Blockchain and Distributed Ledger

Information

- SSE curriculum, 2nd year, 1st semester, 6 CFU
  - Matteo Dell'Amico, matteo.dellamico@unige.it, office #228
  - Marina Ribaudo, marina.ribaudo@unige.it, office #231

Lectures
  - Tue 9:00 - 11:00 AM, room 214
  - Thu 9:00 - 11:00 AM, room 217

Exam rules
  - https://2023.aulaweb.unige.it/mod/page/view.php?id=51457

Course content

- Bitcoin, Blockchain, Script, Proof-of-Work,…
- Ethereum, Smart Contracts, Solidity, Exercises
- Permissioned blockchains
- Security and privacy: attacks, defenses, (de)anonymization
- Speed: level-2 payment networks
- …

Today lecture

Introduction

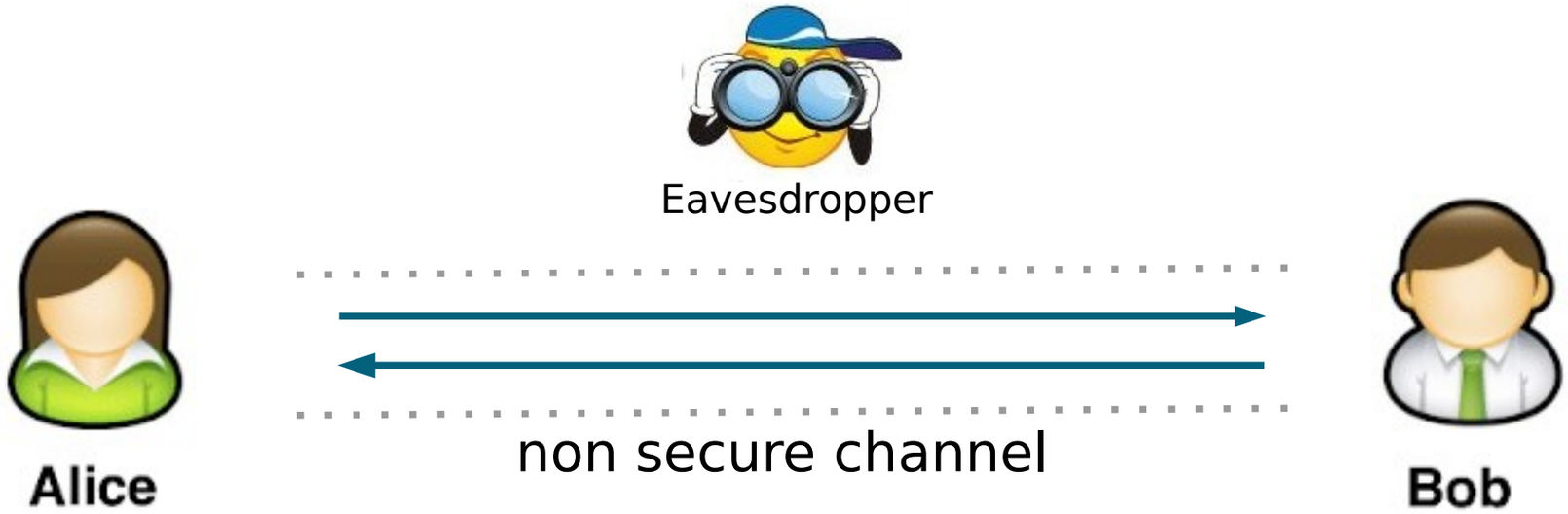
- Black box cryptography
- Currencies and cryptocurrencies
- Decentralization vs Distributed systems
- Web3
- Do you need a blockchain?

Black box cryptography

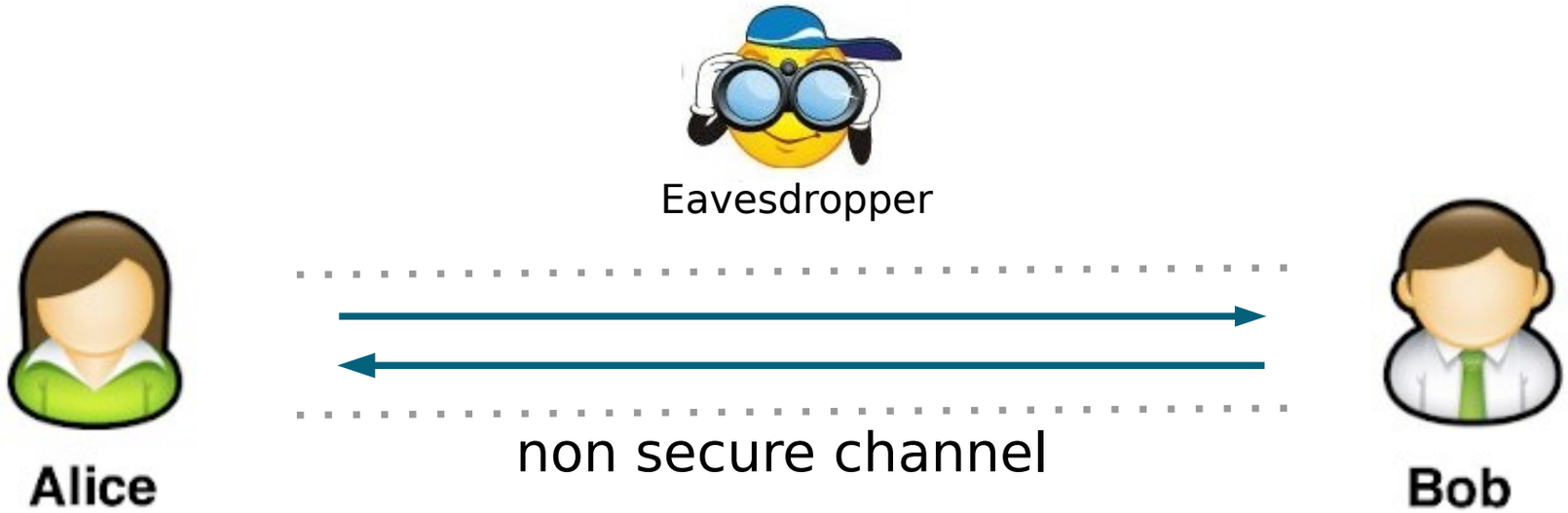
Definition

- Cryptography
  - from Ancient Greek **κρυπτός** (kryptós,"hidden, secret") and **γράφειν** (graphein, "to write")
  - techniques for **secure communication** in the presence of **adversaries**
- It is about protocols that **prevent third parties** from **reading private messages**
  - aspects such as **confidentiality**, **integrity**, **authentication** and **non-repudiation** are central to modern cryptography

