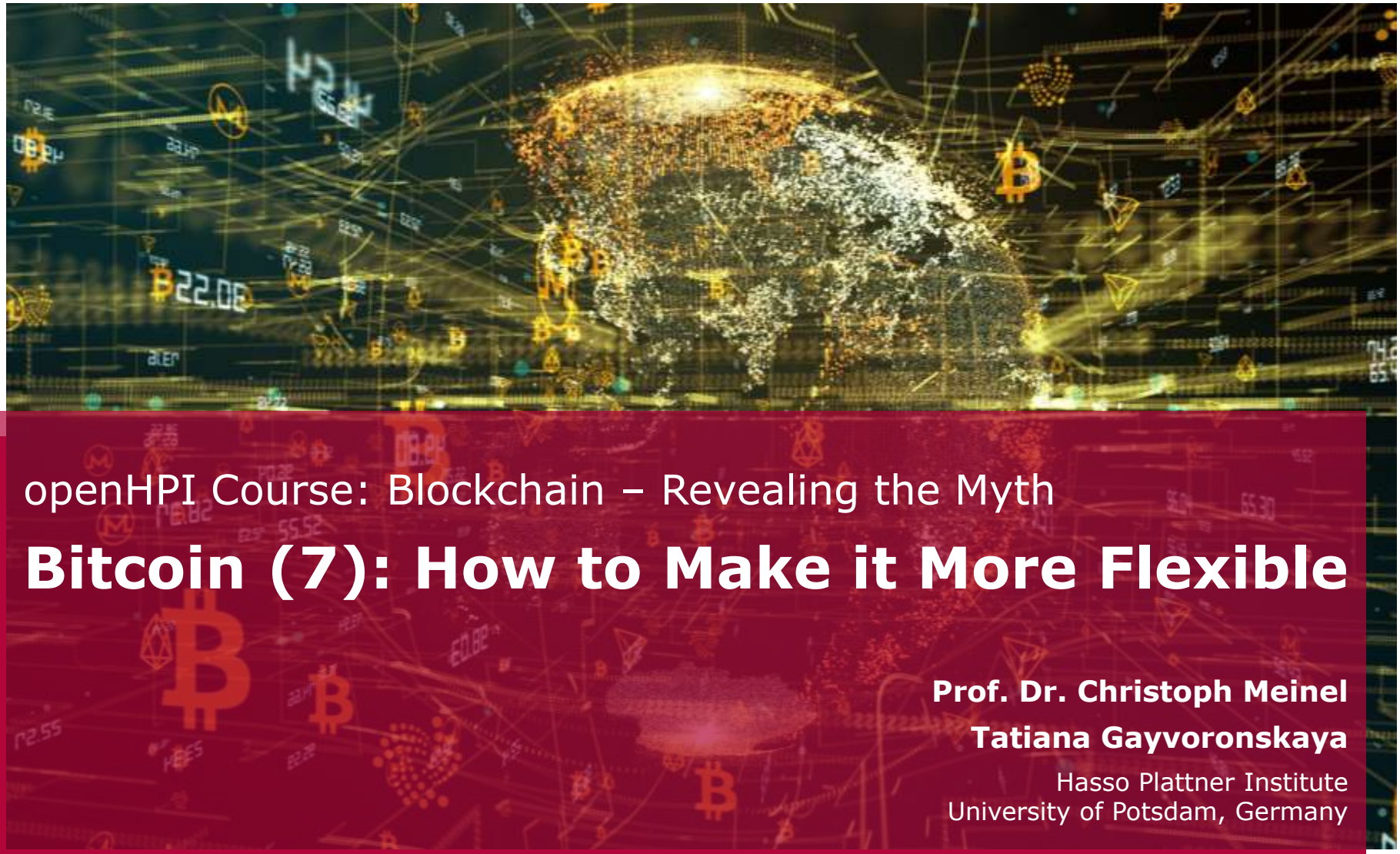


openHPI Course: Blockchain – Revealing the Myth

Bitcoin (7): How to Make it More Flexible

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

Bitcoin: How to Make it More Flexible?

