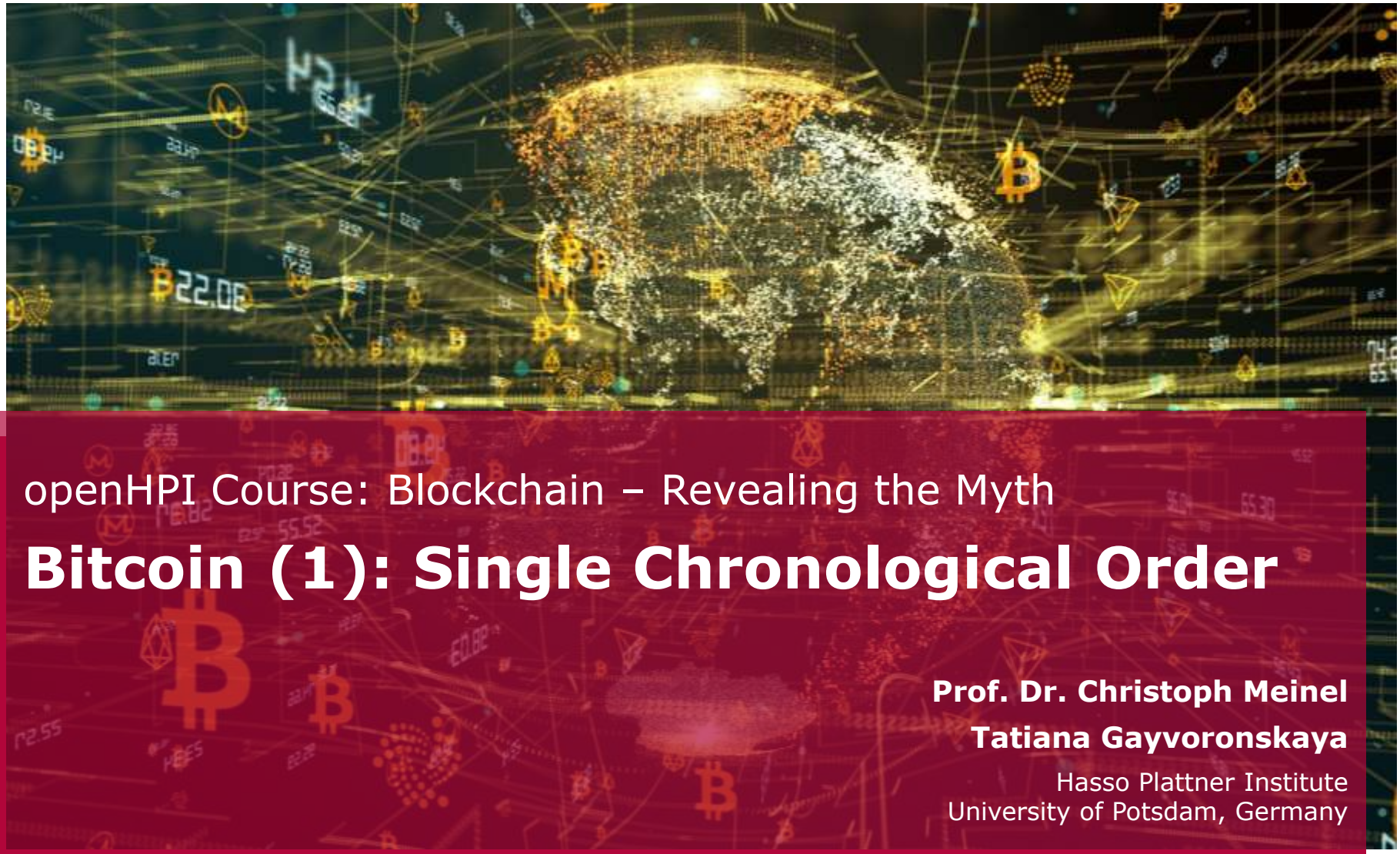


openHPI Course: Blockchain – Revealing the Myth

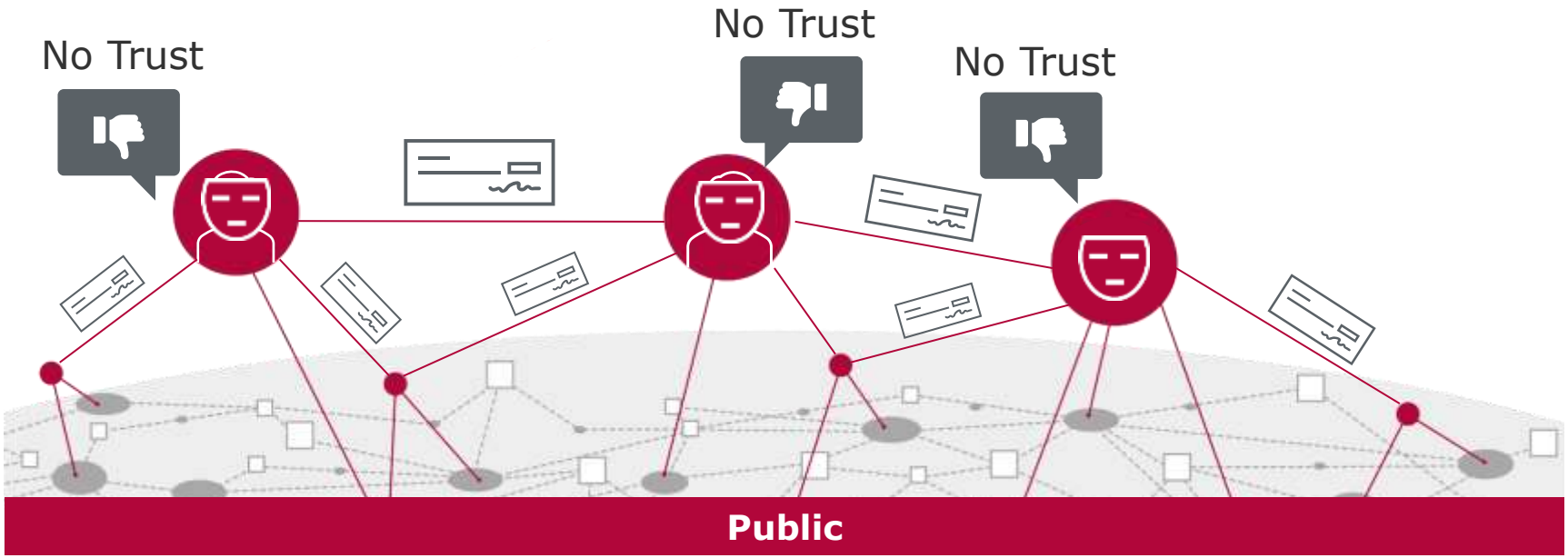
Bitcoin (1): Single Chronological Order

Prof. Dr. Christoph Meinel
Tatiana Gayvoronskaya
Hasso Plattner Institute
University of Potsdam, Germany

A System for Electronic Transactions Without Relying on Trust (1/2)

Starting Point: **No central authority** and **no trusted third party**

- To prevent, that the malicious receiver of a money transfer claims not to have received anything
  - we do not trust other users
  - we make all transactions **public** and us **anonymous**



A System for Electronic Transactions Without Relying on Trust (2/2)