Secure communication

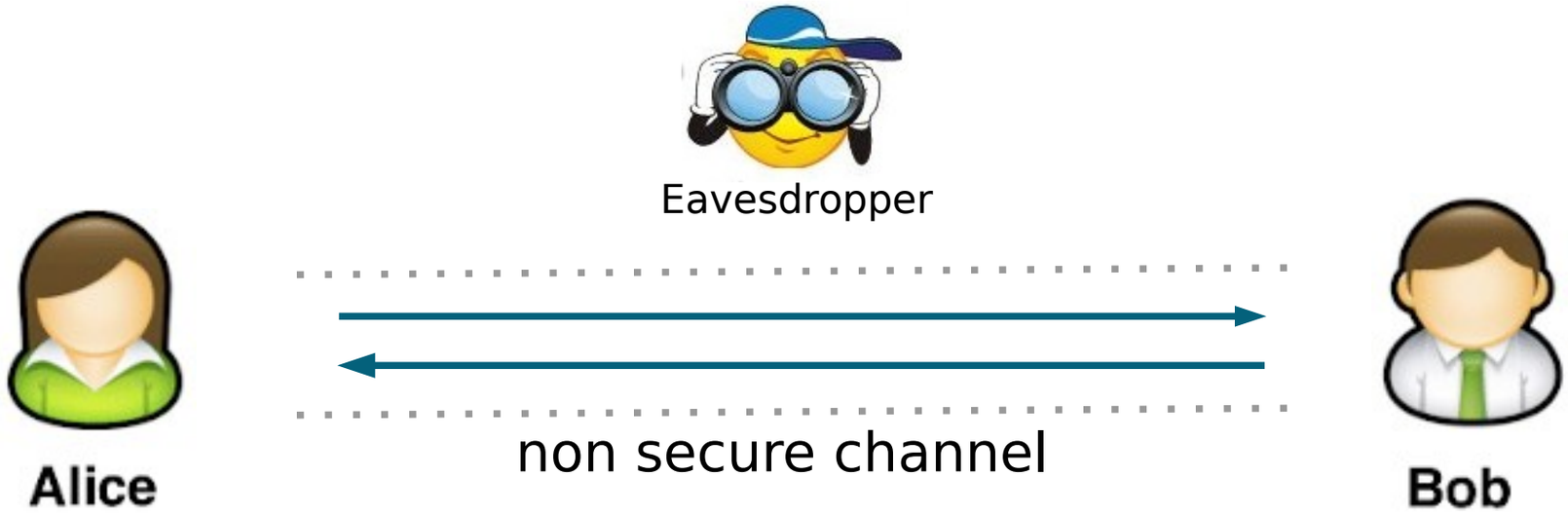


Secure communication



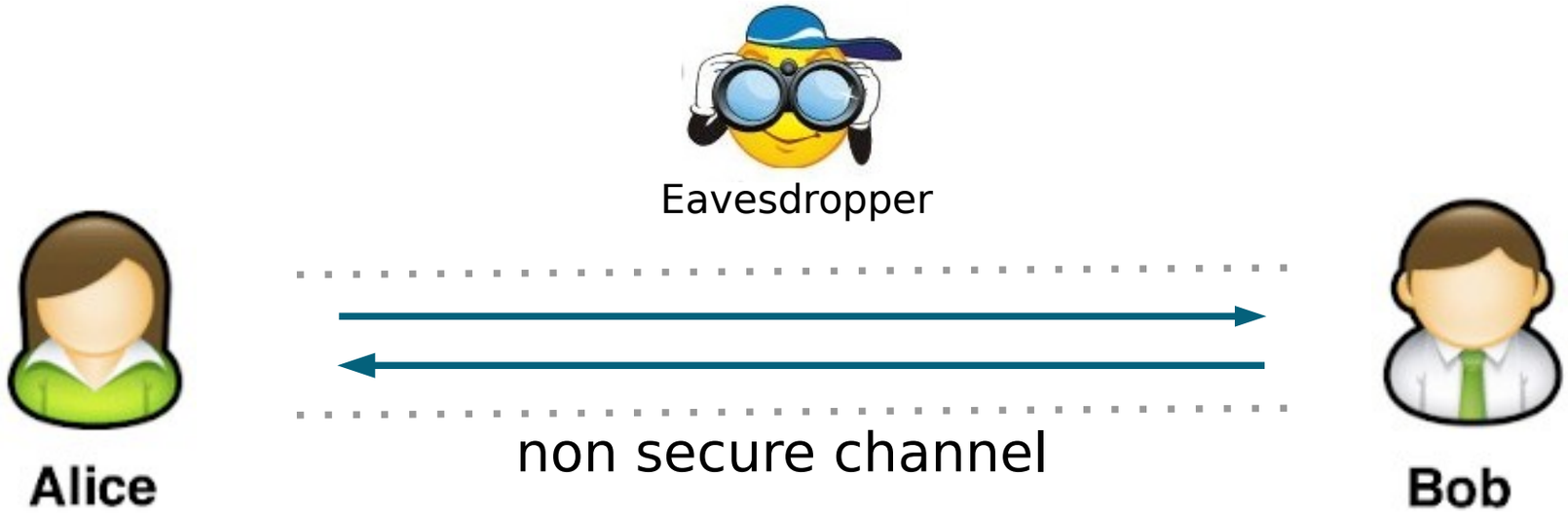
Alice and Bod could use a **secret algorithm** to **hide their communication**...

Secure communication



...but they use a **known algorithm**, and **hide the parameters**, e.g. the **keys**

Secure communication



The simplest algorithm is a **substitution cipher**, in which the letters or words in the message are substituted for others, based on a code shared in advance between the parties

Example: Caesar Cipher, or ROT 13, where the **key is n=13**

Two main methods

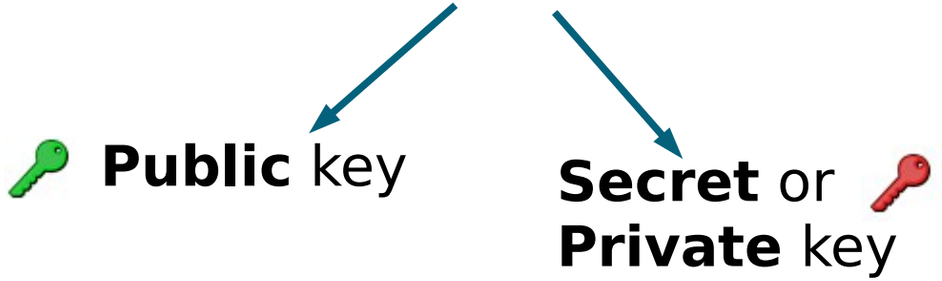
- 1. Symmetric cryptography
- 2. Asymmetric cryptography

Symmetric cryptography

- The **same key** is used to **encrypt** and **decrypt**
- It is **shared** by Alice and Bob
- They must **exchange** the key **in a secure way**
  - It might be difficult and costly
- Analogy: Alice and Bob have a **lock** and one copy each of **a key that opens/closes it**
  - Alice just puts her message in a box, passes it to Bob, only he can open the box

Asymmetric cryptography

- One key for encryption
- Another key for decryption
- The two keys must match, e.g., they are mathematically related
- Each user has a **pair of keys** (Pk, Sk)



Asymmetric cryptography

- **Both keys are required to perform an operation**
  - Data encrypted with the private key is deciphered with the public key
  - Data encrypted with the public key is deciphered with the private key
- Asymmetric cryptography is **often used to exchange the secret key** to prepare for using **symmetric cryptography** to encrypt data
- Encrypting data with the private key creates a **digital signature**

Analogy

- A **special lock** that is closed by one key and opened by another one
- **Encryption:** Alice makes the “closing key” public (everybody has a copy) and keeps the “opening key”
  - Bob puts the message in a box, closes it with the public key
  - Only Alice can open the box
- **Signature:** Alice makes the “opening key” public
  - She puts the message she sent in a box
  - If Bob can open the lock with Alice’s “opening” key, the message must come from her

Asymmetric cryptography

- Alice has a pair of keys (Pk_A , Sk_A)

Asymmetric cryptography

- Alice has a pair of keys (Pk_A , Sk_A)
  - Bob sends a message M to Alice, and uses her public key

    M ----------Pk_A 🔑----------→ C (**encrypt**)

Asymmetric cryptography

- Alice has a pair of keys (Pk_A , Sk_A)
  - Bob sends a message M to Alice, and uses her public key

    M ----------Pk_A  → C (**encrypt**)
  - Alice receives the ciphertext C, and uses her secret key

    C ----------Sk_A  → M (**decrypt**)

Asymmetric cryptography

- Alice has a pair of keys (Pk_A , Sk_A)
  - Bob sends a message M to Alice, and uses her public key

    M ----------Pk_A 🔑→ C (**encrypt**)
  - Alice receives the ciphertext C, and uses her secret key

    C ----------Sk_A 🔑→ M (**decrypt**)

- This guarantees **confidentiality**

Asymmetric cryptography

- **Everyone knows the public key and the algorithm**, but without the private key it is not possible to decrypt
- The algorithm uses a **one-way** function which is easy to compute, but difficult to invert
  - Examples
    - List of phone numbers
      - Easy to find a phone number for a given name (if names are sorted)
      - Given a phone number, difficult to find the corresponding name
    - Given two prime numbers, p and q
      - **Easy** to compute m = p*q, even for large values of p and q (**multiplication**)
      - Given m, **difficult** to find p and q (**factorization**)

Prime numbers

- Multiply is easy — 19 * 23 ?
- Factorize is difficult — 323 ? (semiprime)

- With large numbers is very difficult also for computers
- The algorithms to compute keys use large prime numbers, modular arithmetic, exponential and logarithmic operations...

