



Alice and Bob

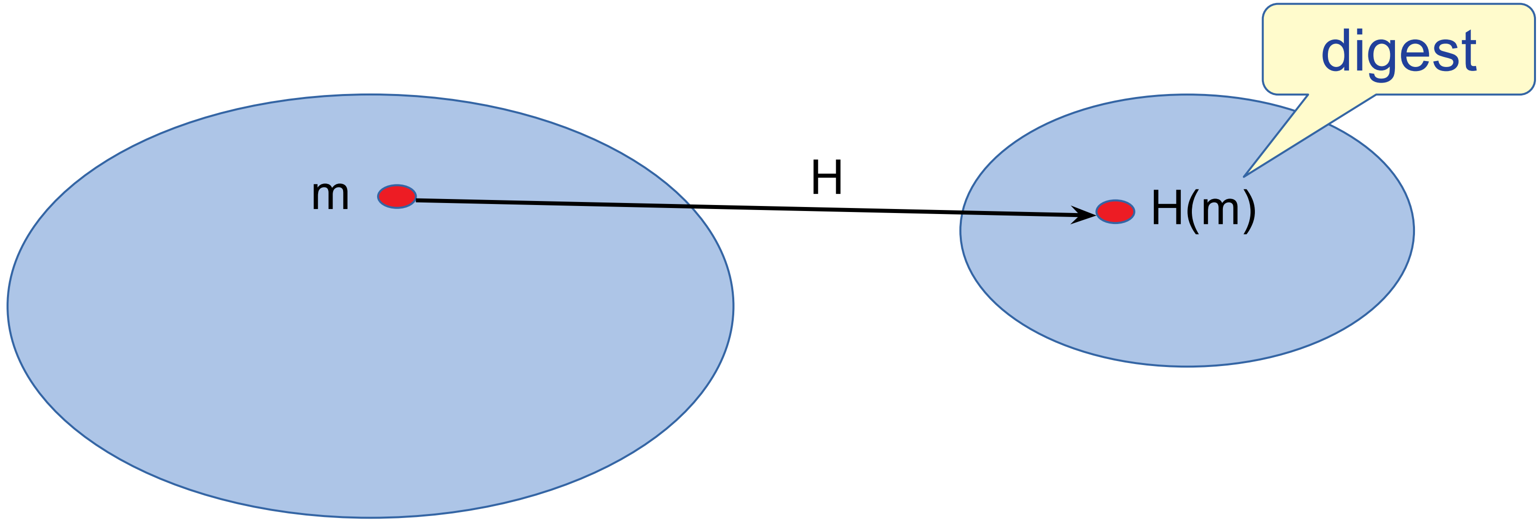


1 Alice encrypts message M with Bob's public key B+

2 Bob decrypts the message M' with his own private key B-

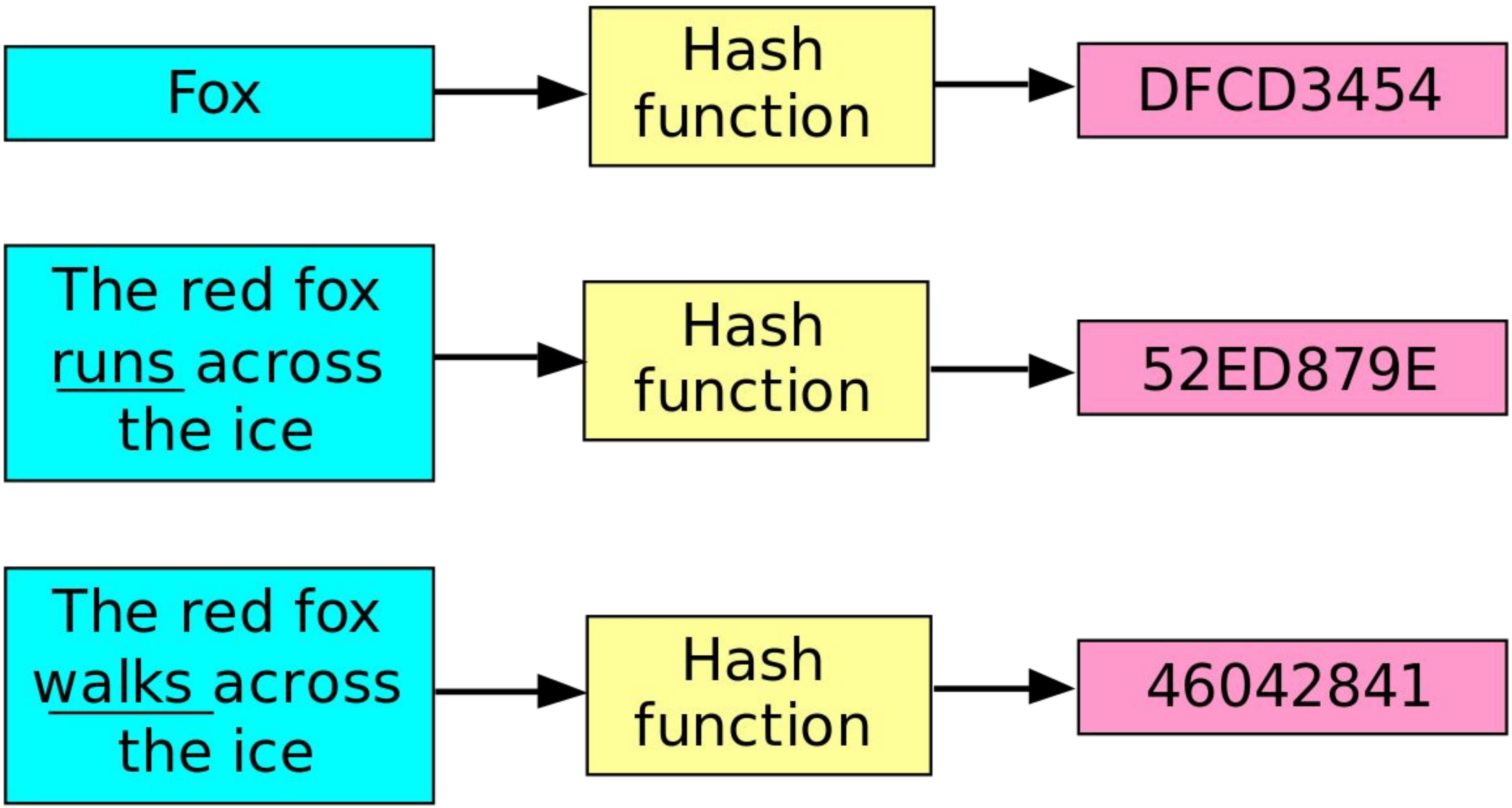
Hash Functions and Data Integrity

Hash Functions



Message *M* --------------> hash(*M*) --------------> *digest*

Hash Functions



Cryptographic Hash Functions

- **Special class of hash functions**
  - In these slides when we refer to "hash function", we actually mean a "cryptographic hash function".
- Applications
  - Integrity verification of messages and files
  - Digital signature
  - Password verification
  - Blockchain's Proof-of-work

Example (SHA-256)

msg1.txt

```
All work and no play makes Jack a dull boy
 All work and no play makes Jack a dull boy
  All work and no play makes Jack a dull boy
```

sha256

5f10e43e591ed245374fae017f8c11e429f6bc6ebf42f2d1d75fb4d6e39b8f3b

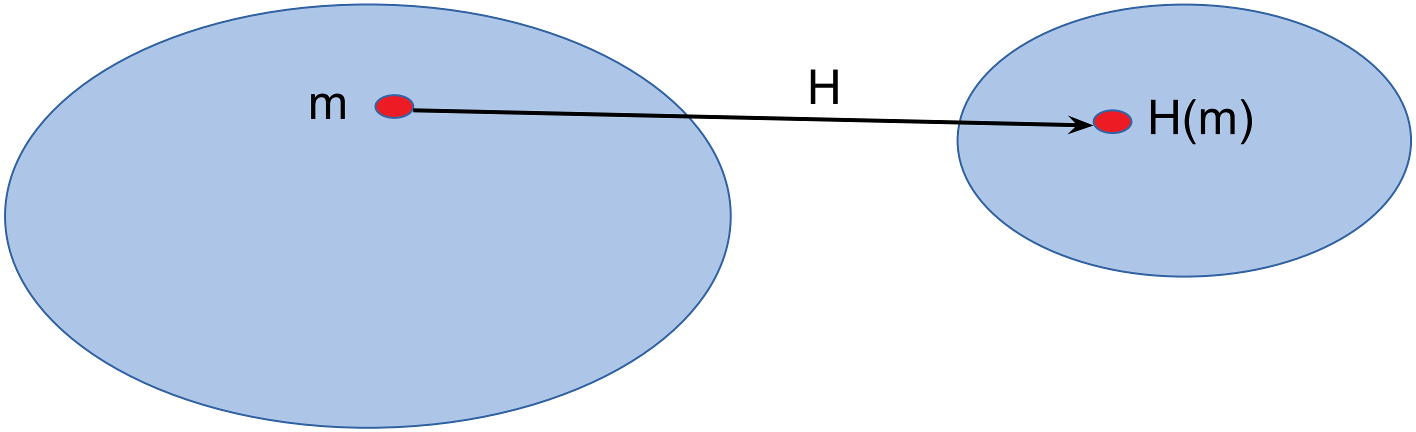
msg2.txt

```
All work and no play makes Jack a dull boy
 All work and no play makes jack a dull boy
  All work and no play makes Jack a dull boy
```

sha256

369c932a24add019689c3896657b4c625dc7864d4959aaccaffa2b75254e955b

(Cryptographic) Hash Functions



- Computationally efficient
- Security properties
  - Collision-free
  - Hiding
  - Crypto-puzzle friendly

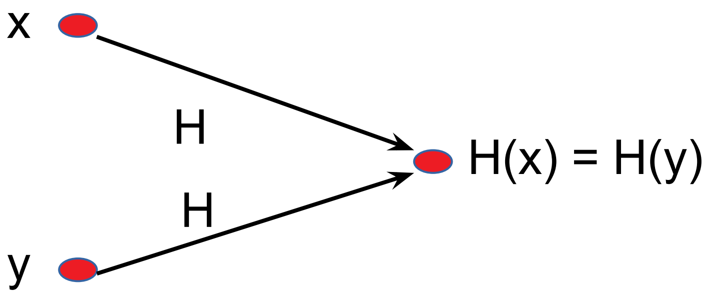
1) Hash Property: Collision-free

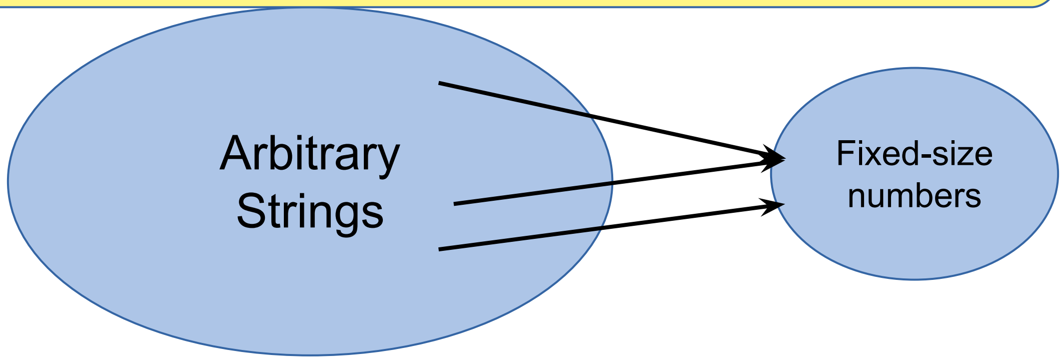
Cannot ~~have~~ find collisions

Pigeon principle: if n objects are put in m containers, with n > m, then at least one container must contain more than one object

Collisions exist!

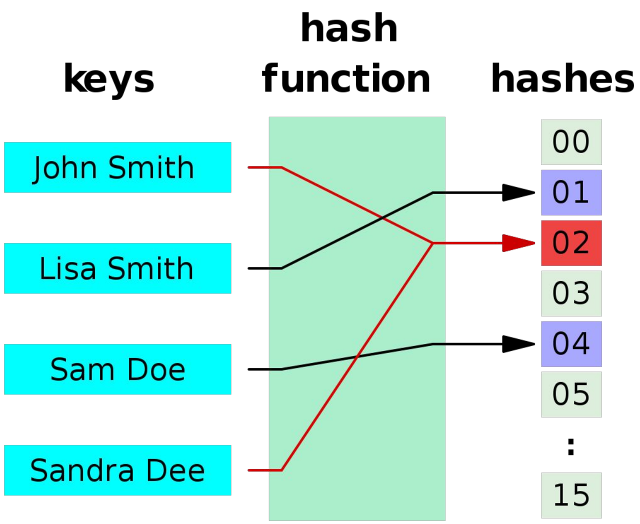
We cannot find *x* and *y* such that *x != y* and *H(x)=H(y)*





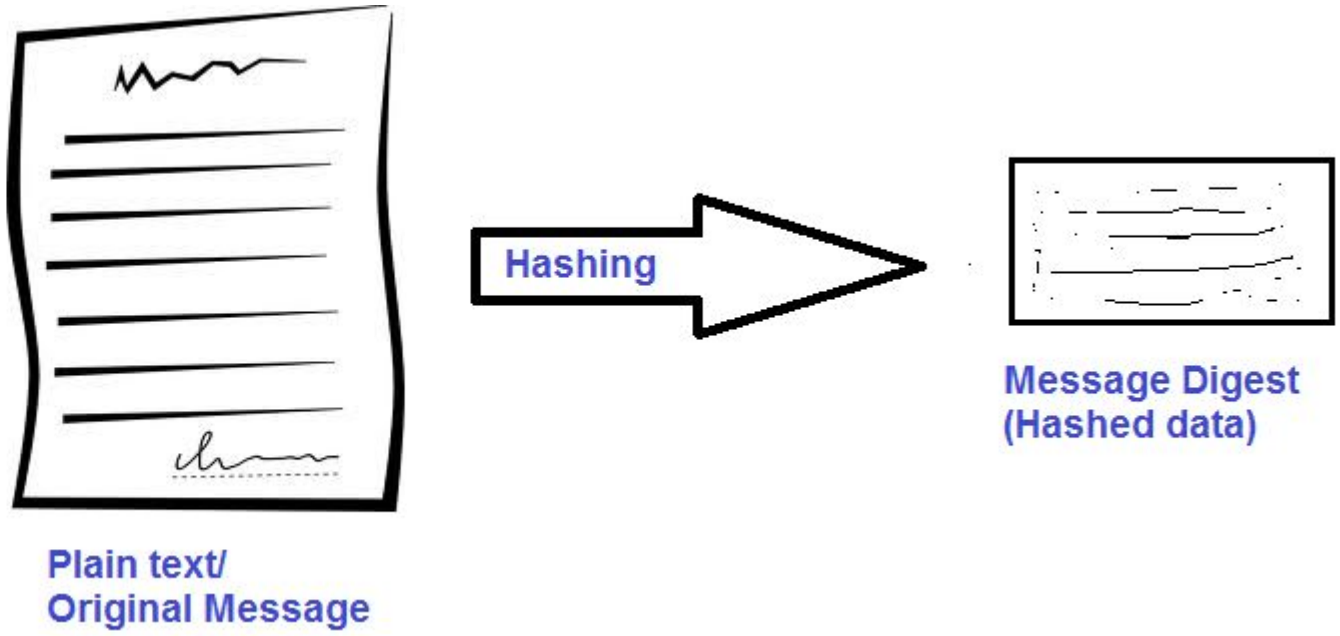
1) How to find a collision

- If output is 256 bits, then we have 2^{256} possible output values
- You need to guess
- If we pick 2^{130} randomly chosen input 99.8% chance that two of them will collide
  - This works no matter what H is …
  - … but it takes too long to matter