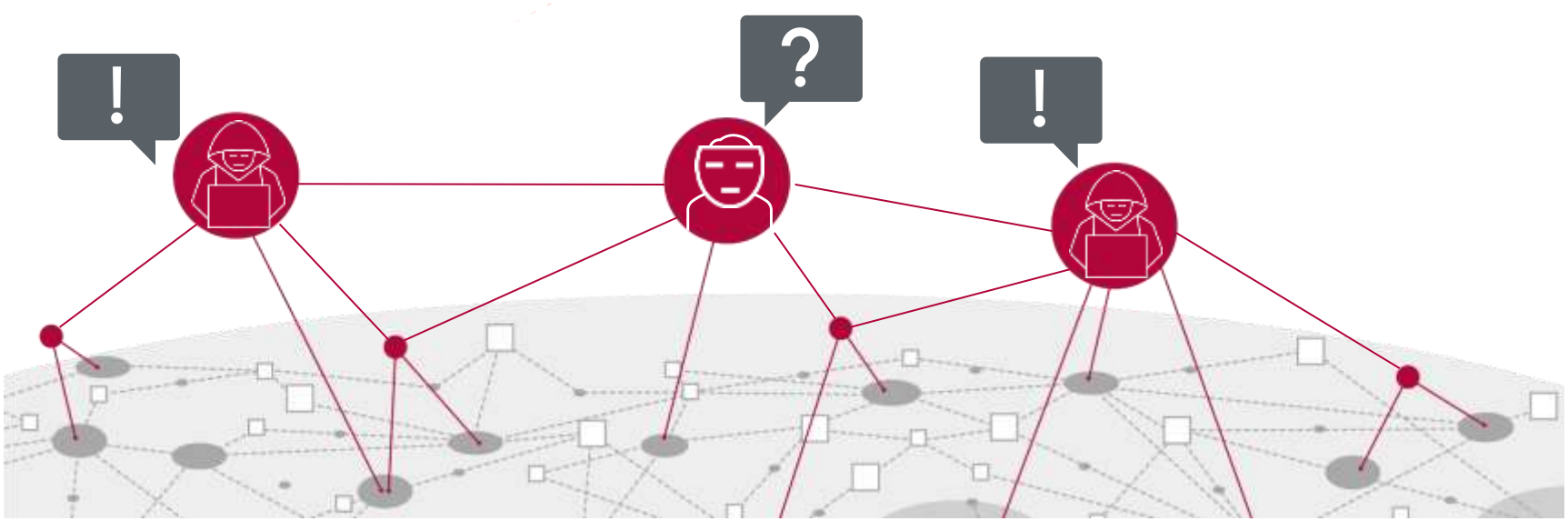
Question:

- How do we organize ourselves, our network, without a central authority or a trusted third party?
- How do we protect ourselves from fraud?
- …

- To accomplish this without a trusted third party, we need a system for participants to **agree on a single history of the order** in which transactions were received
- The payee needs a **proof** that at the time of the transaction, the **majority of nodes agreed** that the money hasn't been spent already

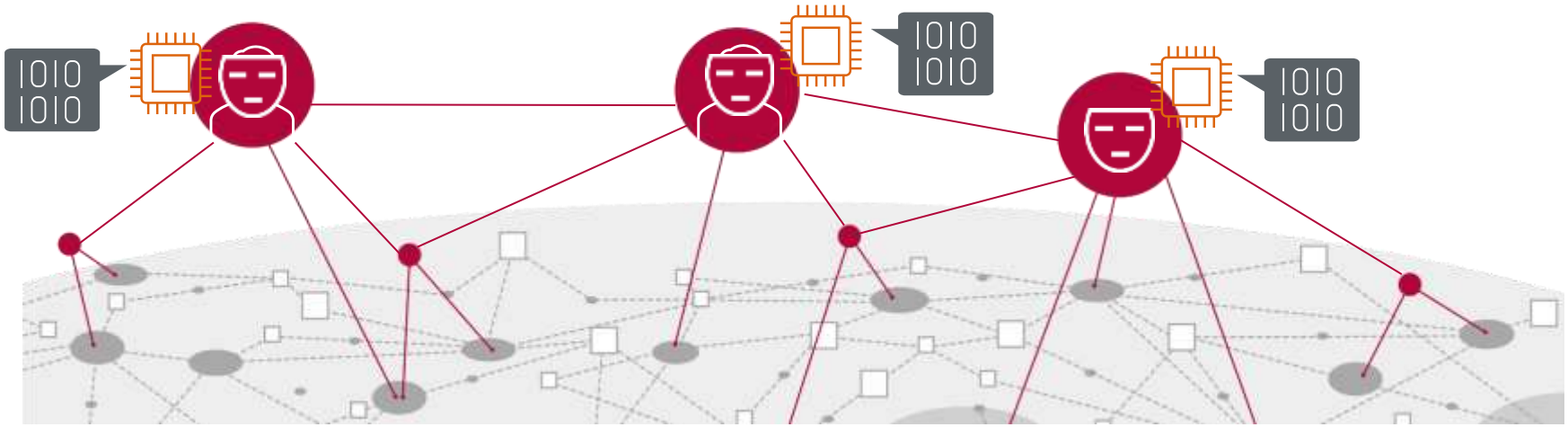
Single Chronological Order
Majority Decision Making (1/2)