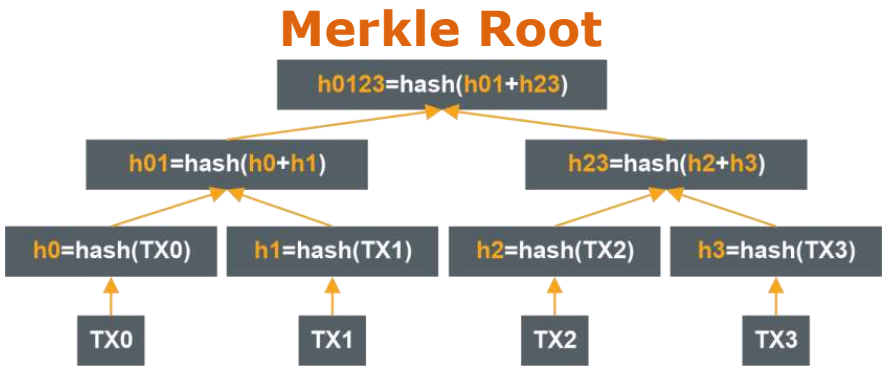
In this **final part** of our discovery of the **Bitcoin system**, we consider the **final piece** of the Bitcoin solution, namely, to make the system **more flexible**

Review:

- We have a **framework of coins** made by digital signatures
- We have a system for participants to **agree on a single history of the order** of transactions
- We have a solution to the **double-spending problem** using a peer-to-peer distributed **timestamping** to generate **computational proof of the chronological order** of transactions
- We provided an **incentive** for users **to comply with the rules** and to **provide a proof-of-work**
- Users can **leave** and **rejoin** the network **at will**

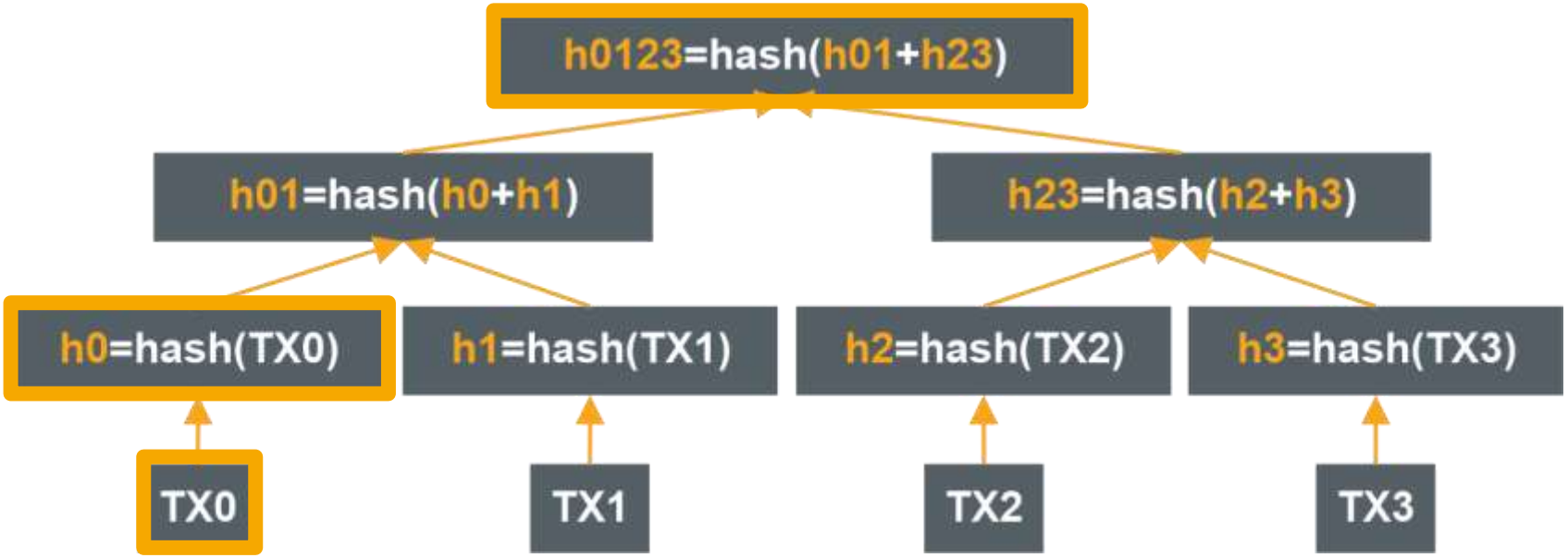
Bitcoin: How to Make it More Flexible? Reclaiming Disk Space