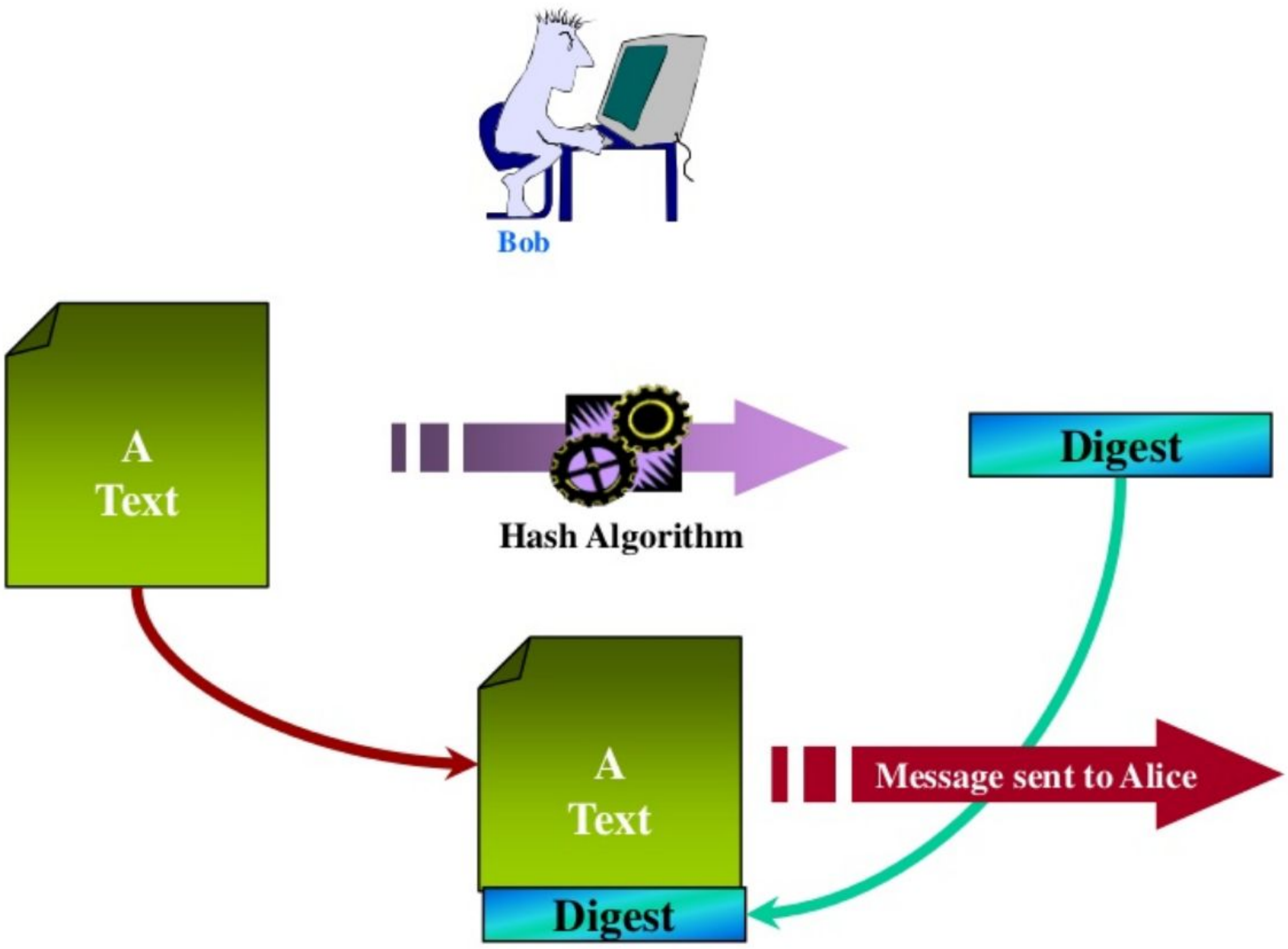


1) Application: Hash Digest Identification

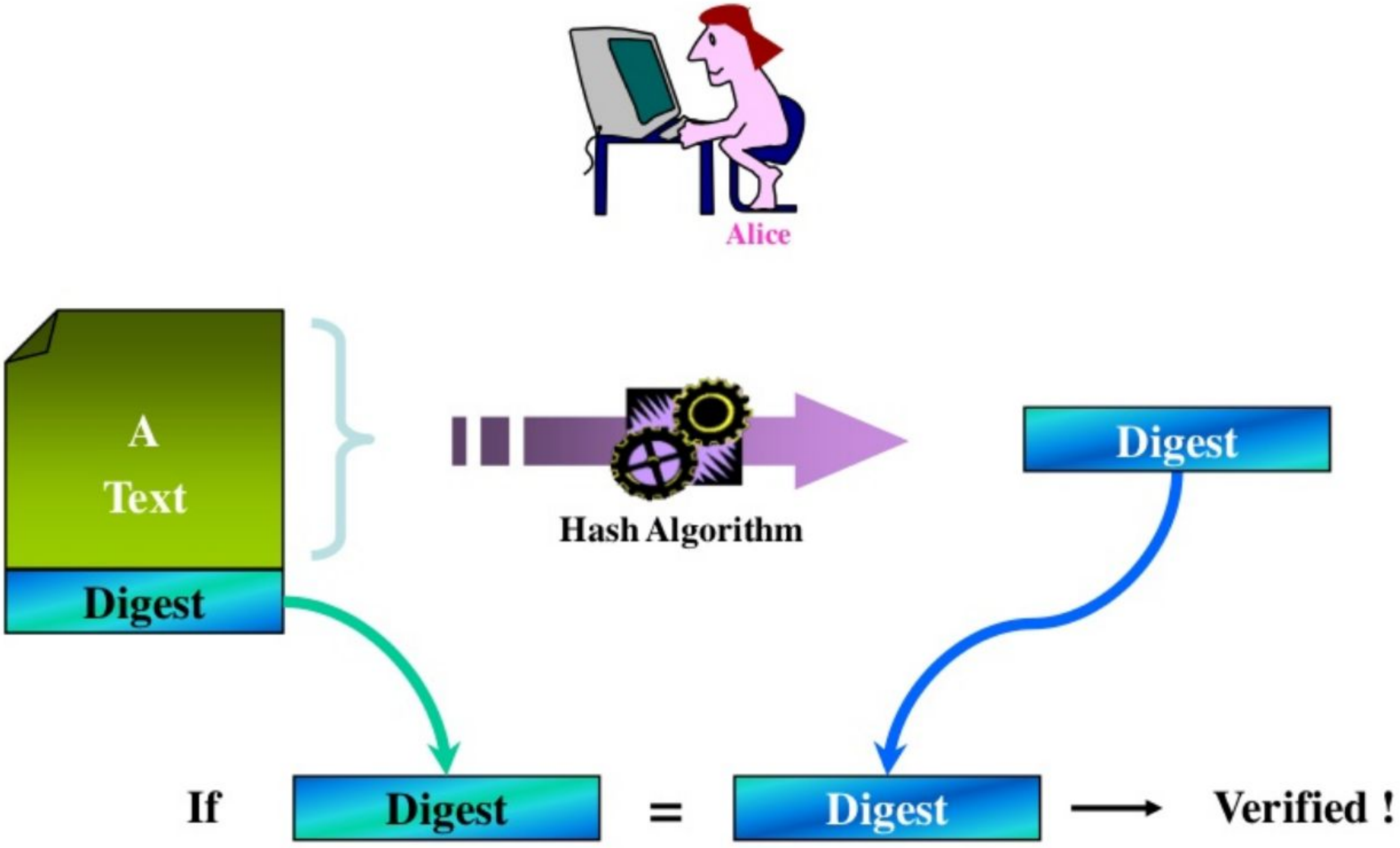
- If we know H(x) = H(y) → we can assume that x = y
- To recognize a file, just remember its hash
- Useful because the hash is small in size (256 bits)



1) Application: Data Integrity



1) Application: Data Integrity - Verify



2) Hash Property: Hiding

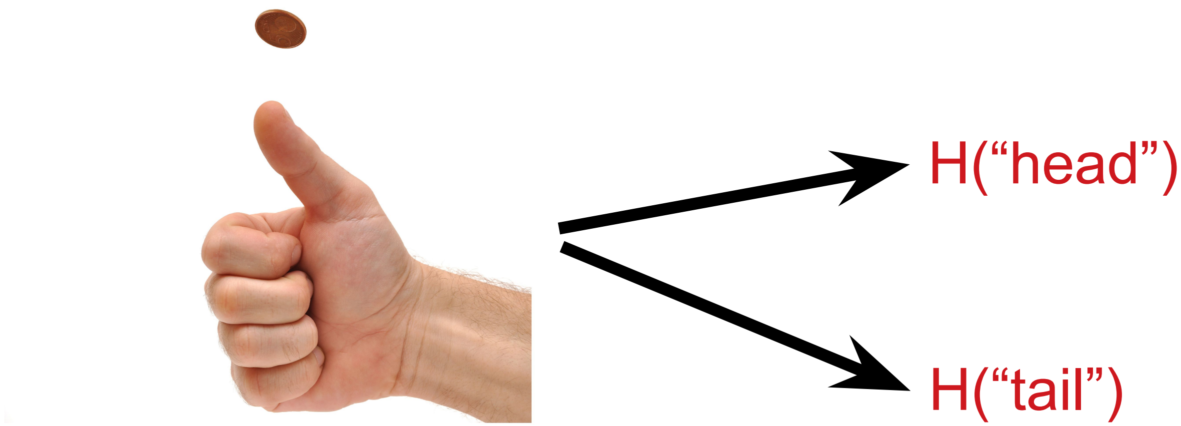
- Given H(x) it should not be easy to find x

H(moat) = 98E2W0dja8A...aslOSa216F3

H(m**ao**t) = E6712e3awa4...gz3wle3A9C9

digest length is always the same
(256 bits for SHA256)

2) Example where Hiding fails

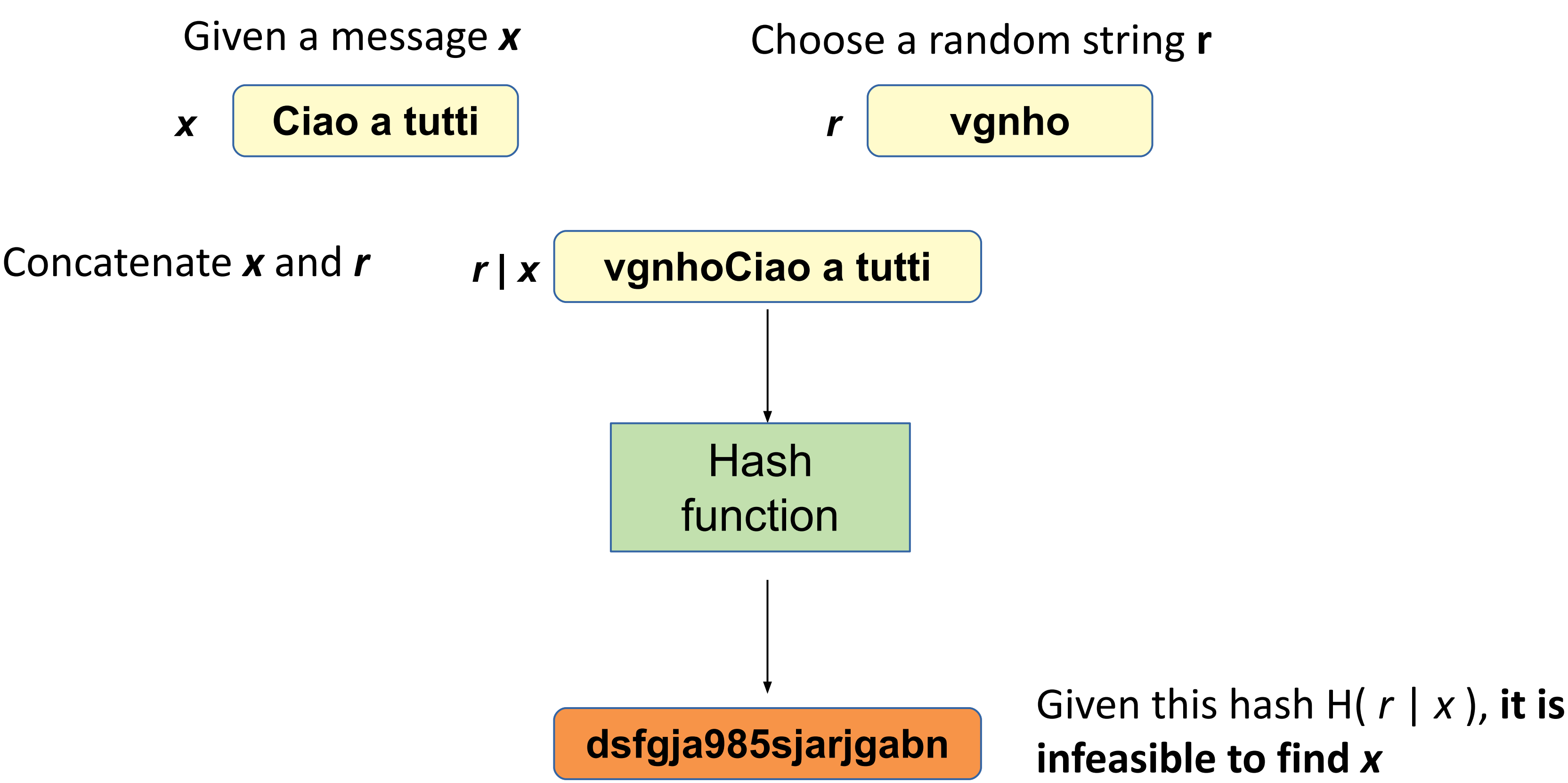


- Looking at the hash results, it's easy to tell if x was a cross or a head
- Only two inputs!
- Just hash the two inputs and look at the hash result

2) Application: Commitment

- Want to bind yourself to a value, reveal it later
  - "seal a value in an envelope" now, and
  - "open the envelope" later

3) Hash Property: Crypto-puzzle friendly



Authentication, Non-repudiation and Data Integrity

Communication in an insecure channel

It is needed a service that provides **proof of the integrity and origin** of data, and that **authenticates** the parties.