- **Everyone** in our network **can vote** for the **proper order of transactions**
- Therefore, the majority decision is represented by the **majority of votes**
- Imagine if the majority would base on **one-IP-address-one-vote**, it could be **subverted** by anyone able to **allocate many IPs**



Single Chronological Order
Majority Decision Making (2/2)

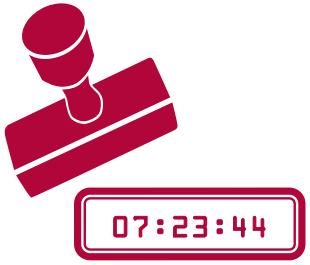
- The idea of the Bitcoin founder was similar to that in Adam Back's Hashcash, to cast the **vote done in form of a proof of work**
- **Work** consists of a complex **computational task** demanding a huge **computational CPU effort** to be solved
- **Proof-of-work** is essentially **one-CPU–one-vote**
- **System is secure** as long as **honest nodes collectively control more CPU power** than any cooperating group of attacker nodes



Single Chronological Order Timestamping (1/2)

We **announce** all our transactions publicly by **broadcasting** them to all users and we **vote** with our **CPU effort** for a **proper order** of transactions

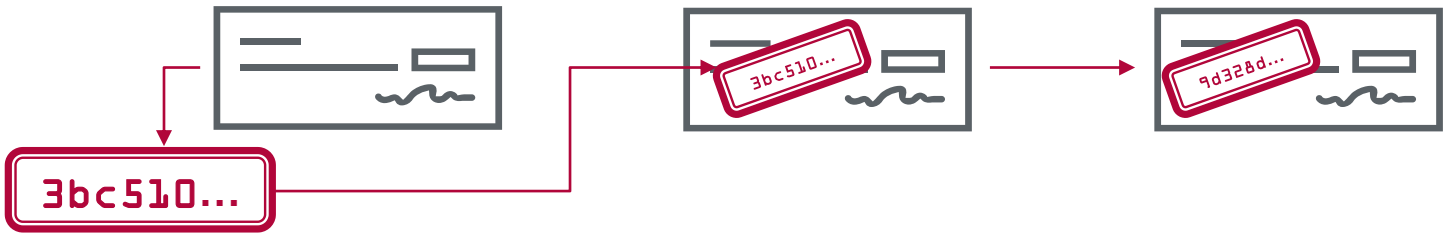
- How exactly does this **order** look like, i.e. how does the **timestamping** work?
- Unlike physical currency, a **"digital banknote"** is a digital file that can be **duplicated or falsified**. Thus, the same single "digital banknote" can be easier spent more than once **(double-spending)**
- How do we protect from double–spending, that the history of the order was **not changed afterwards**?



Single Chronological Order Timestamping (2/2)

Timestamping and the **proof** that the **order and contents** of the transactions have **not been manipulated afterwards** can be provided by means of **timestamps linked by hashes**