Hash Functions

- Hash come from *hasher*, cut in small pieces, e.g. chop
  - Good representation of data
  - One-way
- Example in real life
  - From vegetables you can prepare a soup (**easy**)
  - From a soup is **difficult** to obtain the vegetables back
- Hash functions provide a way to map **data of arbitrary size** to **fixed-size values**

Hash Functions

Message M of **arbitrary** length

h()

00100111010101

Digest h of **fixed** length

Hash Functions

mobydick.txt

sha256sum

4dffc0bf542de6751f3918787fdd5afadce51c3fdff062ab70349d0834a56395

Properties of h()

- 1. **Deterministic**
- 2. **Quick** to compute the hash
- 3. **One-way** (also called pre-image resistant)
  - Given y it is computationally infeasible to find m’ such that h(m’) = y
- 4. **Collision resistance**
  - It is computationally infeasible to find two messages m, m’ such that h(m) = h(m’)

Properties of h()

- 5. With **minor modifications** in the input, **h() changes totally**, e.g., you cannot “adjust” hashes
- Notice that
  - A big book with hundreds of pages can be hashed. By changing only one letter in one page, the new hash is totally different
  - It is possible to hash any kind of data (image, text, video); from the hash value it is not possible to recover the type of the original document
  - Hashes are similar to **fingerprints** in real life: from a fingerprint you **cannot recover information** of the person, you need to **compare** the fingerprint with others

Hash Functions

mobydick.txt

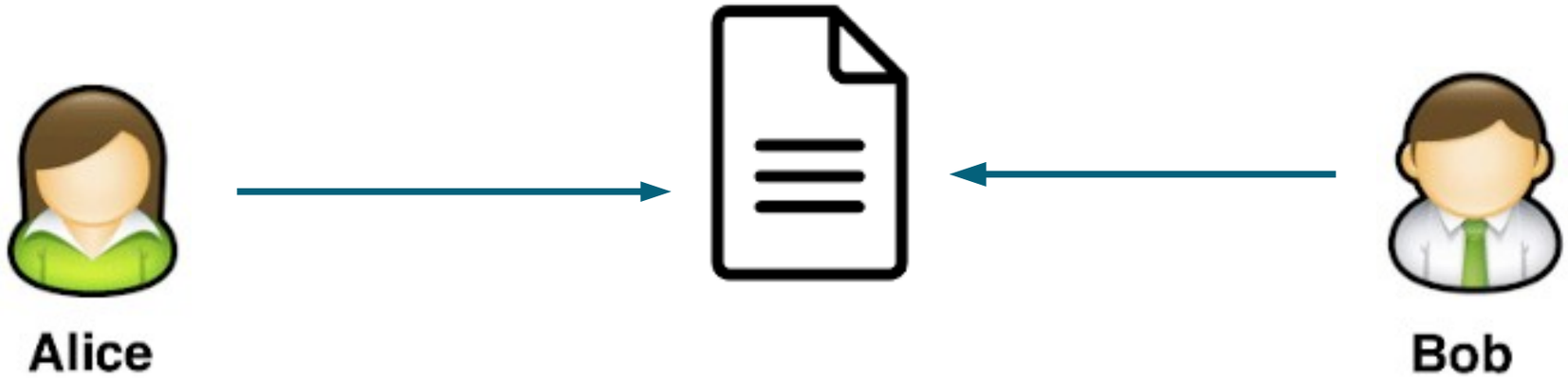
sha256sum

4dffc0bf542de6751f3918787fdd5afadce51c3fdff062ab70349d0834a56395

Adding one “.” in the file

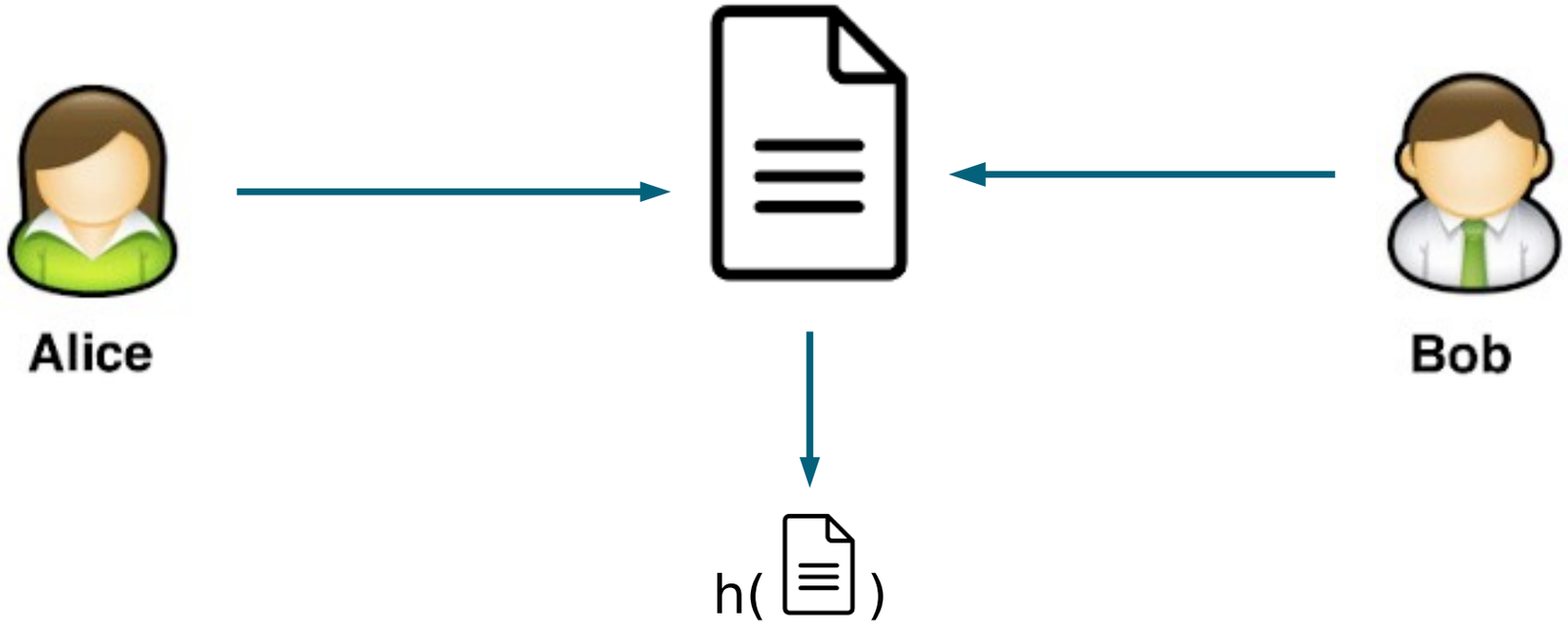
1eb64b7c370af468e85d320cd915a6aa71aa5d3eed6df32fd697c8111a3994e5

Example



Alice and Bod want to **share a contract** and they want to be sure that **none of them changes the contract**

Example



They can compute the **hash of the contract** and store it in a safe place. If there are changes in the contract, by applying again the hash function a different result is obtained (so it is possible to check)

Uses of h()

- **Integrity checks**
  - modifications on documents/messages can be malicious or can occur because of errors (data loss over the network)
- **Digital signatures**
  - combined use of asymmetric cryptography and hash functions

Digital signature

- A digital signature is a **sequence of bits** associated to a message to **authenticate the person who signed it**
- Bob receives a message from Alice and he wants to be sure that the message comes from her
- **Alice** is the **signer** and she has a pair of keys (Pk_A, Sk_A)
  - Alice will use the **secret key Sk_A** to **sign**
  - Bob can **verify the signature** with **Pk_A**

Digital signature

- **Alice** has a **message M** and computes **h(M)**
- To **sign** she applies $\mathbf{Sk_A}$ to h(M) and produces **sign(M)**
- She sends the signed message, e.g. pair **(M,sign(M))**

Digital signature

- **Alice** has a **message M** and computes **h(M)**
- To **sign** she applies $\mathbf{Sk_A}$ to h(M) and produces **sign(M)**
- She sends the signed message, e.g. pair **(M,sign(M))**

Digital signature

- **Bob** receives the signed message **(M,sign(M))**
- Splits the message **M** from the signature **sign(M)**
- **Extracts h(M)** from the signature using $\mathbf{PK_A}$
- **Computes his version of h(M)** from the message M just received and checks if they are equal

Digital signature

- **Bob** receives the signed message **(M,sign(M))**
- Splits the message **M** from the signature **sign(M)**
- **Extracts h(M)** from the signature using **PK_A**