- **Authentication**
  the sender of the message is who he says he is
- **Non-repudiation**
  the sender cannot deny having sent it
- **Integrity**
  the message has not been altered along the path from sender to recipient

How? -> Digital Signature

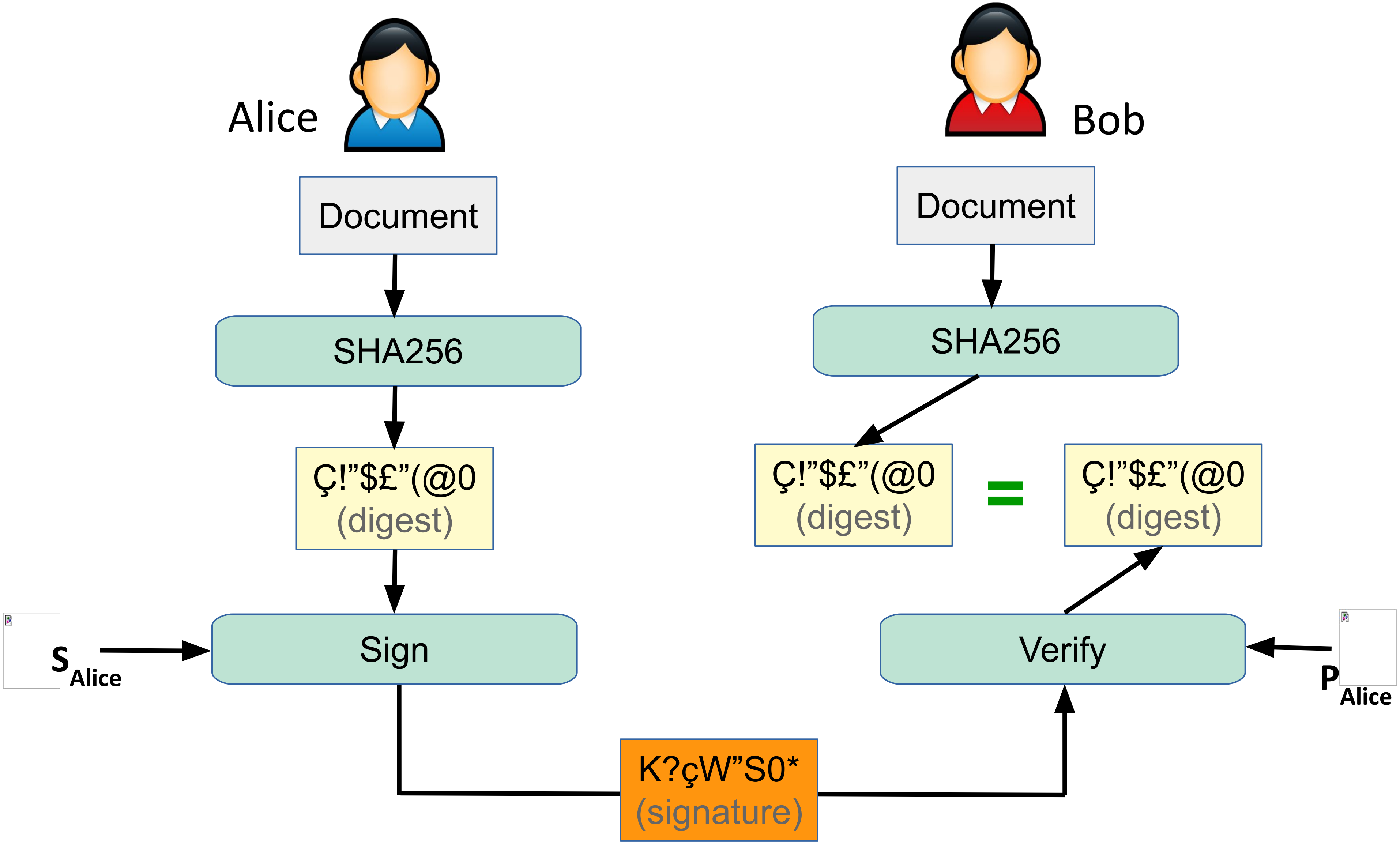
The most common method of proving the **authenticity** of digital messages (documents) is the use of a **digital signature**

- **Authentication**: a valid digital signature gives the recipient a very strong reason to believe that the message was created by a known sender
- **Non-repudiation**: the sender cannot disown a document he has signed
- **Integrity**: ensures that the message has not been altered during transport (using digest)

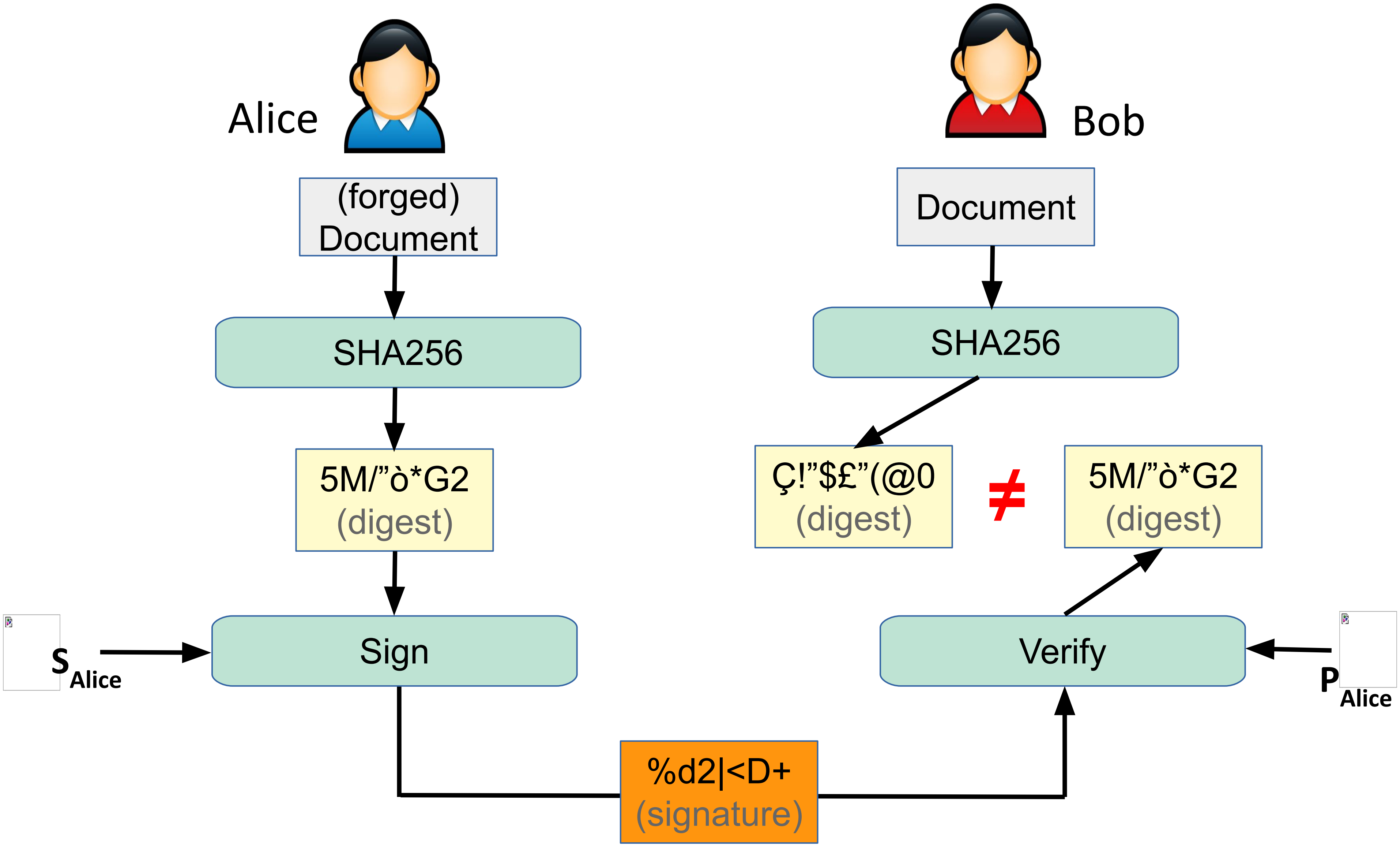
Digital Signature

- **Authentication -> Digital Certificates**
  a form of public key infrastructure on which the digital signature depends
- **Non-repudiation -> Asymmetric Cryptography**
  cryptography gives the mathematical proof of a signature
- **Integrity -> Data Hashing**
  ensures a very (very very) low probability that the data will be altered

Alice signs a document for Bob



Alice signs a (forged) document for Bob



Digital Certificate

An electronic document attesting to the unique association between a public key and the identity of a person