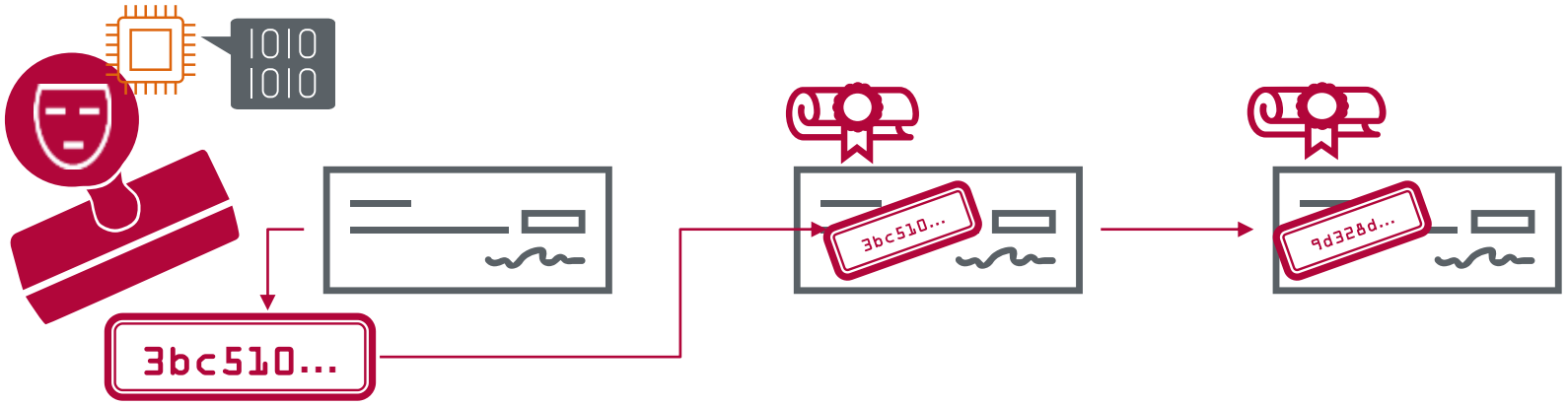
- Each transaction is **timestamped before its hash is computed** and **widely published**
- **Each timestamp** includes the **timestamp of the previous transaction in its hash**, forming a **chain**, with each additional timestamp reinforcing the ones before it
- Link to the previous timestamp **proves** that the previous transaction must have **existed** already otherwise it could not get into the hash of the transaction



Single Chronological Order
Chain of Hashes

Each user **broadcasts** its new transaction

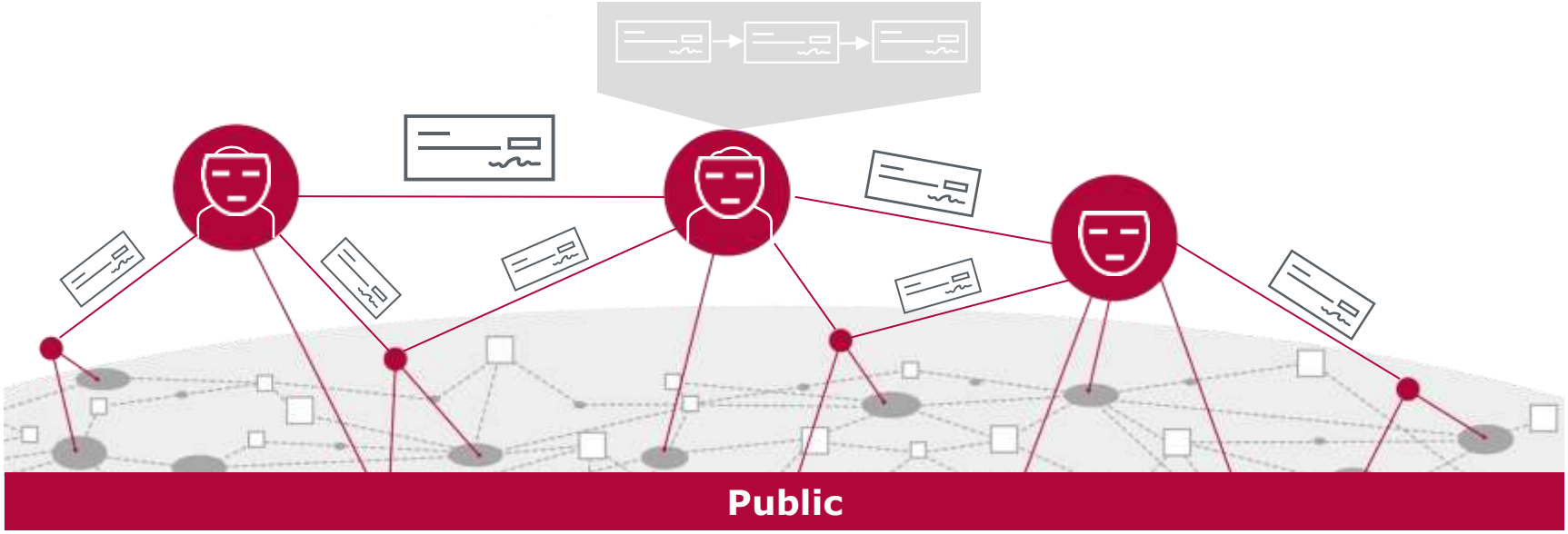
- Users accept the transaction only if it is **valid** and not already spent
- Users express their **acceptance** of the transaction **by working** on creating the next hash in the chain, using the hash of the accepted transaction as the previous hash, and **works on finding a difficult proof-of-work** for it
- This **proof** together with the **hash** of the previous transaction are added to the new transaction