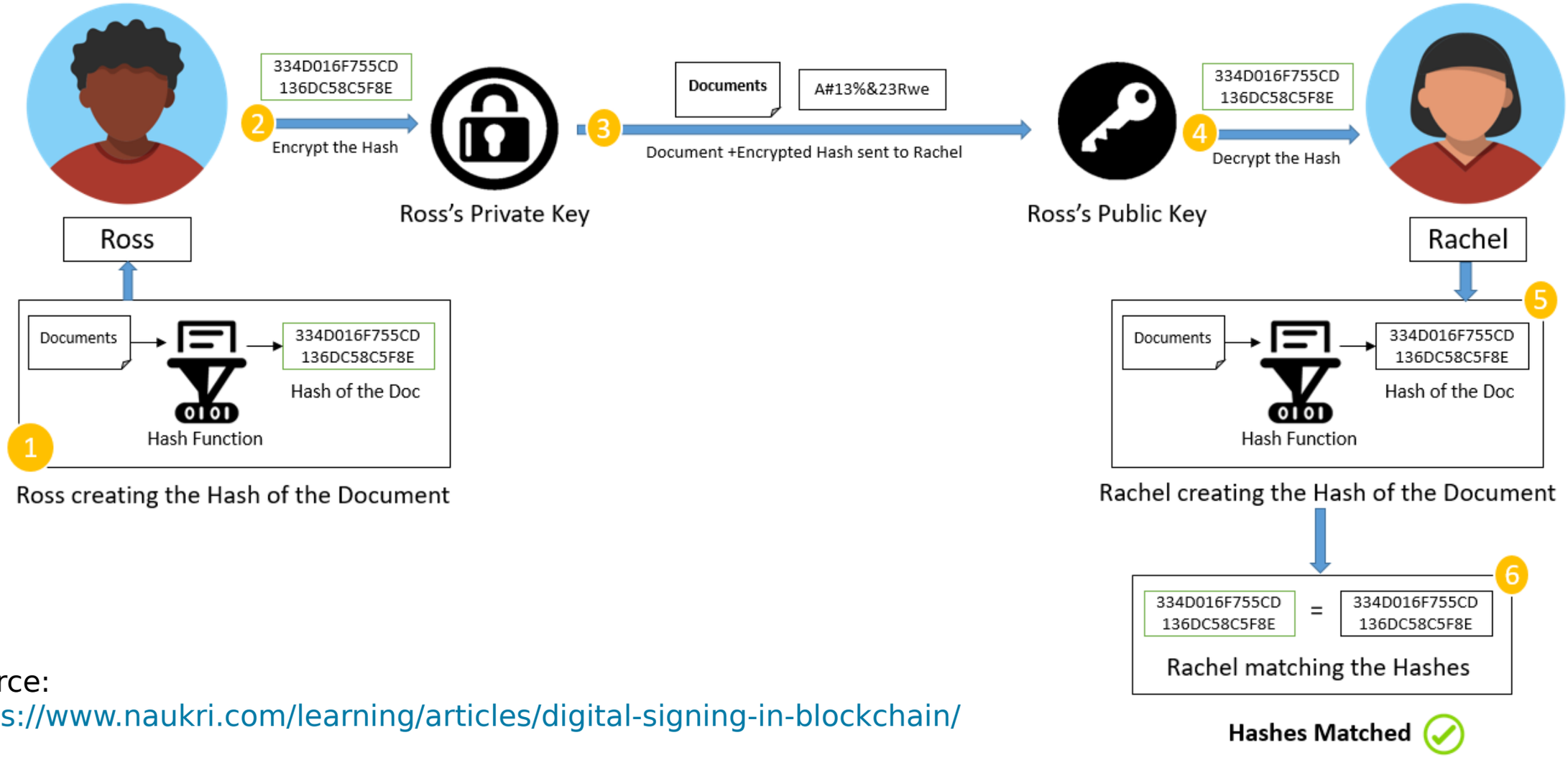
Digital signature

- If the two hashes are different
  - there is a problem with the identity of the sender (Alice)
  - or the message has been altered
- and the message is discarded

Digital signature

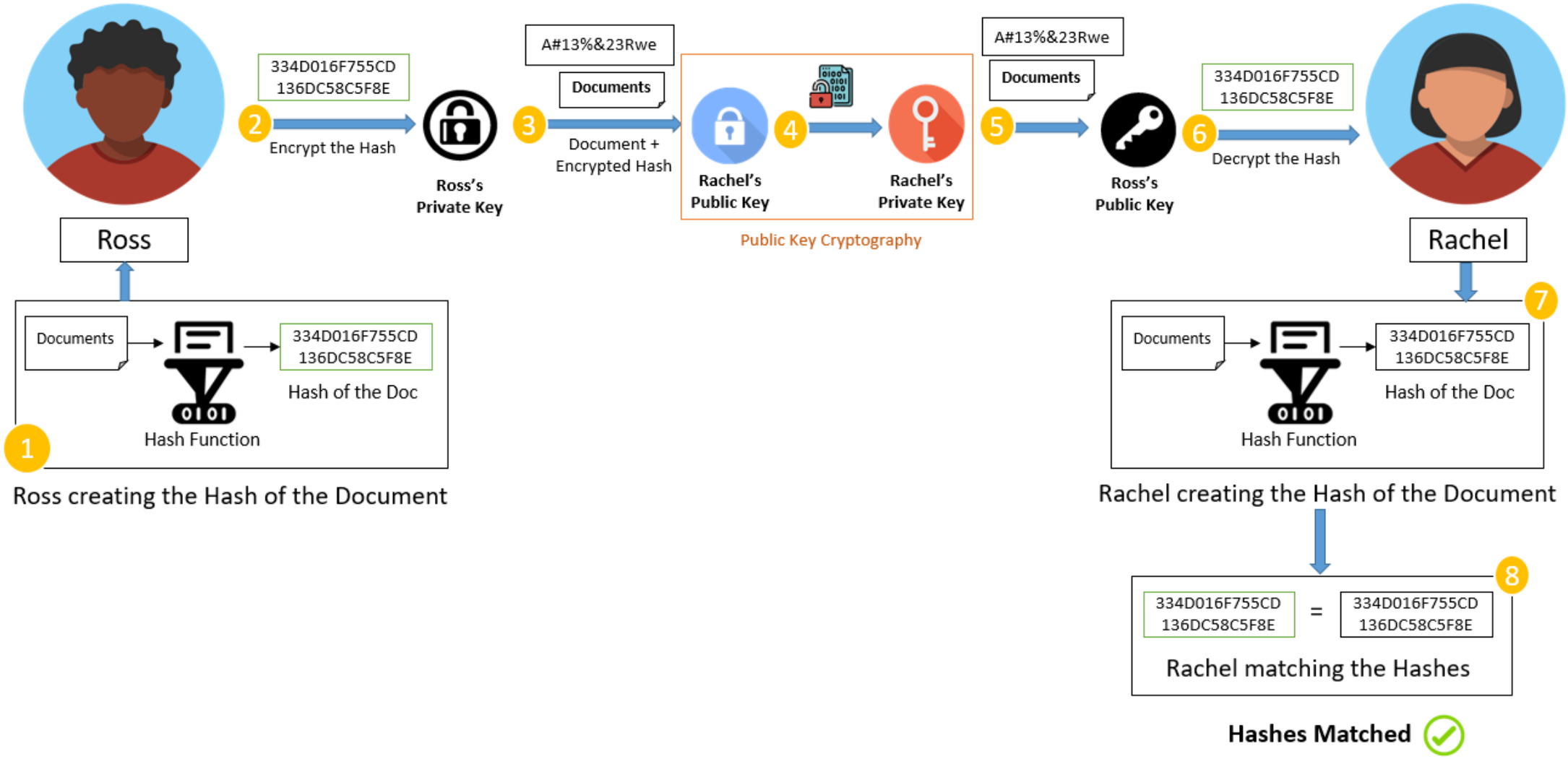
- 1. **Authentication**
  - it is possible to verify the identity of the sender using their public key
- 2. **Non repudiation**
  - the sender can not say they did not send a message signed with their private key
- 3. **Integrity**
  - modifications in the message invalidate the hash

Problem: anyone can read

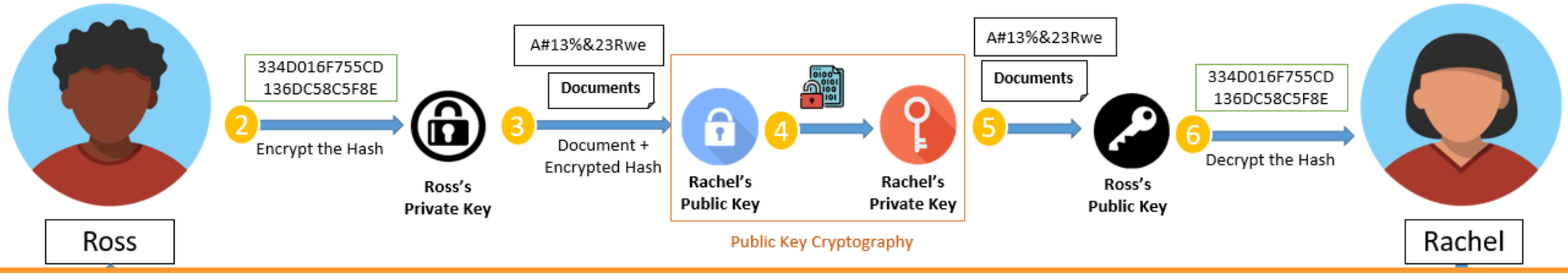


Source:
https://www.naukri.com/learning/articles/digital-signing-in-blockchain/

Solution: use $PK_{receiver}$



Solution: use $PK_{receiver}$



As we will see, **digital signatures** in blockchains (and elsewhere) are used to **authenticate transactions**

The user has to prove to the network that they are authorized to issue transactions and spend their balance