Alice 

Digital Certificate

Name: Alice

Address: c.Ramiro de Maeztu, 7

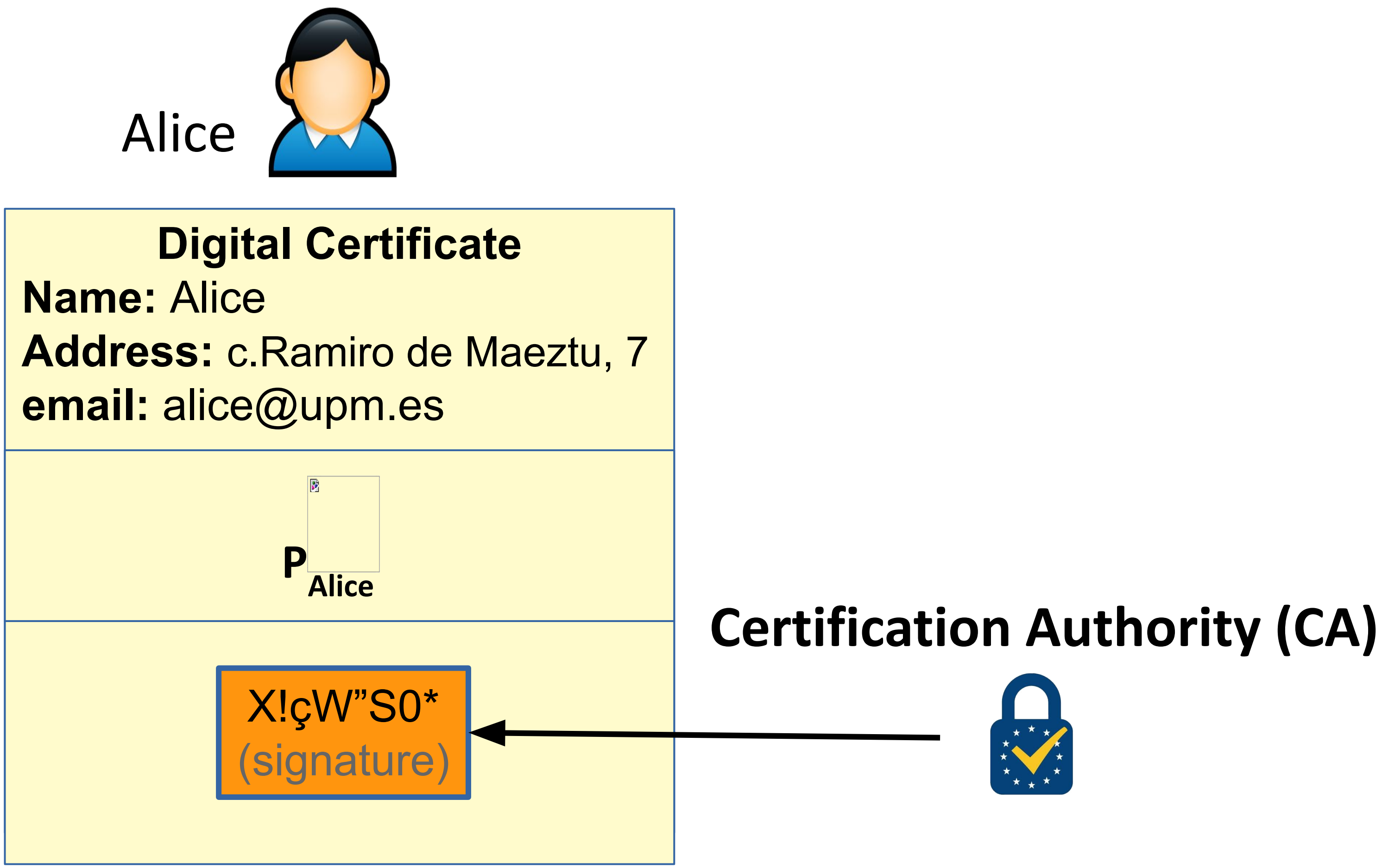
email: alice@upm.es

P_{Alice}

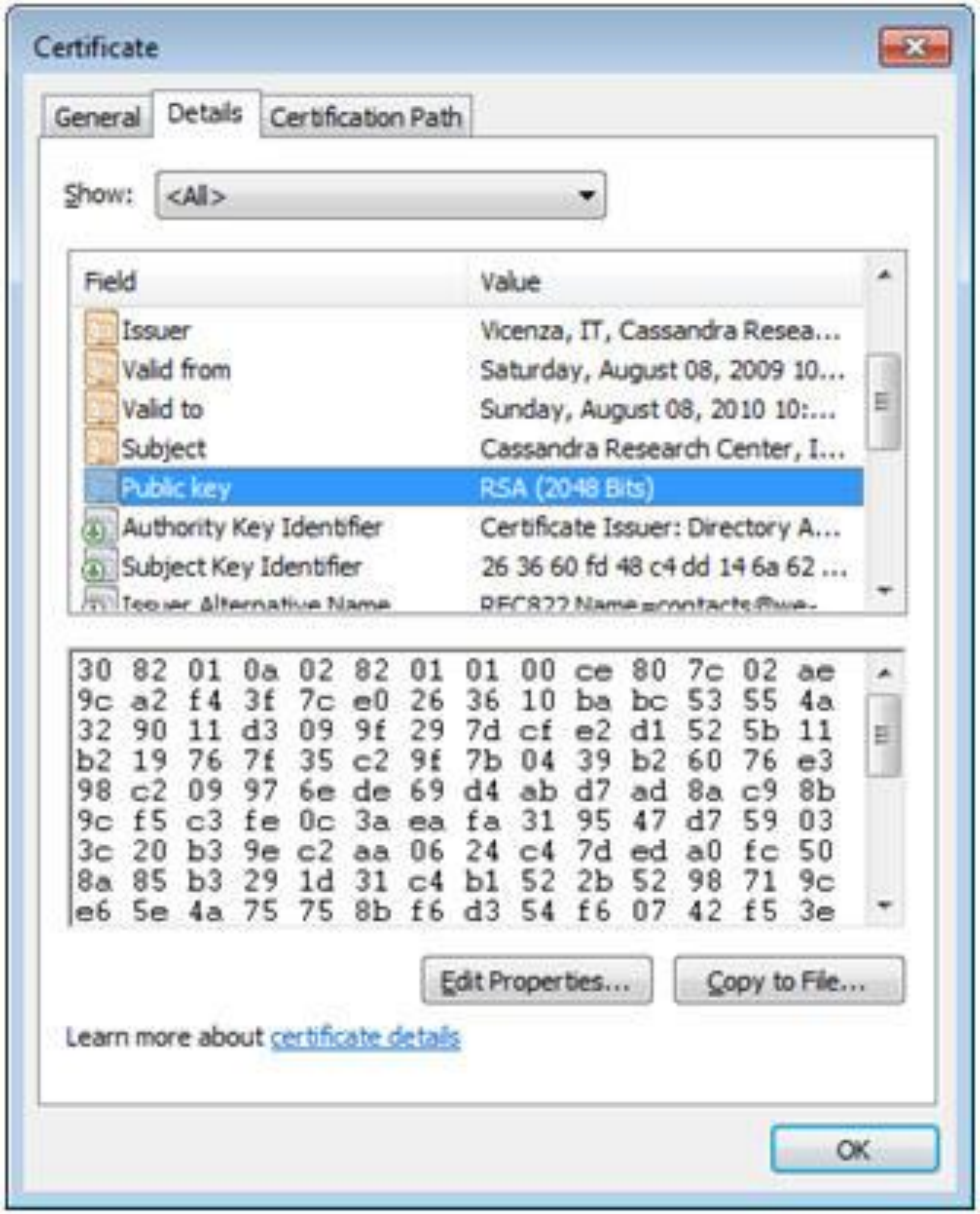
X.509 Public Key Infrastructure (RFC 5280)

- **Public Key Infrastructure (PKI)**
  - set of processes and means that allow trusted third parties to verify and/or vouch for the identity of a user
  - as well as associate a public key with that user.
- X.509 certificates are used in many Internet protocols, including **TLS/SSL**, which is the basis of **HTTPS**

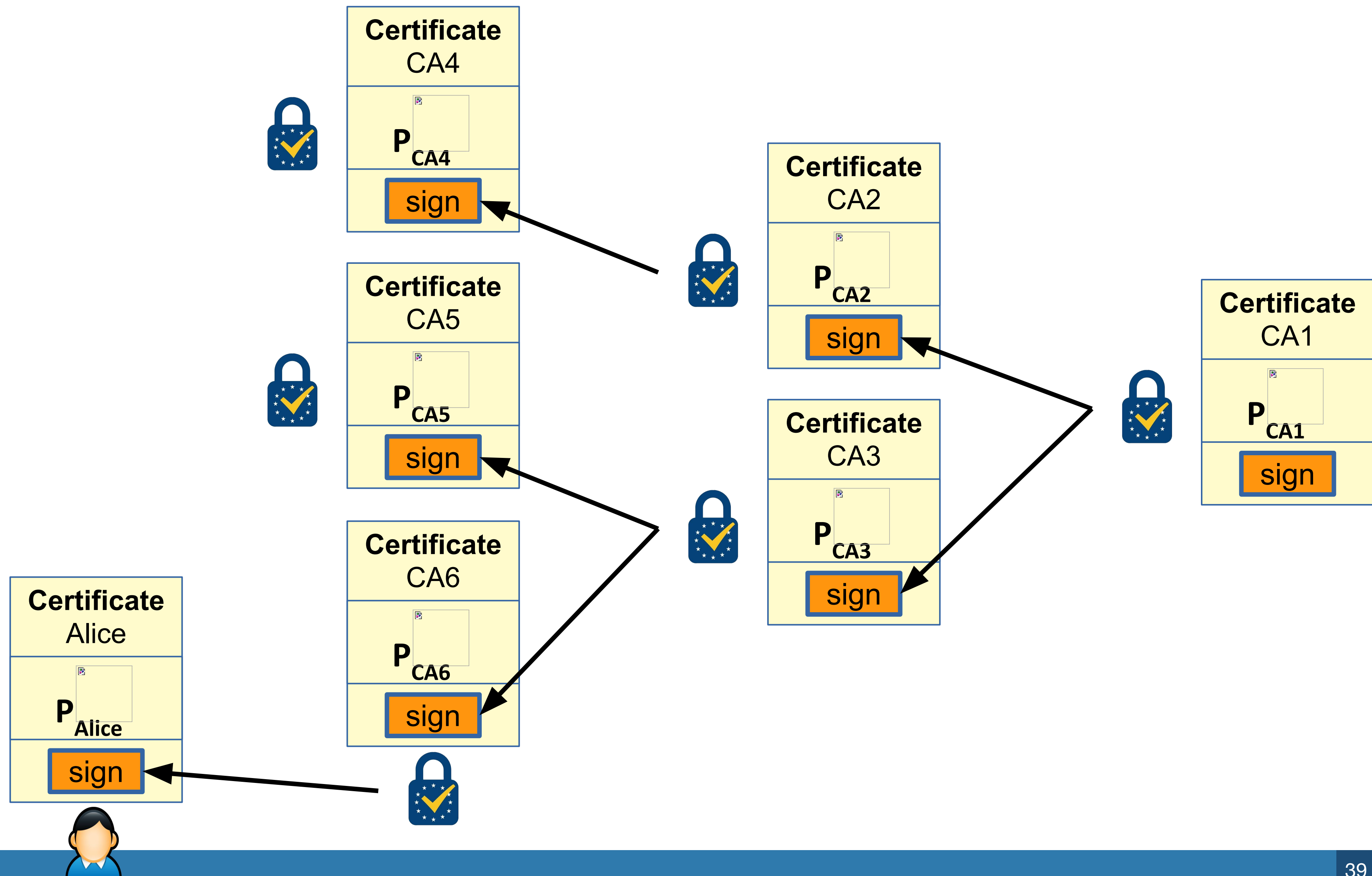
X.509 Digital Certificate



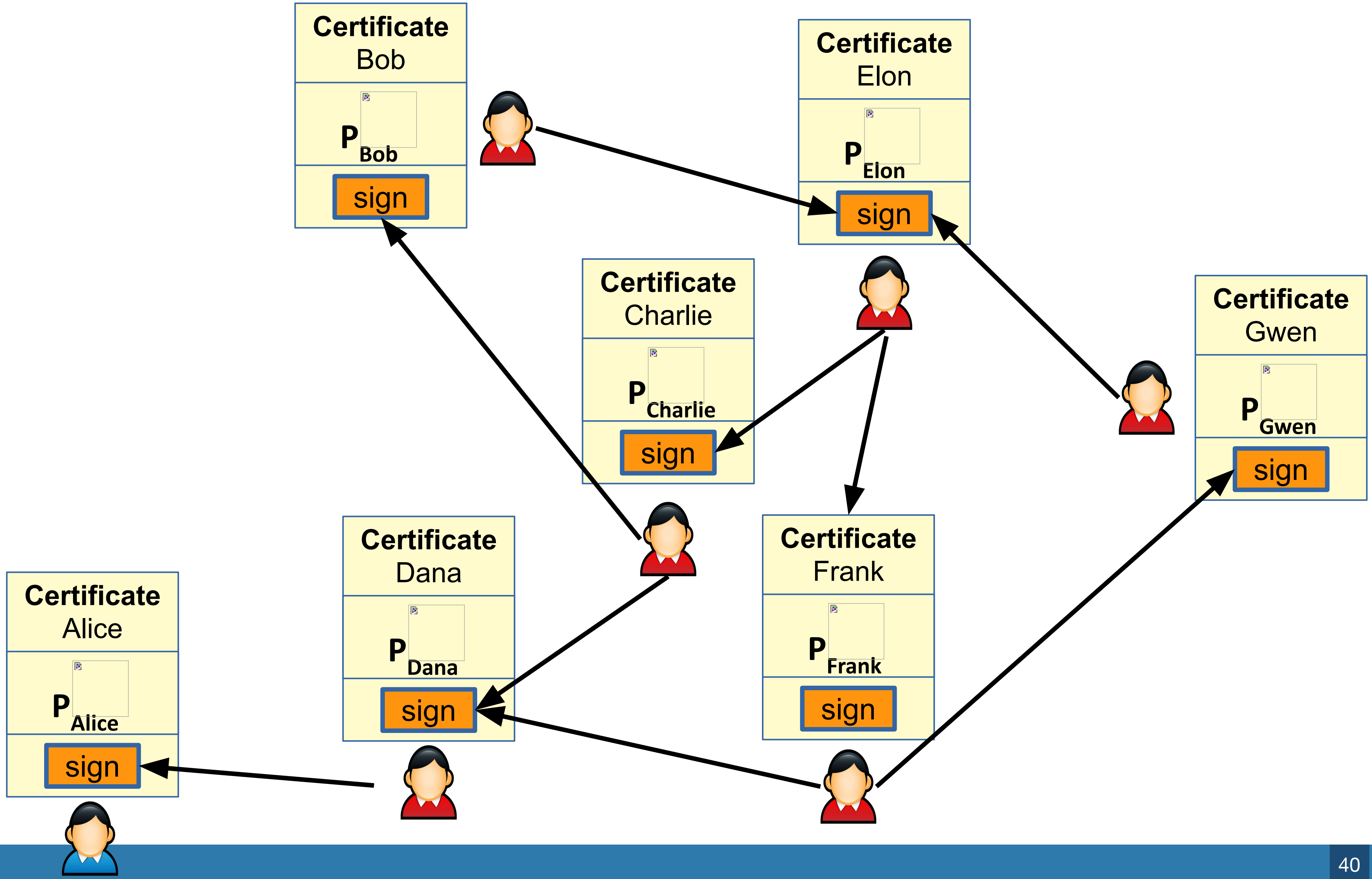
X.509 Digital Certificate



X.509 Public Key Infrastructure



Trust Network (PGP)



Trusted Timestamping

Time Stamping based on Public Key Infrastructure

- The application of a timestamp enables the **existence and content** of the document to be established in a specific point in time.
- The **RFC 3161** standard defines the process of trusted time stamping based on a PKI X.509
- **A trusted time stamping** process consists in the application of a timestamp on a digital document, by an **Accredited Certifier, e.g. Time Stamping Authority (TSA)**, by means of a digital signature on the document.

Timestamp

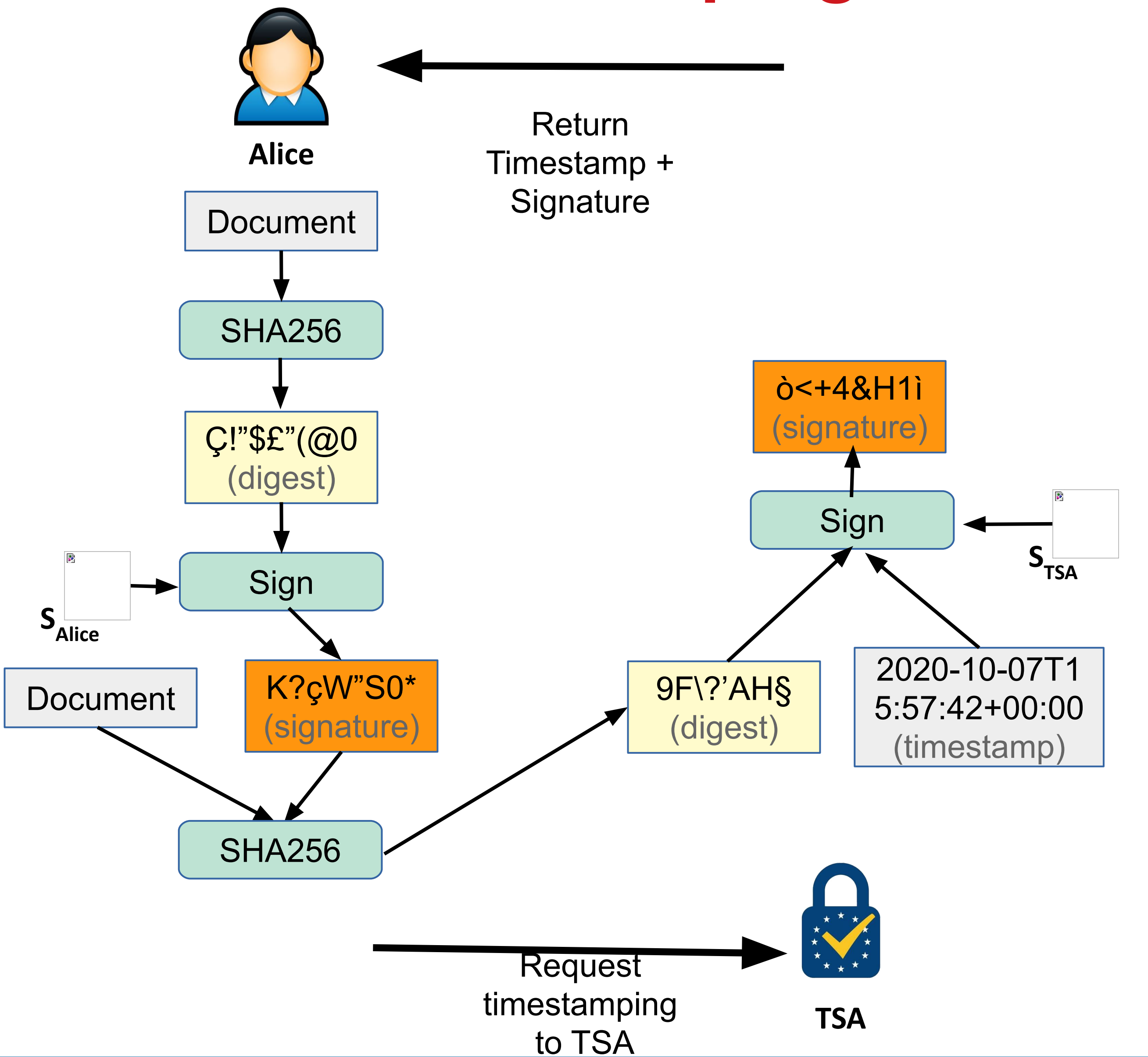
- Sequence of characters representing a date and/or time to ascertain the actual occurrence of a given event