Single Chronological Order Review (1/3)

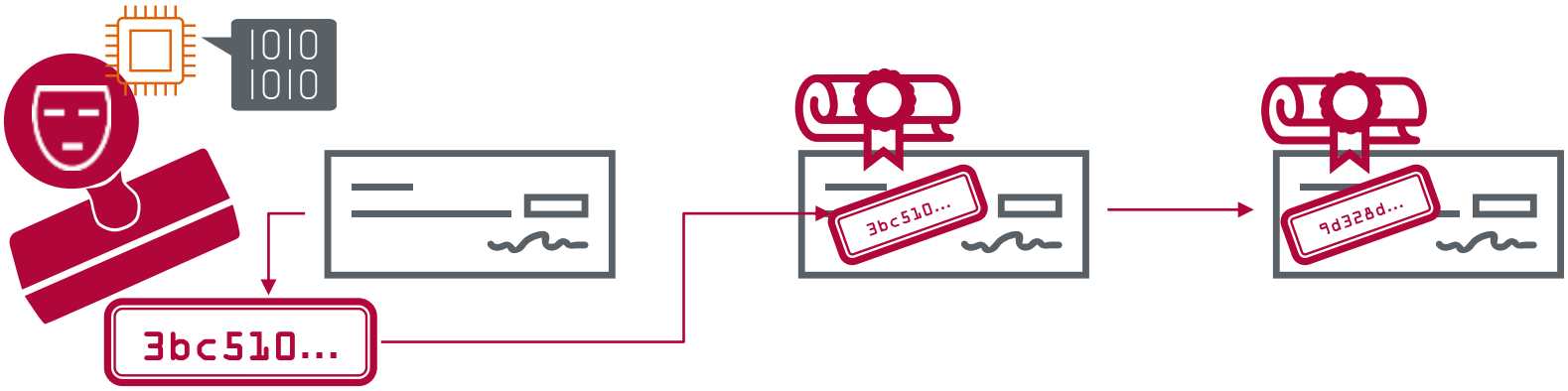
Let us summarize the individual steps, how the Bitcoin network is organized and protected

- Each user **broadcasts** its new transaction
- Users accept the transaction only if it is **valid** and not already spent
- Users express their **acceptance** of the transaction **by working** on creating the next hash in the chain