Hashes Matched

Currencies and cryptocurrencies

Currencies

- **Gift economy**
  - Valuables were not sold, but rather given without an explicit agreement for immediate or future rewards
- **Barter**
  - System of exchange in which participants in a transaction directly exchange goods or services for other goods or services
  - Reciprocal exchange, not delayed in time

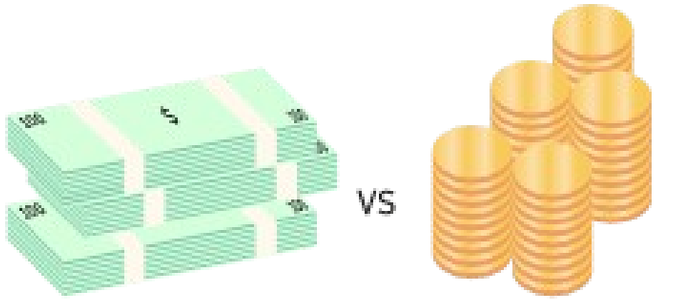
Currencies

- **Commodity money**
  - Derived from the commodity out of which the money is made

  - It has an **intrinsic value** and it was a convenient form of trade in comparison with the barter system

Currencies

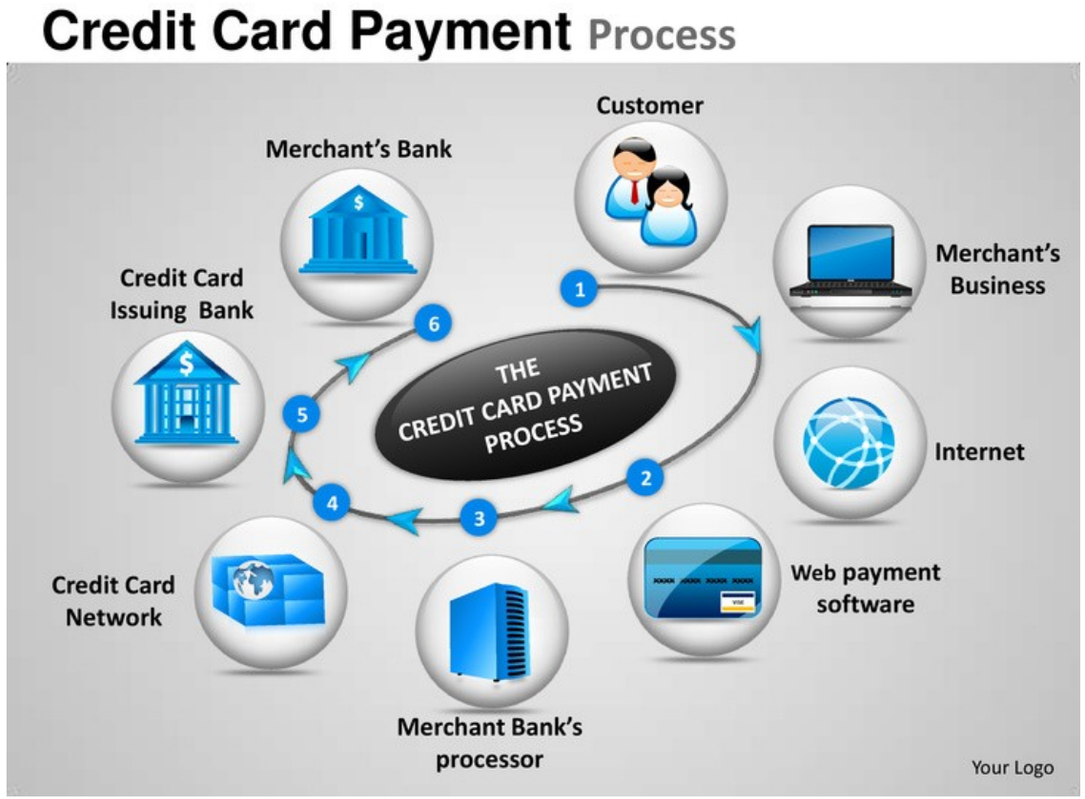
- **Fiat money**
  - Type of money that is **issued and regulated by the government**. Its most important feature is that it **has no intrinsic value of its own**, it holds value only because the government issues, maintains, and regulates it



FIAT MONEY VS COMMODITY MONEY

Currencies

- Having fiat-based currency requires a lot of third-party consensus to try to avoid frauds in the system
- For example



Ledger

- A **ledger** is a **record for economic transactions** that includes cash, accounts receivable, inventory, accounts payable, debt, costs, salaries, expenses, …
- It is the **primary record used by banks and other financial institutions to reconcile book balances**
- The financial statements of banks, financial institutions, and enterprises are compiled using ledger accounts

Ledger

- For financial transactions with fiat currency, **third-party trust systems** (VISA, MasterCard, banks) **maintain information** about every transaction on their ledgers
- Blockchain has changed the landscape making everyone part of the ledger…

Cryptocurrencies

- A cryptocurrency is a **digital currency**
- Digital means that there is nothing physical
  - It is possible to **pay** for objects and **exchange** with other crypto or fiat currencies
- Cryptocurrecies are **decentralized**, e.g. there are no central authorities
- **Maths replaces banks**
  - Instead of trusting some “entity”, you **trust cryptography**

Cryptocurrencies

- Other cryptocurrencies (who did not survive) were studied before Bitcoin
- Also PayPal was initially introduced as a cryptocurrency, but then it became the payment system we know today
- Suppose we want to create a cryptocurrency

Create a new coin

- Steps
  - Create new coins
  - Transfer coins

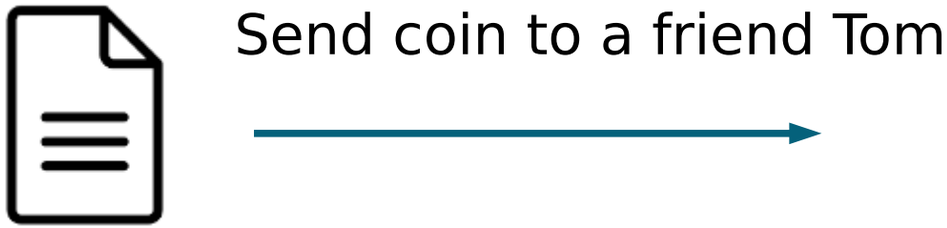
Create a new coin

- Steps
  - Create new coins
  - Transfer coins

Create a coin, e.g., a file

Create a new coin

- Steps
  - Create new coins
  - Transfer coins



Create a new coin

- Steps
  - Create new coins
  - Transfer coins

Send coin to a friend Tom

Is this a good model?

Create a new coin

- Steps
  - Create new coins
  - Transfer coins

Send coin to a friend Tom

Am I really the sender?