2020-10-07T15:54:19+00:00

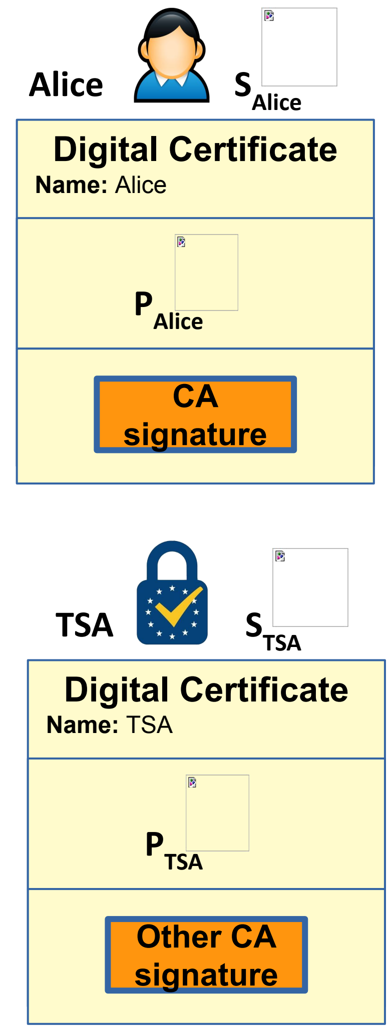
- **ISO 8601** standard for representation -> used in network protocols to limit the possibility of error.
- In most computers it is derived via **Unix time** -> the number of seconds since January 1, 1970

1602086059

Trusted Time Stamping



PKI



Distributed Time Stamping

- Instead of a single TSA we can rely on a **distributed** algorithm that guides **different parties** talking to each other to reach a **consensus** ->
- With the advent of **blockchain** and related technologies (**Distributed Ledger Technologies**), the hash of digital documents can be embedded in a transaction that is stored in the blockchain.
- In this case, the immutability of the DLT proves the time when that data existed.

Distributed Time Stamping: Issues

- The security of this approach comes from the **consensus mechanism**. E.g. in Bitcoin this is **Proof of Work (PoW)**, a huge amount of computational work done every time a new block is added to the blockchain.
- Tampering with the timestamp would require more computational resources than the rest of the combined network.
- However, the **protocol of many DLTs makes its timestamps vulnerable to some degree of manipulation** -> a timestamp can be moved up to two hours into the future and data with antecedent timestamps can be accepted earlier.

electronic IDentification Authentication and Signature (eIDAS)

electronic IDentification Authentication and Signature

- The EU Regulation 910/2014 - **eIDAS**
- Common legal basis for **secure electronic interactions** between citizens, businesses and public administrations
- EU-wide **interoperability of electronic signatures and time stamping** systems

Electronic Signature vs Digital signature

- ‘Electronic signature’: “**data in electronic form which is attached to or logically associated with other data** in electronic form and which is used by the signatory to sign;” (eIDAS Article 3.10)
- Digital signature: data appended to, or a **cryptographic transformation of a data unit** that allows a recipient of the data unit to **prove the source and integrity of the data unit** and protect against forgery (ETSI TR 119 100)
- The digital signature is used to provide concrete and practical instances of electronic signatures ->
  All electronic signatures are not necessarily digital sign.

Legal effects of electronic signatures

- Across all EU Member States, the legal effects of electronic signatures are laid down in Article 25 of eIDAS.

- An **electronic signature shall not be denied legal effect and admissibility as evidence in legal proceedings** solely on the grounds that it is in an electronic form or that it does not meet the requirements for *qualified electronic signatures*.

eIDAS identifies 3 types of e-signatures

1. (Simple) Electronic Signatures
   The eIDAS regulation defines a foundation for all electronic signatures.

   Examples
   Signing an e-mail with your name or entering a PIN code

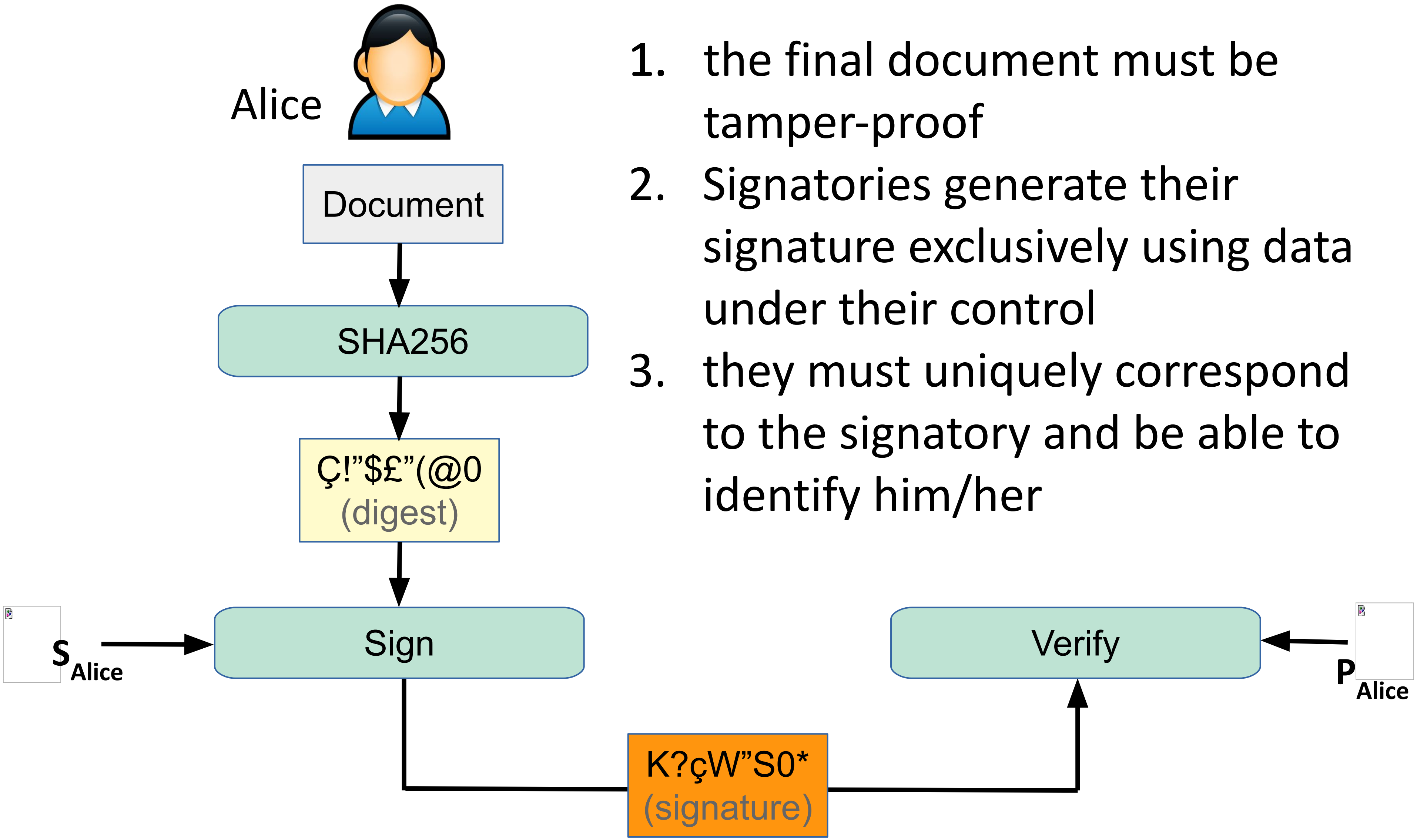
eIDAS identifies 3 types of e-signatures

2. Advanced Electronic Signatures (AdES)
   AdES signatures must uniquely correspond to the signatory and must be able to identify him/her. **Signatories generate their signature exclusively using data under their control**, while the final document must be tamper-proof.

   Examples
   ←**Digital Signatures**
   XAdES, PAdES, CAdES, Associated Signature Container Baseline Profile without Qualified Certificate, graphometric signature, biometric signature, etc.

Advanced Electronic Signatures (AdES)

1. the final document must be tamper-proof
2. Signatories generate their signature exclusively using data under their control
3. they must uniquely correspond to the signatory and be able to identify him/her



eIDAS AdES

- The formats that these advanced electronic signatures must possess are defined in the

  Implementing Decision (EU) 2015/1506 (Article 1):

  - *"Member States [...] shall recognise **XML, CMS, PDF** advanced electronic signatures at conformance level **B, T or LT level** [...]"*

- *"Advanced electronic signatures mentioned in Article 1 of the Decision must comply with one of the following **ETSI** technical specifications [...]"*

  XAdES, CAdES, PAdES

ETSI

- European Telecommunications Standards Institute
- Non-profit organisation responsible for creating and maintaining this set of technical standards (in support of the eIDAS legal framework).

XAdES: XML Advanced Electronic Signature

- Signatures encoded in a **readable text format** that complies with the rules of **XML** (Extensible Markup Language)

- XAdES is **both human- and machine-readable**, which makes it suitable for a wide variety of cases (JPEG images, MP3 multimedia files, any type of binary data, PDF documents, etc.)

- Advantage -> Facilitates ***automatic processing***, supports ***multiple signatures***, two different signatories can sign the same document or groups of documents in parallel or sequentially.

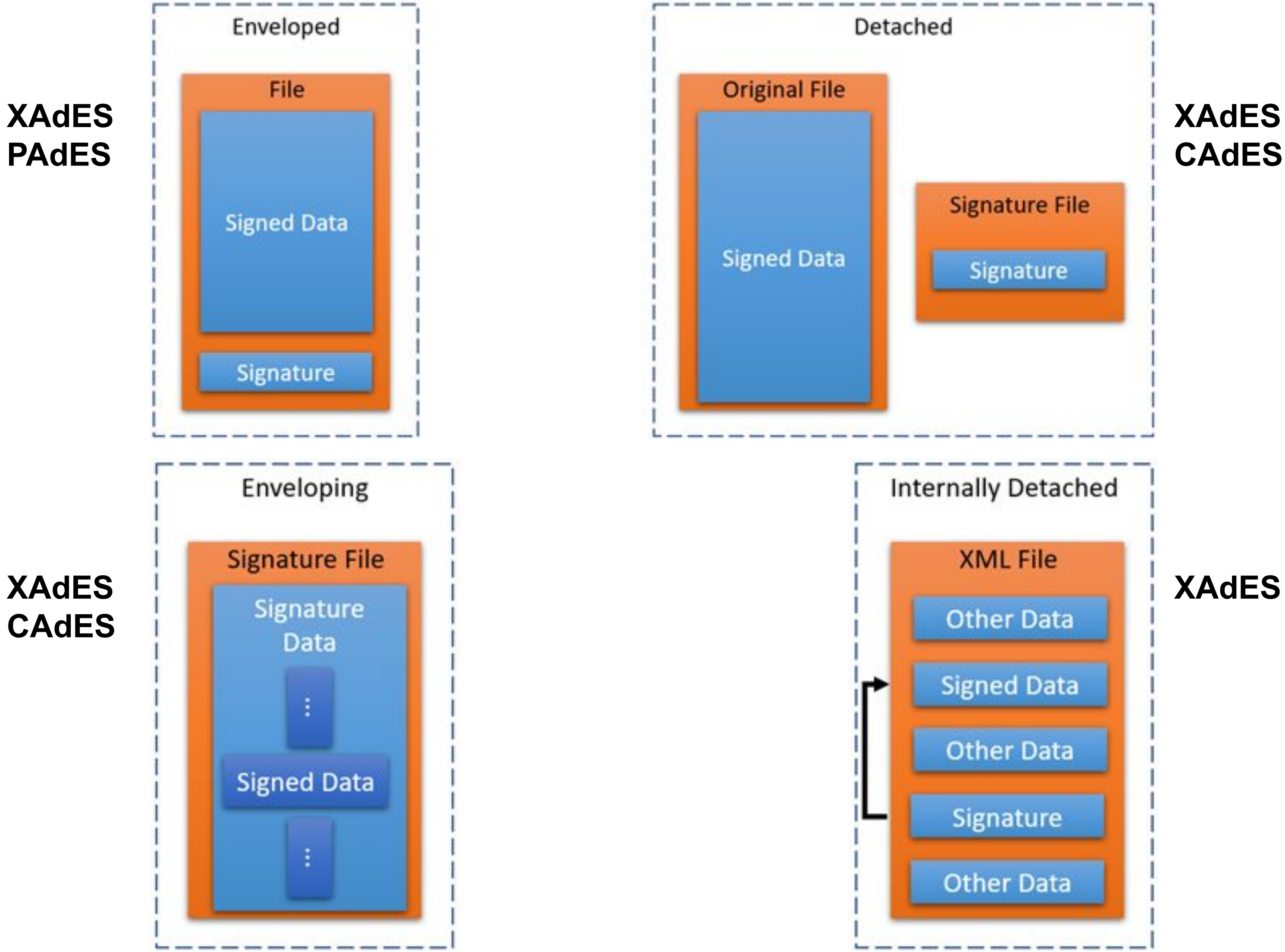
CAdES: CMS Advanced Electronic Signature

- CMS -> Cryptographic Message Syntax, an IETF Standard for encrypted messages.
- Its features are very similar to those of XAdES, except that CAdES can only be applied to **binary data**.
- In addition, it lacks some key concepts of XAdES such as multi-document signing.