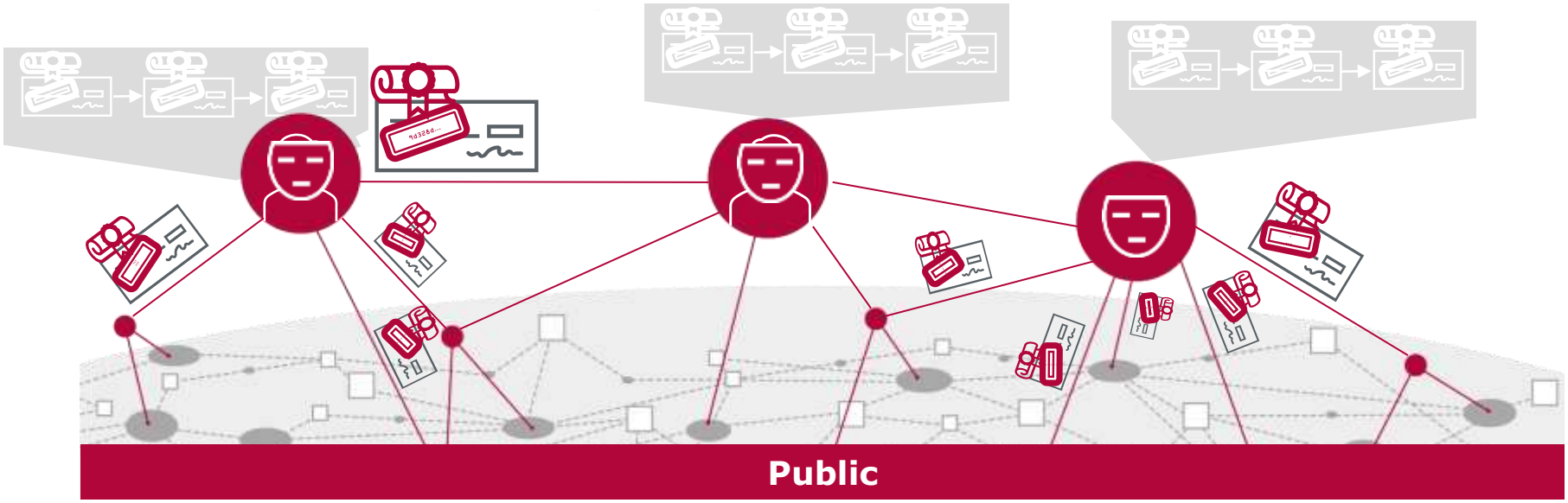
Single Chronological Order Review (2/3)

- Users use the hash of an accepted transaction as the **previous hash** for the **next** arriving transaction
- To underpin the acceptance, they work on finding a solution of a difficult **computational task** (proof-of-work)
- This **proof** together with the **hash** of the previous transaction are added to the new transaction



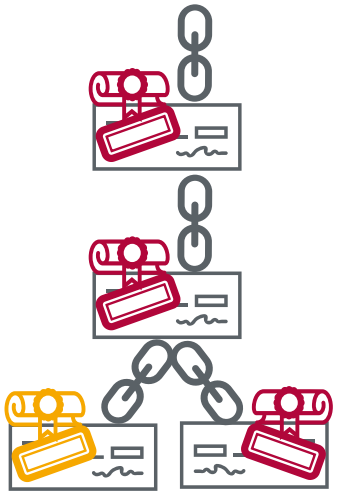
Single Chronological Order Review (3/3)