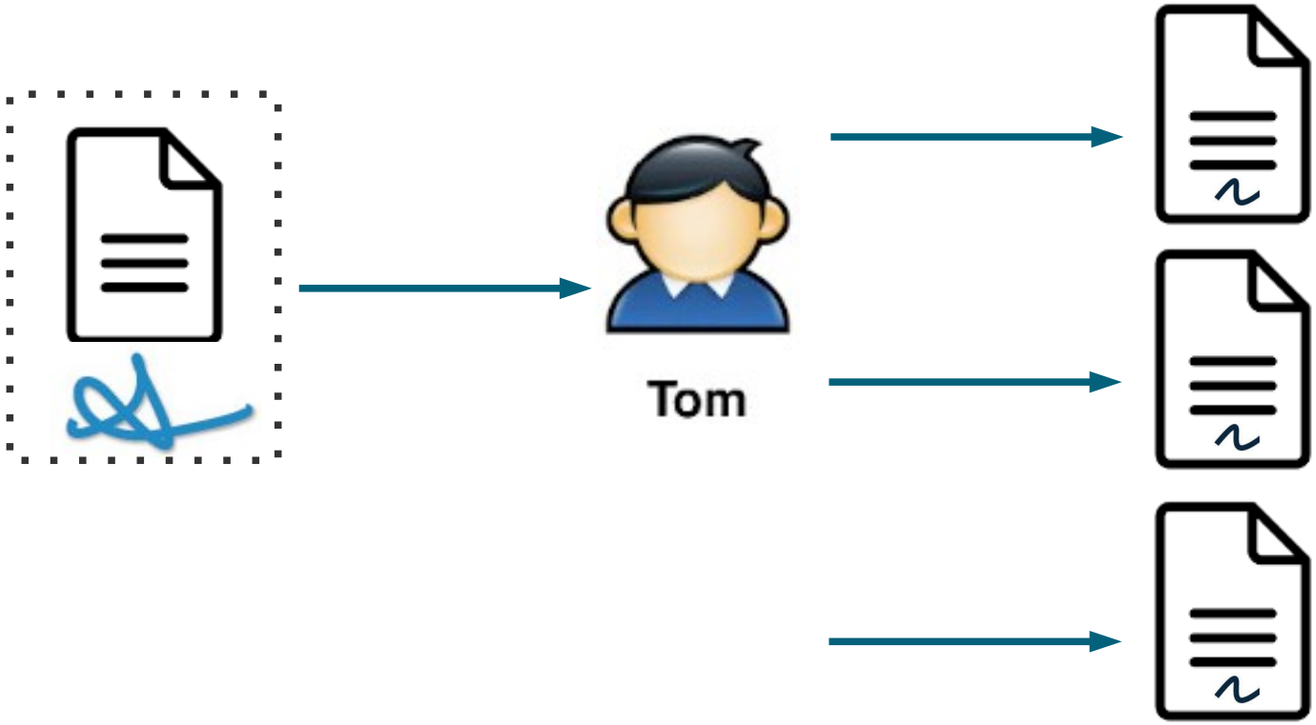
Create a new coin

- Steps
  - Create new coins
  - Transfer **signed** coins

Send **signed** coin to a friend Tom

Create a new coin

- Now suppose Tom cheats

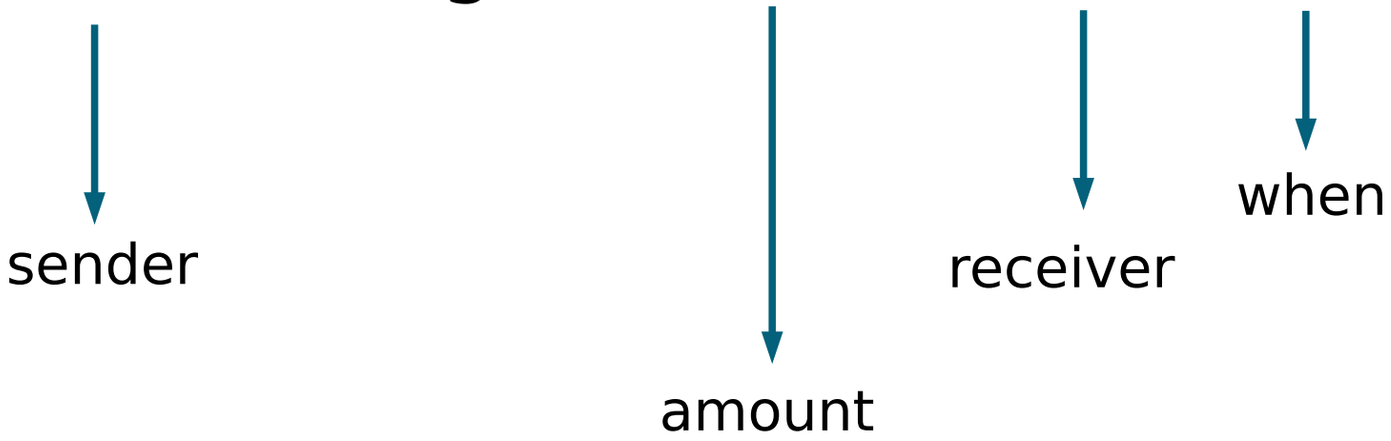


Tom **spends** the **same coin multiple times**

How can we avoid this?

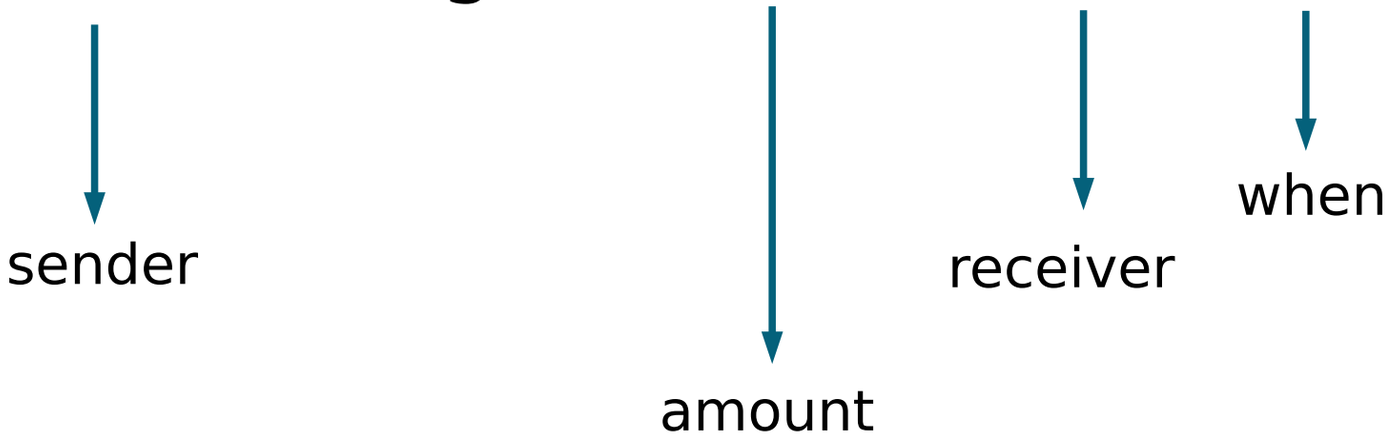
Create a new coin

- To avoid double spending **all actors** in the system must **communicate**
- “I am **Alice** and I gave **1 coin to Tom today**”



Create a new coin

- To avoid double spending **all actors** in the system must **communicate**
- “I am **Alice** and I gave **1 coin to Tom today**”



Problem: being in a network, messages can arrive late

Create a new coin

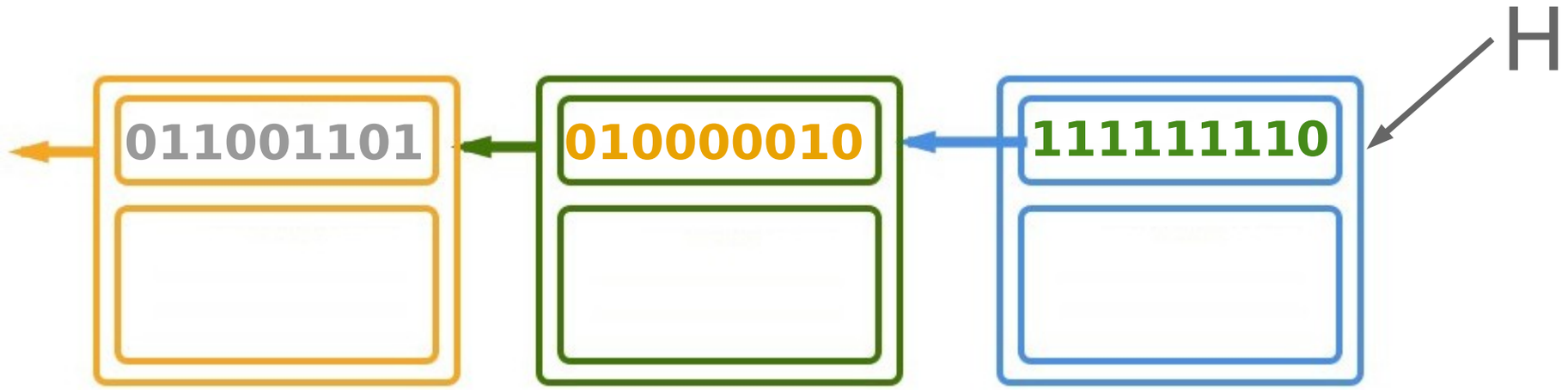
- We need to add some **other piece** of **cryptography**
  - All actors in the network can **hash** the **history** of the transactions
  - If **two actors know a different history**, theirs **hashes are different**, and they can check what happened
- Unfortunately, even with this solution, point-to-point communication is not realistic…

Create a new coin

- We can replace real identities with **digital identities**
  - **Public key** can be used, but it is too long
  - **Hash** the public key and get the **Address**
- We also need an **initial issuer** of the coins
- And we need some “entity” keeping track of the history