PAdES: PDF Advanced Electronic Signature

- This format is **more restricted** than XAdES
  -> **only for signing PDF files**
- By default, the electronic signature is always embedded in the signed PDF document, which is only readable by humans.
- It is therefore not suitable if the data must also be read by a computer.
- PAdES does not support parallel signing and requires PDF software to sign and verify the electronic signature e.g. -> Adobe Reader.

Packaging of a Signature



Associated Signature Containers (ASiC)

Data container that holds a **group of file objects and their associated digital signatures/and or time assertions** using the ZIP format. The internal structure includes:

- **The root folder,** that holds all the container content
- **A “META-INF” folder,** that holds metadata and signatures

ASiC Simple (ASiC-S): associates one file object with either a signature file or a time assertion file.

ASiC Extended (ASiC-E): holding one or more signature or time assertion files that apply to their own sets of file objects.

“B, T or LT level”

- **B** -> *Basic Signature*, can be validated as long as the signing certificate is valid (not revoked or expired).
- **T** -> *Signature with Time*, previous level + a timestamp token on the signature as unsigned properties.
- **LT** -> *Signature with Long-Term Validation Material*, signature that provides the long-term availability of the validation material, i.e. previous level + certificate + revocation data on the signature + the timestamp(s).
- **LTA** -> *Signature providing Long Term Availability and Integrity of Validation Material,* previous level + timestamp on the validation material

eIDAS identifies 3 types of e-signatures

3. Qualified Electronic Signature (QES)

   QES is a stricter form of AdES. It has the same legal value as traditional signatures.

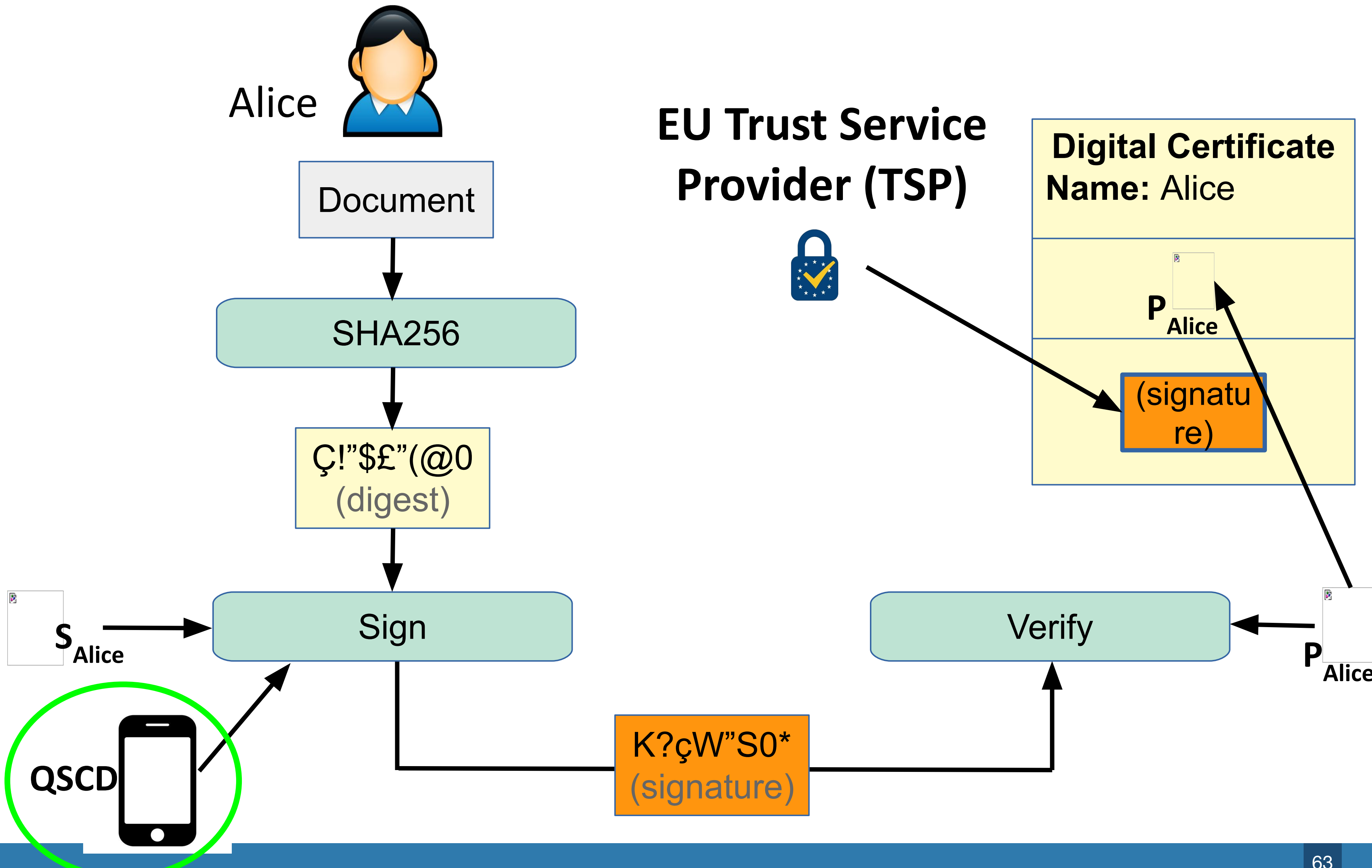
   It requires signatories to:

   a. use a digital ID based on a **Digital Certificate**, issued by a qualified **EU Trust Service Provider (TSP)**
   b. use a **qualified signature creation device (QSCD)**

   Examples

   XAdES, PAdES, CAdES with Qualified Certificate and secure device: smart card, USB token, or smartphone with one-time password

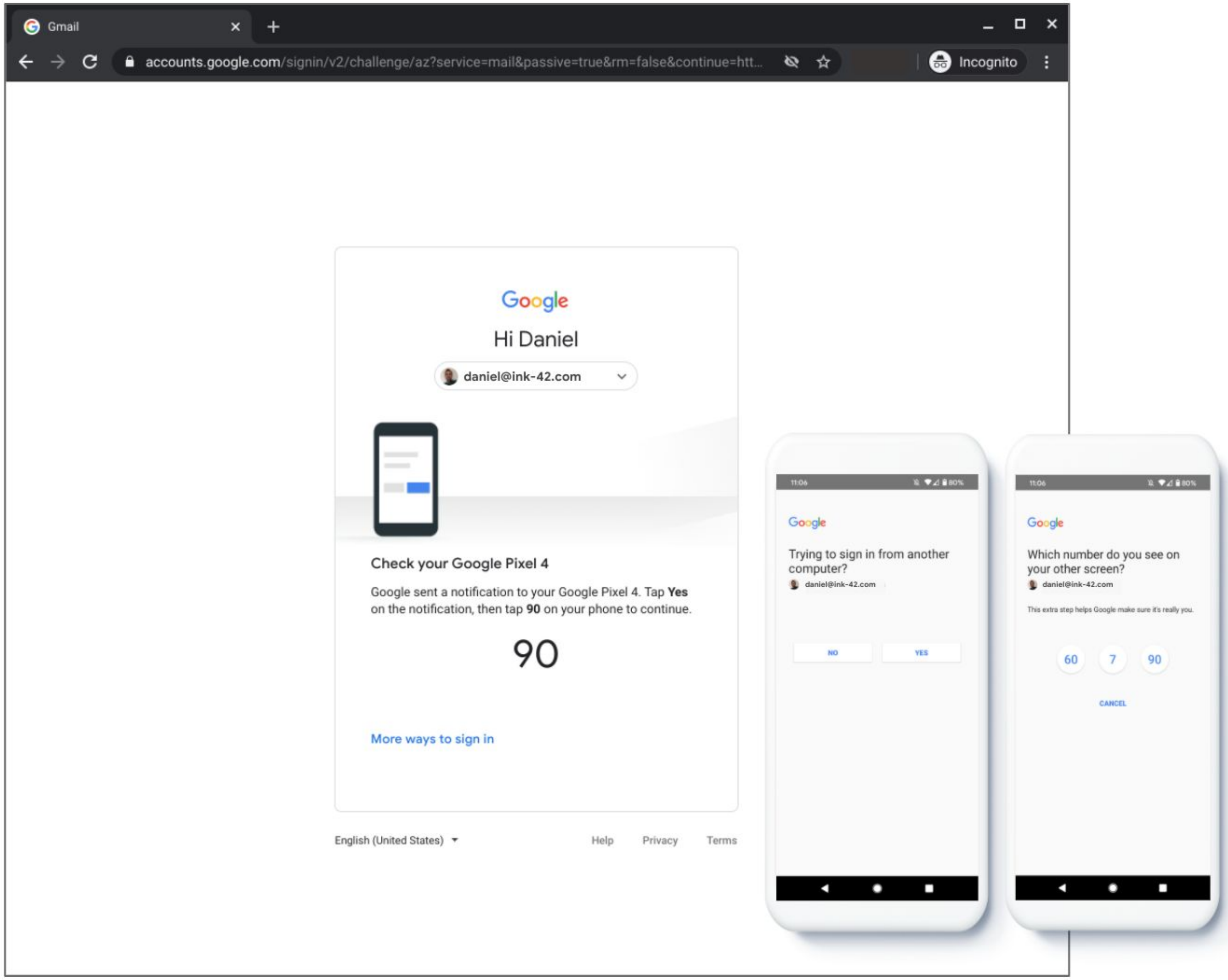
Qualified Electronic Signature (QES)



One-time passwords and two-step verification

- A strong authentication combines two or more of:
  - Something you **know**
  - Something you **have**
  - Something you **are** (fingerprint)
- A combination of the first two is the most common and is known as **two-step verification**:
  - *You know*: A personal password
  - *You have*: A physical object such as a bank security token or a smartphone with a One-Time Password.

Two-step verification



Demo eIDAS Tools

https://ec.europa.eu/cefdigital/DSS/webapp-demo/sign-a-document

Decentralized Identity: DID and VCs

Keys as Identity

Mirko Zichichi

associated to

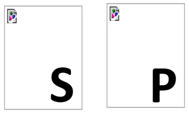
X1457cb4jiuKlo98hgf

Public Key

(alphanumeric)

How to create a new identity

Creating a new asymmetric key pair (***sk, pk***)

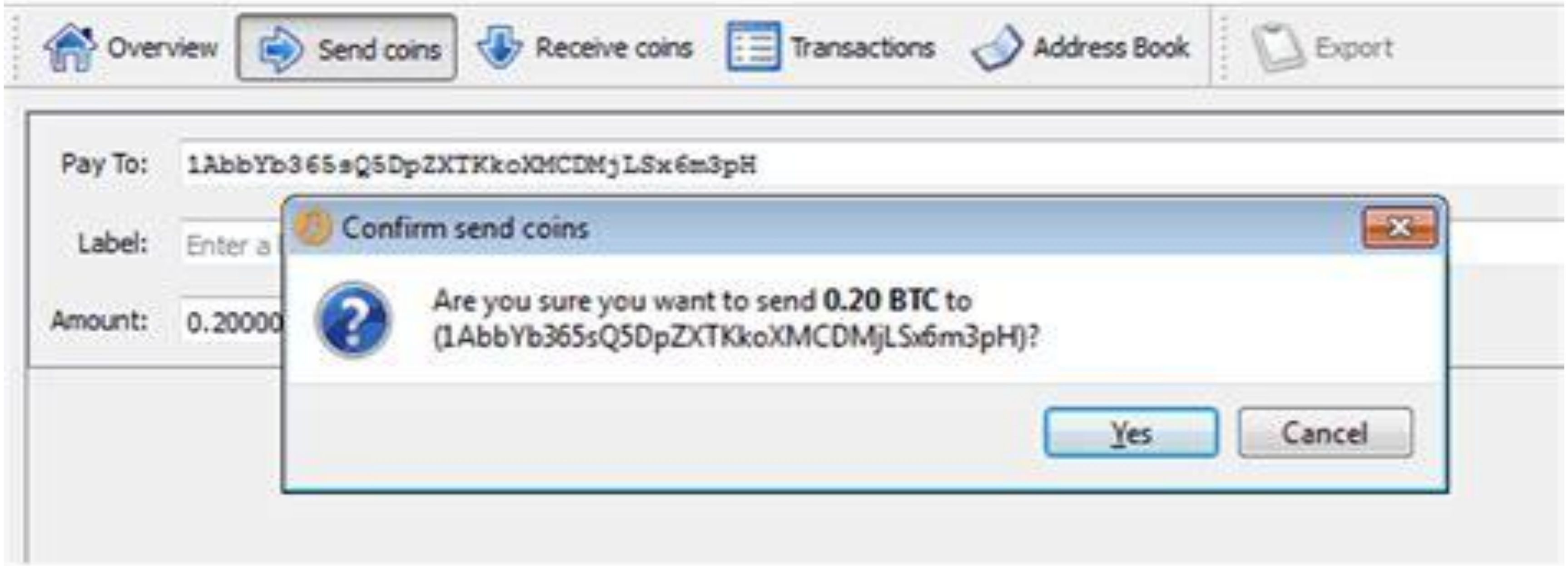


- The public key ***pk*** is the "name" of the identity
  - pseudonym
    - If ***pk*** "looks random", no one can associate it with an individual
  - often called an address
    - better to use Hash(***pk***) as the address
- The private key ***sk*** allows you to "speak for" the identity (sign)
  - Who knows the ***sk*** can control the identity

Decentralized identity management

- Anyone can create a new identity at any time and make as many as they like!
- The probability of generating the same key as another user is negligible
- No central coordination point
- No central authority to register identities in the system
- These "decentralized identities" are called "addresses" in Bitcoin

Decentralized identity management



Decentralized Identifier (DID)

- A type of identifier that enables a **verifiable and decentralized digital identity**
- They are based on the **Self-Sovereign Identity** paradigm
- A DID identifies any **subject** (e.g. a person, an organisation, a thing, a data model, etc.).
- These identifiers are designed to allow the **controller** of a DID to demonstrate control over it
- Allow to be implemented independently of any centralised registry, identity provider or certification authority

Decentralized Identifier (DID)

A DID is a simple text string consisting of three parts:

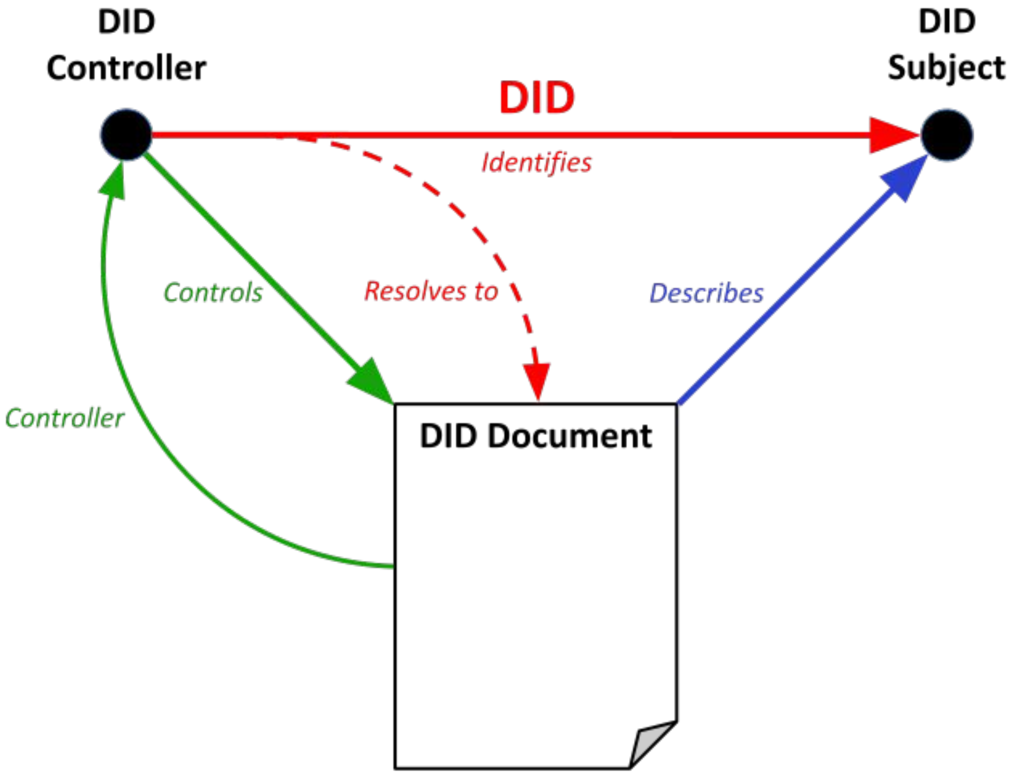
1. the did URI scheme identifier
2. the identifier for the DID method
3. the DID method-specific identifier.

https://www.w3.org/TR/did-core/

DID Document

The DID resolves to a **DID document**

A DID document contains information associated with the DID, such as ways to cryptographically authenticate a DID controller.



DID Document

The standard elements of a DID doc

1. **DID** (for self-description)
2. **Set of public keys** (for verification)
3. **Set of auth methods** (for authentication)
4. **Set of service endpoints** (for interaction)
5. **Timestamp** (for audit history)
6. **Signature** (for integrity)

Example


```
{
  "@context": ["https://w3id.org/did/v0.11", "https://w3id.org/btcr/v1"],
  "id": "did:btcr:xyv2-xzpq-q9wa-p7t",
  "publicKey": [
    {
        "id": "did:btcr:xyv2-xzpq-q9wa-p7t#satoshi",
        "controller": "did:btcr:xyv2-xzpq-q9wa-p7t",
        "type": "EcdsaSecp256k1VerificationKey2019",
        "publicKeyBase58":
                        "owh12LKNuphe97teJTZKQTKNewSVTwjHcskPbq34epCY"
     },
     {
        "id": "did:btcr:xyv2-xzpq-q9wa-p7t#vckey-0",
        "controller": "did:btcr:xyv2-xzpq-q9wa-p7t",
        "type": "EcdsaSecp256k1VerificationKey2019",
        "publicKeyBase58": "owh12LKNuphe97teJTZKQTKNewSVTwjHcskPbq34epCY"
     }
  ],
  "authentication": ["#satoshi"],
  "assertionMethod": ["#vckey-0"],
  "service":
   {
     "id": "did:btcr:xyv2-xzpq-q9wa-p7t#CRS",
     "type": "BTCR-CredentialRepositoryService",
    "serviceEndpoint":
"https://github.com/ChristopherA/self/blob/master/ddo.jsonld?hl=z87f623hkjh578v76dsf873h"
   },
}
```


https://w3c-ccg.github.io/didm-btcr

DID Legal Perspective

- If the user creating a DID is a natural person
  -> the DID will be considered as a pseudonym (constituting personal data) and, therefore, a data that must be compliant with GDPR.

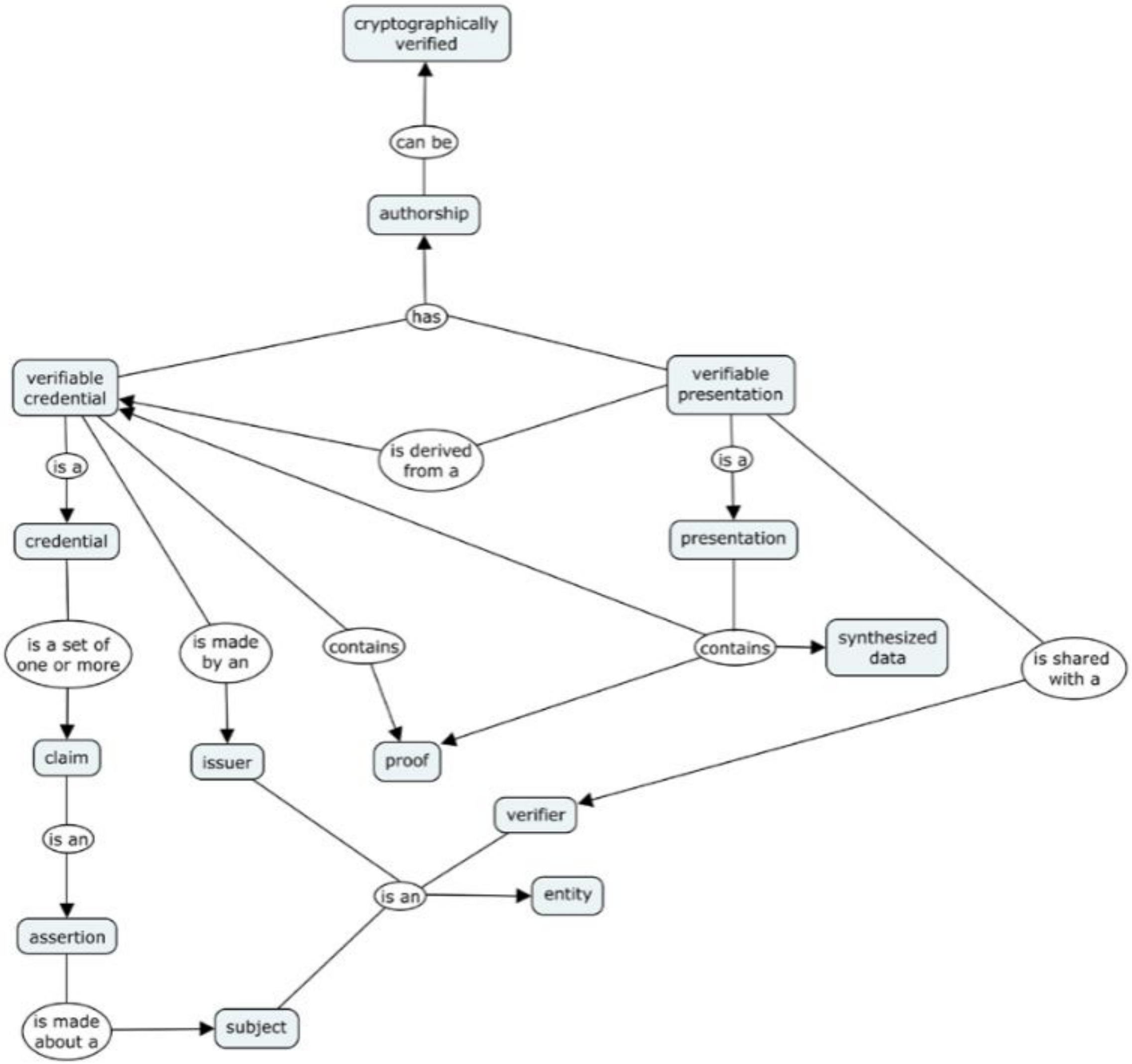
- If the DID is created by a legal person, for itself or for a thing it owns -> it will probably be considered as an asset property of the legal person.

- A DID is under the control of its "owner" because of the existence of a mechanism to assure that control, i.e. public key cryptography
  -> ***"this functionality is supporting the usage of Verifiable IDs as "electronic identification means" in the sense of the eIDAS Regulation"***

Verifiable Credentials

- Upon DID documents can be used:
  verifiable credential sharing syntaxes
- The user can obtain credentials claiming:
  - **identity attributes**, *issued* by **entities** that have previously *verified* them, and *share* them with **third parties**.
- **Self-Sovereign Identity (SSI)** -> complete control of individuals' digital identities and their personal data through disintermediation (or decentralization)
- Enables any subject to share information with third parties by proving to those the ownership of certain attestations or attributes, that are self-asserted or issued by trusted entity
  - **Trust** -> the subjective viewpoint of an individual who has a measure of confidence in another individual or entity

Verifiable Credentials



(Alamillo Domingo, 2019)

SSI Roles

- **Issuer**: issues verifiable credentials
- **Holder**: stores verifiable credentials securely under its own control
- **Verifier**: requests verifiable credentials and verify those
- **Validator**: validates data (e.g. for issuing, exchanging, revoking credentials)

Components [1/2]

- **Issuer component**
  - for *issuing* of **credentials**, i.e. sets of (related) claims that have: (i) *metadata*, e.g. date of issuing, (ii) a *digital signature* by which third parties can validate them.
  - for the *revocation* of **credentials** issued

- **Wallet component**
  - for (secure) *storage* of **credentials**
    - -> **self-signed** credentials
    - -> those obtained from **other parties' issuers**
  - for (secure) *storage* of **(private) keys** that can be used to sign or seal data on behalf of its principal

Components [2/2]

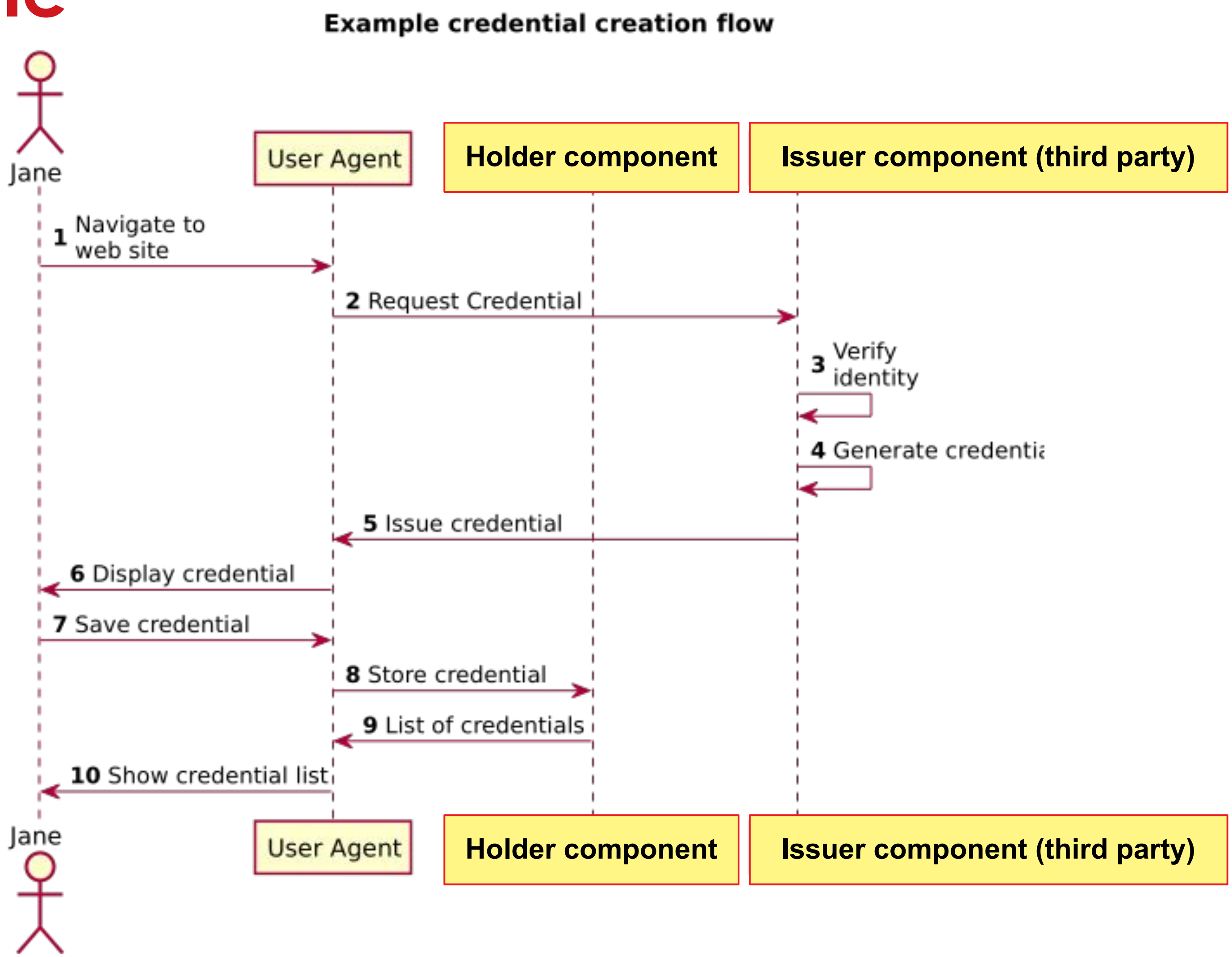
- **Verifier component**

supports the data collector in *acquiring* credentials from other parties. It does so by:

1. *creating* **requests** that ask for such credentials
2. *sending* them to a holder component of a party
3. *receiving* a **response** to such a request
4. *verifying* the **credentials**, i.e. checking the signature and other proofs of the veracity of the claims