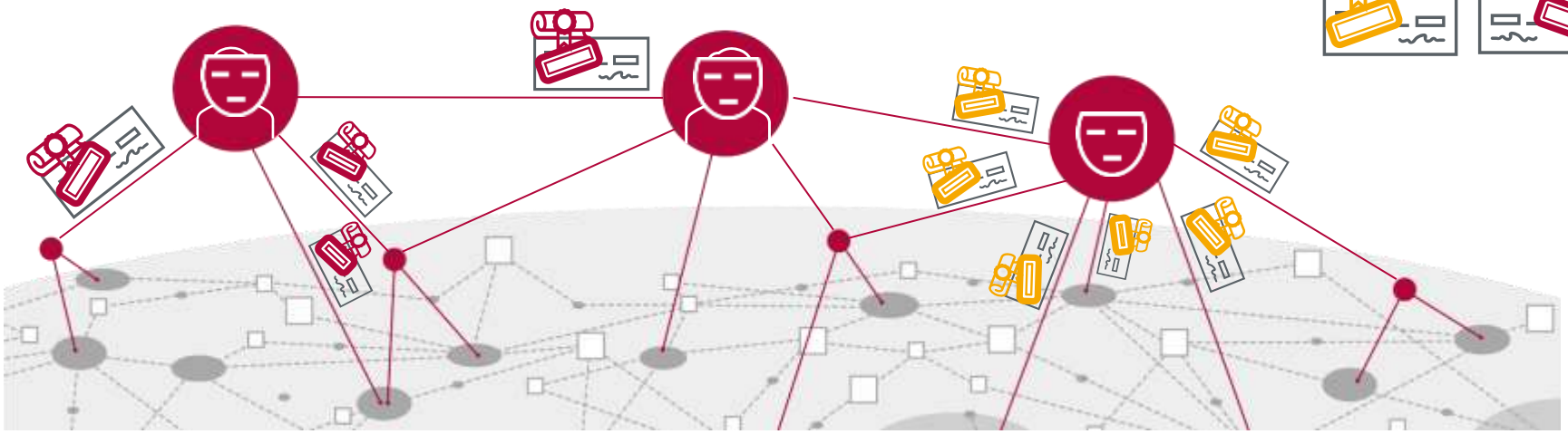
- Then the new transaction is **broadcasted** again
- Users who have **not yet solved the task** verify the newly arrived transaction with the reference and proof and add it to their **existing chain**
- Then they continue working on **further chaining**
- In this way, every user has an **equal copy** of the ordered transactions



Single Chronological Order
Fork – Chain Branches (1/2)

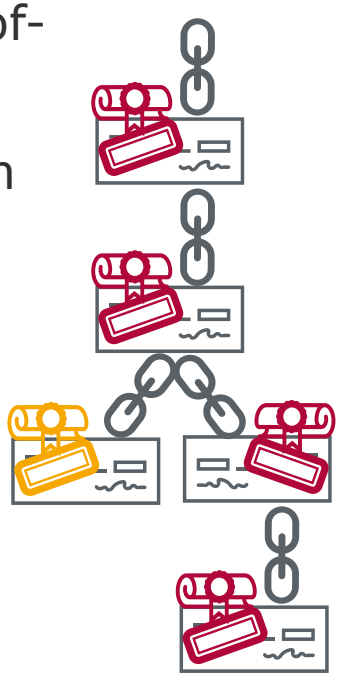
- Since we have a peer-to-peer network without a central authority, it can happen that **several users simultaneously** find a solution for the proof-of-work and each broadcasts a new referenced and proofed transaction
- Some users may receive **one or the other first**
- If these transactions comply with all rules and refer to the same last transaction, it is possible that the **chain will branch "fork"**





Single Chronological Order
Fork – Chain Branches (2/2)

- In that case, users work on the **first transaction** they received, but save the **other branch** in case it becomes longer
- The fork will be broken when the next solution for proof-of-work is found and **one branch becomes longer**
- The users that were working on the other branch will then **switch to the longer one**
- Thus, after a time, only a **single chain** will prevail
- The shortest chain is ignored; the transactions therein do not expire
- What is not yet contained in the valid blocks, will be **saved in the buffer** (**memory pool**) of the user and used for further chaining



Majority Decision is Represented by the Longest Chain

Once the **CPU effort** has been expended to make it satisfy the proof-of-work, the transaction **cannot be changed without redoing the work**

- To **modify** a past transaction, an attacker would have to **redo the proof-of-work** of this and all the following transactions and then catch up with and surpass the work of the honest nodes
- **Imagine**, we have 11 transactions in our chain, to modify the 8^{th} transaction we must redo the proof-of-work of this transaction and all the following 9^{th}-11^{th} transactions

The **longest chain** serves as **proof of the chronological order of transactions**, since it came from the **largest pool of CPU power** (most votes)

Majority Decision is Represented by the Longest Chain

Summary

- **Majority decision** is represented by the **longest chain** in which the **greatest proof-of-work effort** was invested
- If a **majority of CPU power** is controlled by **honest nodes**, the honest chain will grow the fastest and outpace any competing chains