Create a new coin

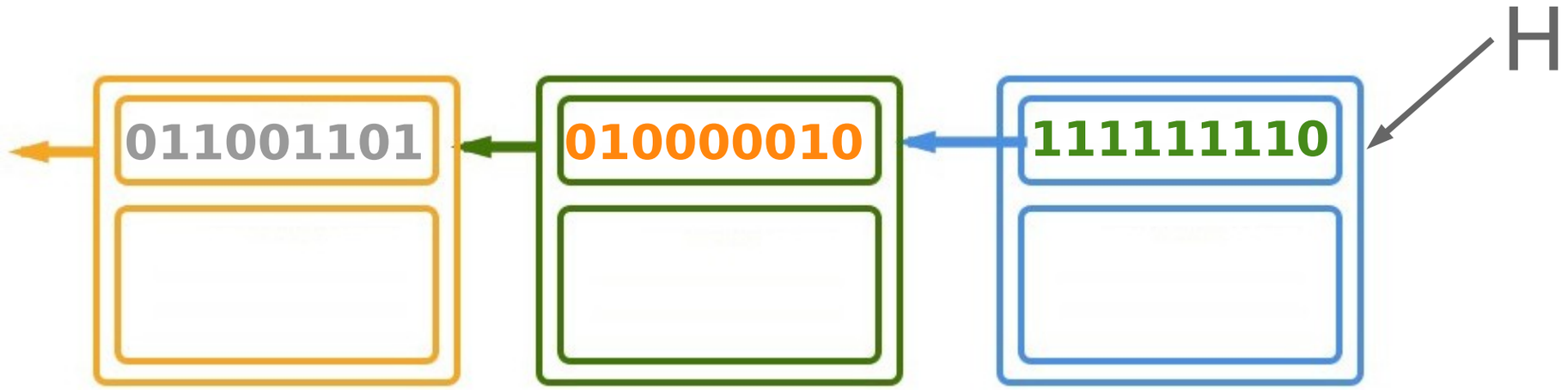
- This “entity” can organize coins transfers (transactions) in blocks



- Blocks are connected via **hash pointers**
  - Data structure that contains the **address** of a given information (a block in this case) and the **hash** of the information so that it is possible **to check the integrity of the information**

Create a new coin

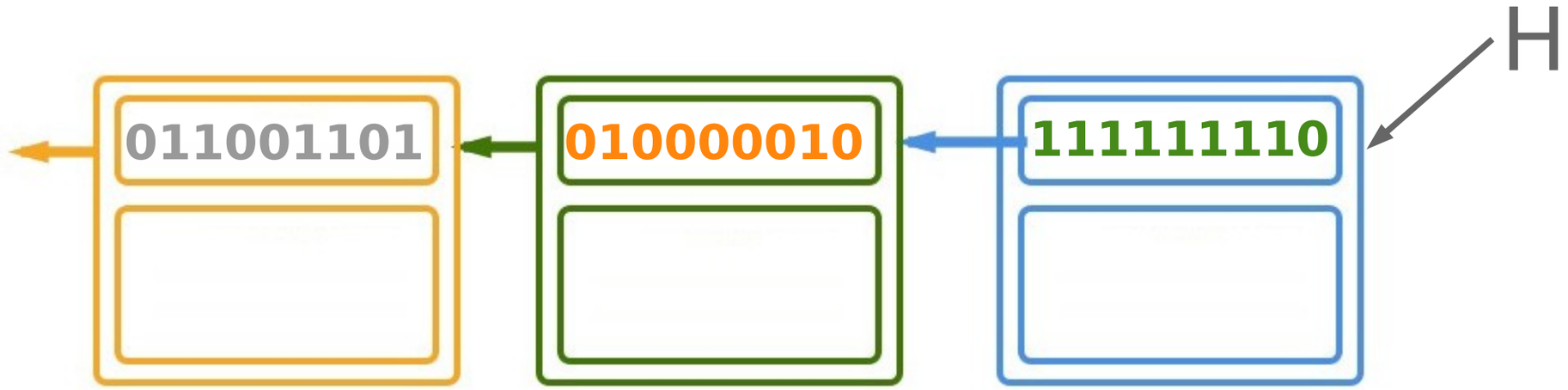
- This “entity” can organize coins transfers (transactions) in blocks



- Blocks contains issues of new coins and valid payments
- This is an **abstract model of a blockchain!**

Create a new coin

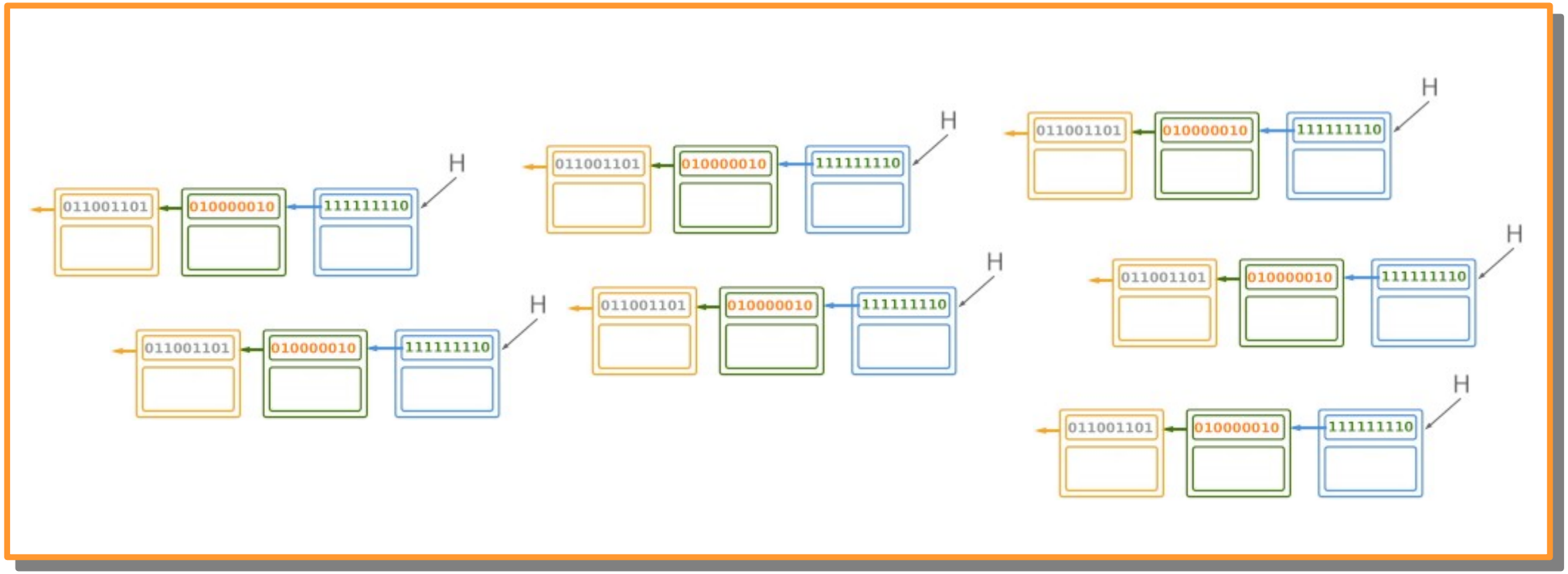
- This “entity” can organize coins transfers (transactions) in blocks



- A **problem** of this model is that the “entity” is too powerful and it is also a single point of failure
- Also, we need to trust the “entity”
- We will see how this was solved in the Bitcoin blockchain

Create a new coin

- **Multiple “entities”** share the same state of the system



Decentralized vs Distributed Systems

Distributed systems

- A **distributed system** is one in which the application and its architecture are distributed over a large number of machines, and possibly physical locations
- This means that multiple computers must **coordinate** to achieve the goals of the overall application

Distributed systems

- The **advantages** of distributed systems are many
  - **Resiliency**, e.g., the ability to adapt and keep working in presence of changes
  - **Redundancy**, e.g., each part of the system can be built to have backups so that if it fails, another copy can be used
  - **Parallelism**, e.g., work can be divided efficiently so that many less expensive computers can be used instead of a single (very expensive) fast computer

Decentralized systems

- All **decentralized systems must be distributed**, but distributed systems are not necessarily decentralized
- The difference has to do with location and redundancy versus **control**
- Metaverse, X, YouTube
  - Highly distributed services, with servers and data centers worldwide. They are distributed, with fault tolerance, extensive coordination, redundancy, and so on

Decentralized systems

- But
  - These services are still “centralized” because, with no input from other stakeholders, **they can change the rules**
  - Platforms others depend on, but with no reciprocity
- The last decade has seen the growth of many highly distributed, yet highly centralized companies, such as AirBnB, Uber, BlaBlaCar, and others

Decentralized systems

- In a decentralized system, there is no super powerful stakeholder with the ability to make and enforce rules without the permission of other network users
- To judge how decentralized a system is, the are several factors to consider

Decentralized systems

- **Open access**
  - At the beginning Internet was seen as revolutionary in part because of its open access: anyone can join
- **No hierarchy**
  - If each member in the system has identical power, control can be exercised through influence or reputation

Decentralized systems

- **Diversity**
  - A diverse system stands in opposition to monoculture, as was for instance Windows for a long time and today is probably Google for web searches
- **Transparency of operation**
  - If some actors have information dominance, e.g., access to greater information, they can centralize the system

Decentralized systems

- Decentralized system have their downsides
- **Speed**
  - Depending on the type of event, decentralized systems can be slower
  - The Bitcoin blockchain can handle approximately 7 transactions a second; VISA and MasterCard are distributed (but not decentralized) transaction-handling systems that can handle more than 50000 transactions a second

Decentralized systems

- Decentralized system have their downsides
- **Censorship resistance**
  - Decentralized systems tend to be much harder to censor because of a lack of central authority
  - This is not necessary a disadvantage but some information (child pornography, hate speech, bomb making instructions) is considered dangerous and immoral and should be censored
  - This is not always possible (for example it is not possible with blockchain)

Decentralized systems

- Decentralized system have their downsides
- **Chaos**
  - Decentralized systems tend to be more chaotic than centralized ones by their nature
  - Each actor works according to their own desires and not in response to an authority
- For all these reasons, decentralized systems are difficult to predict

Decentralized systems

- Blockchain is the base “ingredient” to foster decentralized (**Web3**) applications
- Enables the **Internet of Value**, e.g., an online space where people can instantly transfer value between each other, eliminating the need for the middlemen and most costs
- https://gatehub.net/blog/what-is-the-internet-of-value/

Do you need a blockchain?