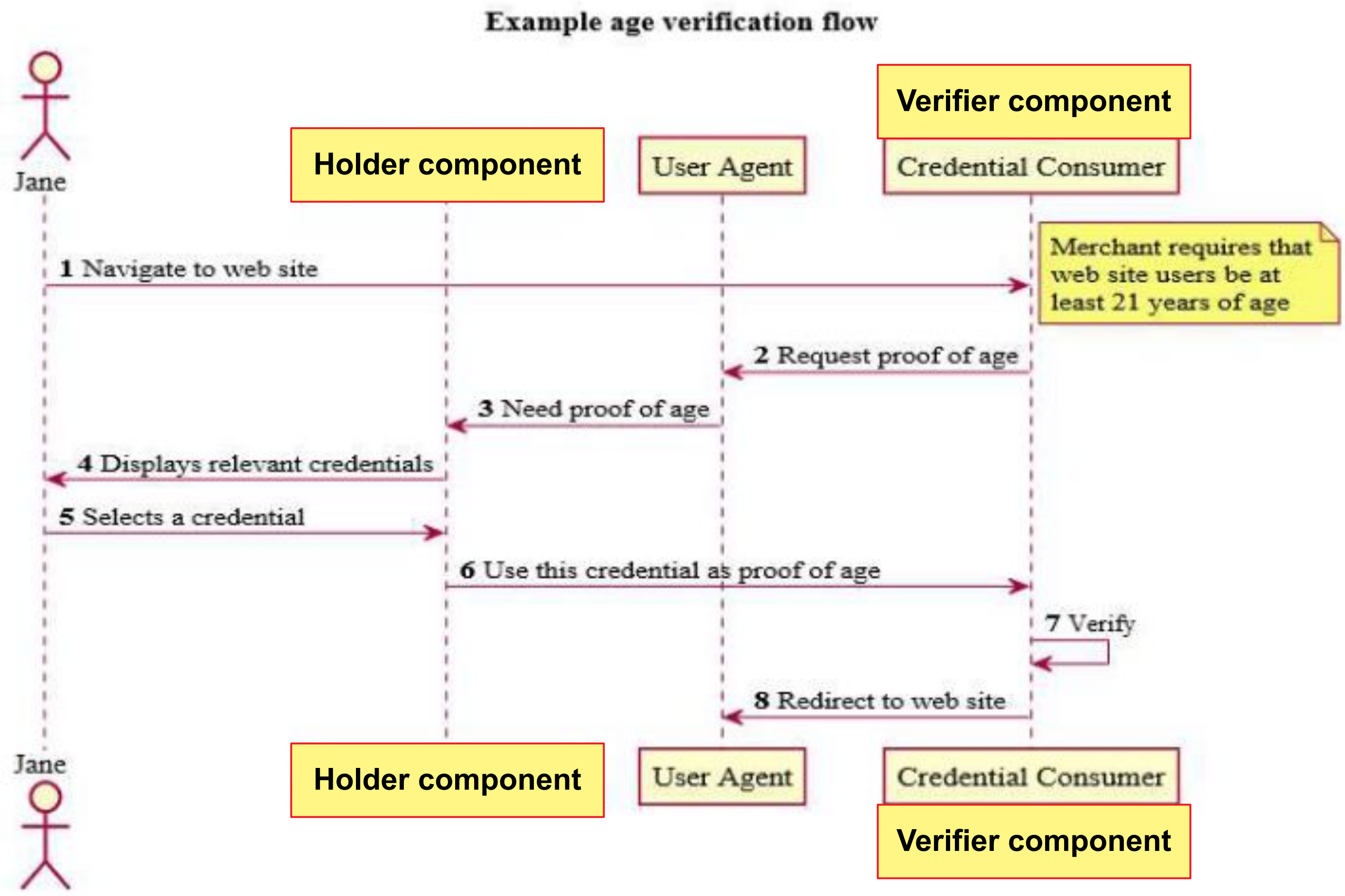
- **Holder component**
    - *handles* **credentials requests** received from a verifier
    - *looks* for the requested data in the wallet
    - if the wallet doesn't have it, the holder may *negotiate* a transaction with an issuer for obtaining these

Example



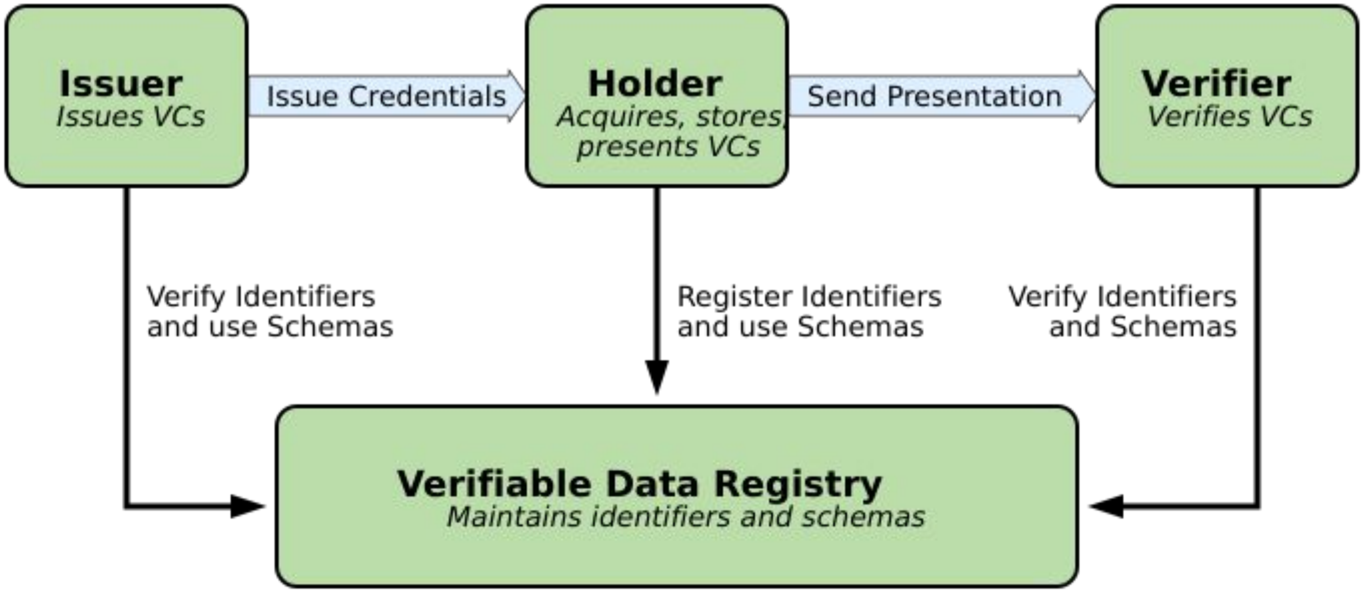
(Alamillo Domingo, 2019)

Example

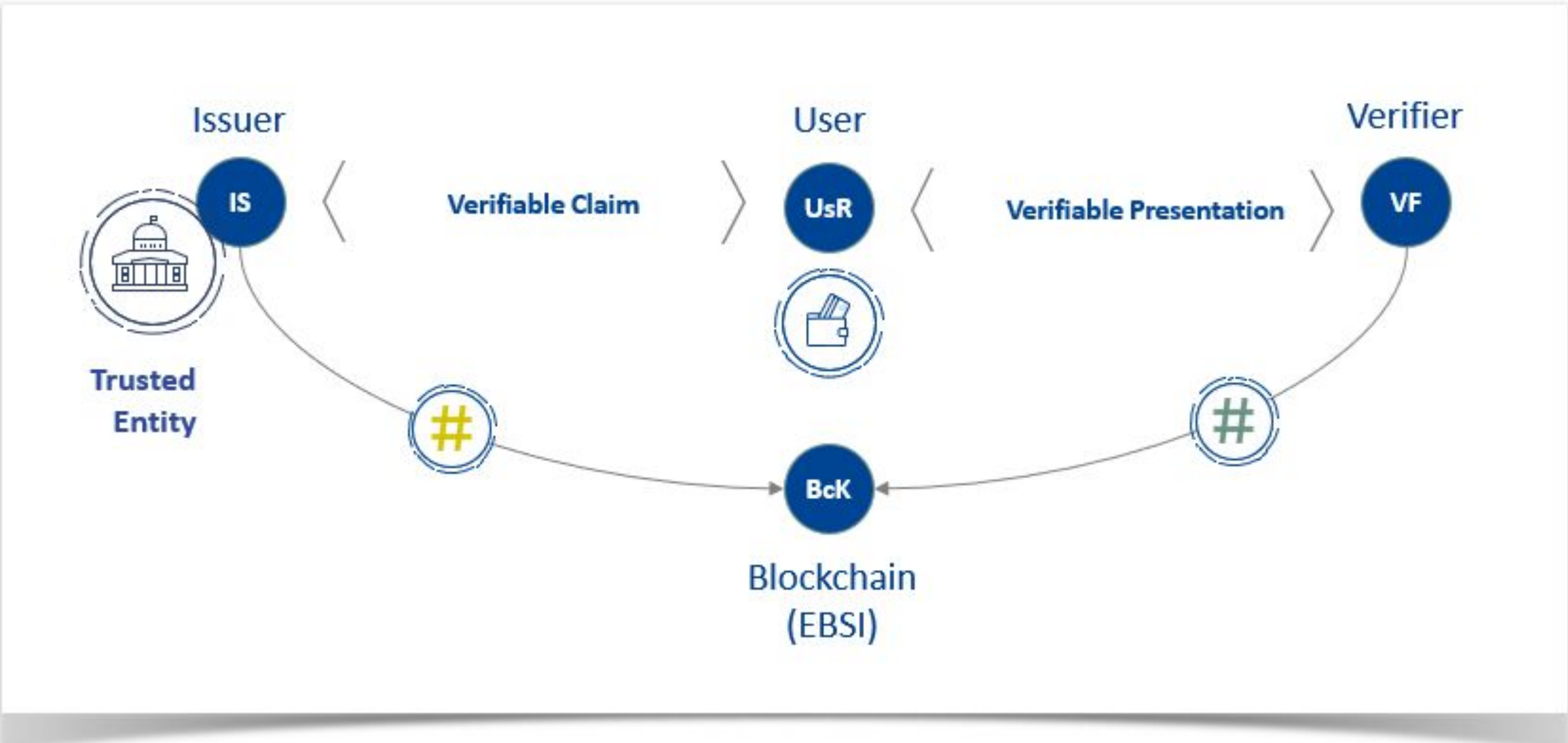


(Alamillo Domingo, 2019)

VC Ecosystem



EBSI: European Blockchain Services Infrastructure



(Alamillo Domingo, 2019)

eIDAS 2

EU Digital Identity

*"Every time an App or website asks us to create a new **digital identity** or to easily log on via a **big platform**, we have no idea what happens to our data in reality. That is why the Commission will soon propose a secure **European e-identity**. One that we trust and that any citizen can use anywhere in Europe to do anything from paying your taxes to renting a bicycle. A technology where **we can control ourselves what data and how data is used**."*

Ursula von der Leyen, President of the European Commission, in her State of the Union address, 16 September 2020.

eIDAS 2

June 6th, 2021

new eIDAS regulation proposal -> **eIDAS 2**

(currently under revision by Member States)

- Brings **EU Digital identities** one step closer to **Self-Sovereign Identity** principles
- Member States obliged to provide certified digital wallets to citizens
- Businesses will also have to accept them as forms of identification

#1: All EU Member States must provide a digital wallet to its citizens

"The draft Regulation requires Member States in Article 6a to issue a European Digital Identity Wallet under a notified eID scheme to common technical standards [...]

European Digital Identity Wallets shall be issued:

(a) by a Member State;

(b) under a mandate from a Member State;

(c) independently but recognised by a Member State."

#2: The creation and verification of verifiable credentials and the registry of electronic data in a DLT are now regulated Trust Services

“‘trust service’ means an electronic service normally provided against payment which consists of:

(a) the creation, verification, and validation of […] electronic attestation of attributes and certificates related to those services; […]

(f) the recording of electronic data into an electronic ledger.’;”

#3: The wallet will be allowed for use in the private sector, and will be mandatory for private parties providing services where strong user authentication for online identification is required

"areas of transport, energy, banking and financial services, social security, health, drinking water, postal services, digital infrastructure, education or telecommunications"

"private relying parties shall also accept the use of European Digital Identity Wallets issued in accordance with Article 6a."

#4: The possibility for European countries to accept Credentials from abroad, even from non-EU countries

“Article 14 - “[…] trust services provided by providers established in the third country concerned shall be considered equivalent to qualified trust services provided by qualified trust service providers established in the Union.’;”

#5: All wallets must technically enable selective disclosure of attributes to relying parties

"The European Digital Identity Wallet should technically enable the selective disclosure of attributes to relying parties. This feature should become a basic design feature thereby reinforcing convenience and personal data protection including minimisation of processing of personal data."

#6: Wallets must enable the storage of qualified and non qualified credentials, and allow signatures with Qualified Electronic Signature

“European Digital Identity Wallets shall enable the user to:

(a) securely request and obtain, store, select, combine and share, in a manner that is transparent to and traceable by the user, the necessary legal person identification data and electronic attestation of attributes to authenticate online and offline in order to use online public and private services;

(b) sign by means of qualified electronic signatures.sign by means of qualified electronic signatures.”

Questions Example

1. What are the levels, simple, advanced and qualified of electronic signatures in the eIDAS?

2. How can someone “control” a decentralized identity?

3. What are the properties provided by a digital signature and how are they conveyed?

References

- William Stallings, Lawrie Brown, *Computer Security: Principles and Practice (3rd Edition)* https://www.cs.unibo.it/~babaoglu/courses/security/resources/documents/Computer_Security_Principles_and_Practice_(3rd_Edition).pdf
- https://ec.europa.eu/digital-building-blocks/wikis/collector/pages.action?key=ESIGKB
- https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Standards+and+specifications
- https://www.w3.org/TR/did-core/
- https://w3c-ccg.github.io/didm-btcr/
- https://joinup.ec.europa.eu/sites/default/files/document/2020-04/SSI_eIDAS_legal_report_final_0.pdf
- Preukschat, Alex, and Drummond Reed. *Self-sovereign identity: Decentralized digital identity and verifiable credentials*. Manning Publications, 2021.
- https://www.w3.org/TR/vc-data-model
- https://ec.europa.eu/futurium/en/system/files/ged/eidas_supported_ssi_may_2019_0.pdf
- https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52021PC0281
- https://gataca.io/blog/here-s-what-the-new-eidas-proposal-really-means-for-the-ssi-community-in-6-key-points