Paper available at https://ieeexplore.ieee.org/document/8525392

Do you need a blockchain?

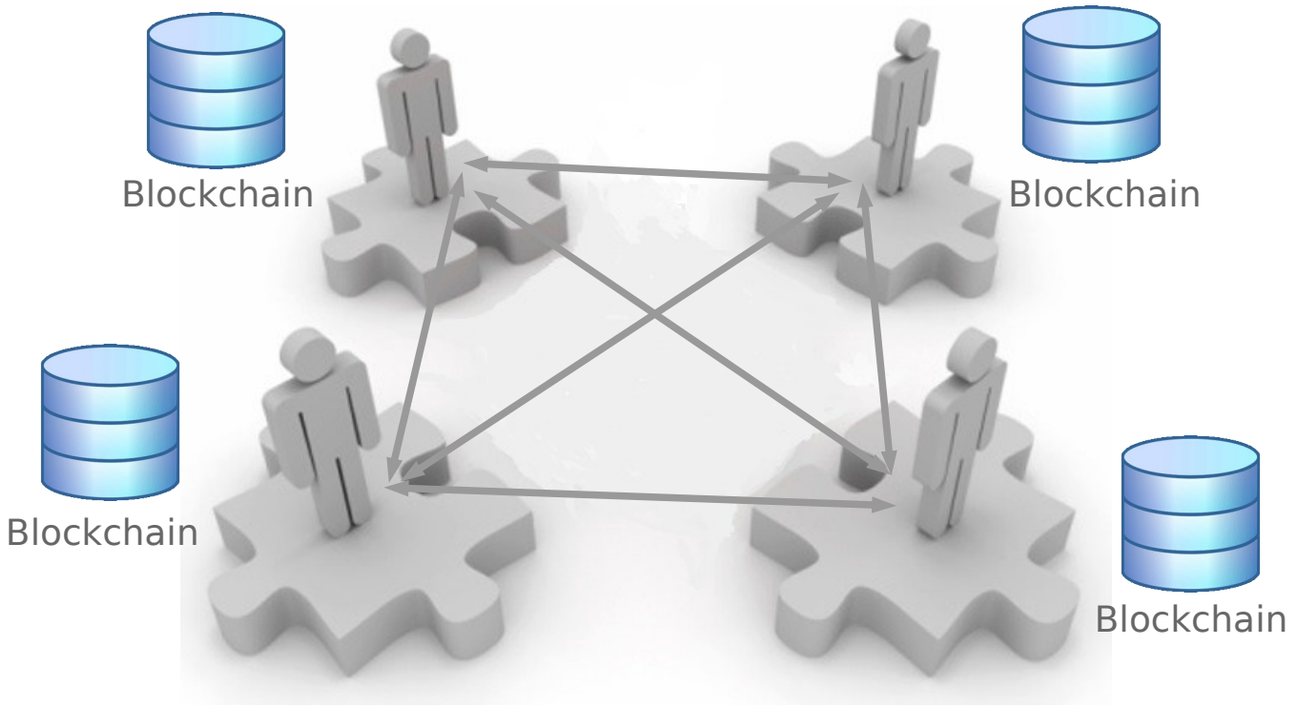
- Payments **before** Bitcoin and its blockchain

Financial DB with all the transactions

Bank, Western Union, Paypal, Money Gram,...

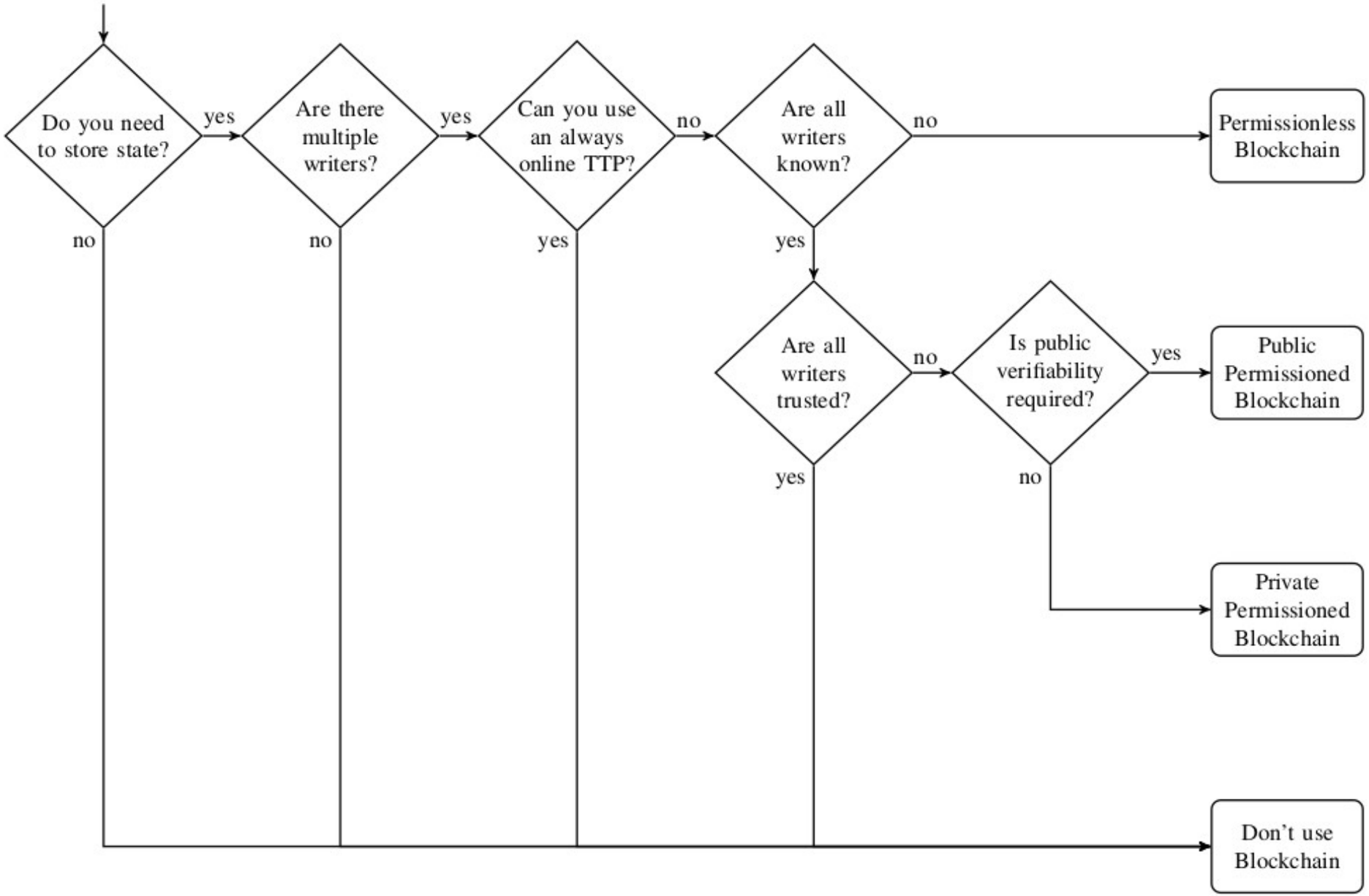
Do you need a blockchain?

- Payments **after** Bitcoin and its blockchain



Do you need a blockchain?

- **Payments** without a central authority are the **first killer application** of the blockchain
- But after that?
  - New proposals appeared and many start-ups were created around the blockchain ecosystem
  - But in many cases a (distributed) database could be enough



Suggested reading: https://ieeexplore.ieee.org/document/8525392