- Once the latest transaction in a coin (**UTXO**) is buried under enough blocks, the **spent transactions before it** can be **discarded** to save disk space
- To facilitate this **without breaking the block's hash**, transactions are hashed in a **Merkle Tree**
- A hash tree ("Merkle Tree") is a tree-like structure (from graph theory) that consists of successive hash values. The **Merkle Root** is the **last hash** value in this hash tree



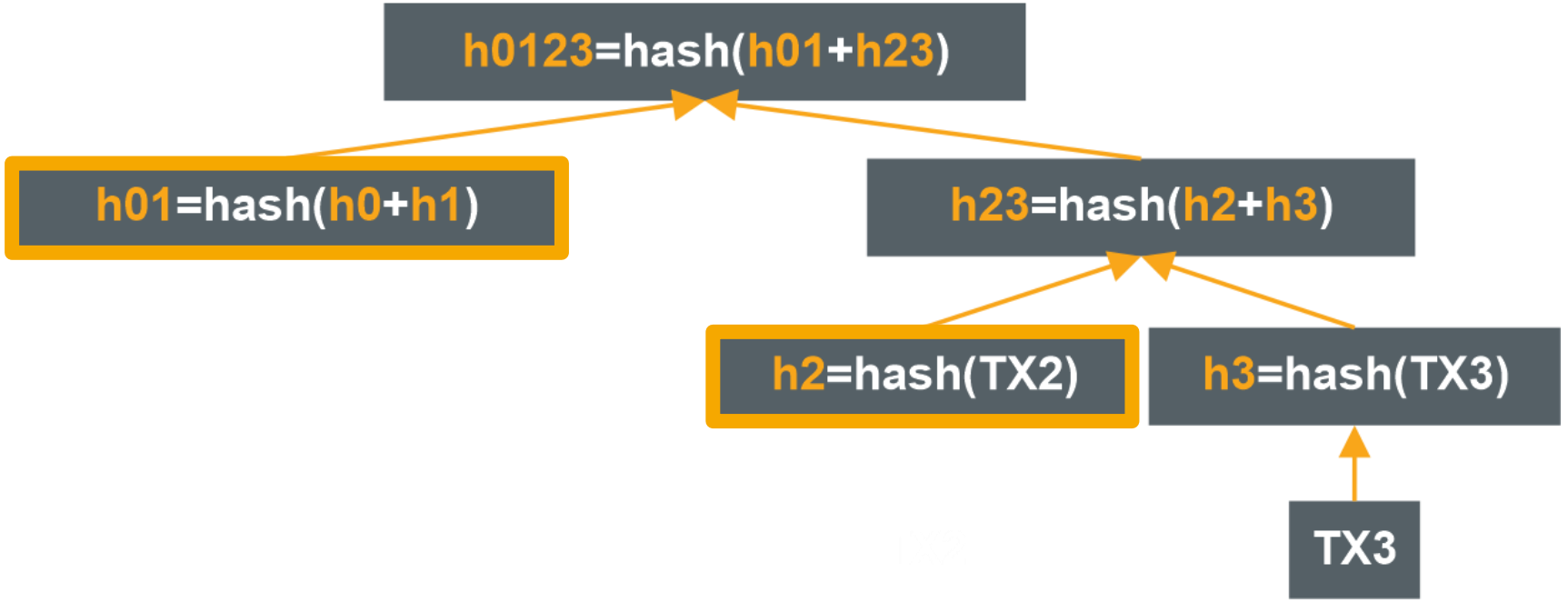
Bitcoin: How to Make it More Flexible? Merkle Tree

- In the shown example we see that a hash value **h0** from transaction 0 (**TX0**) is first created. The same is done with the transactions **TX1**, **TX2** and **TX3**
- Subsequently, **further hash values** are calculated from the first found hash values of the original transactions. In this case, the **root of the tree h0123** is the Merkle Root



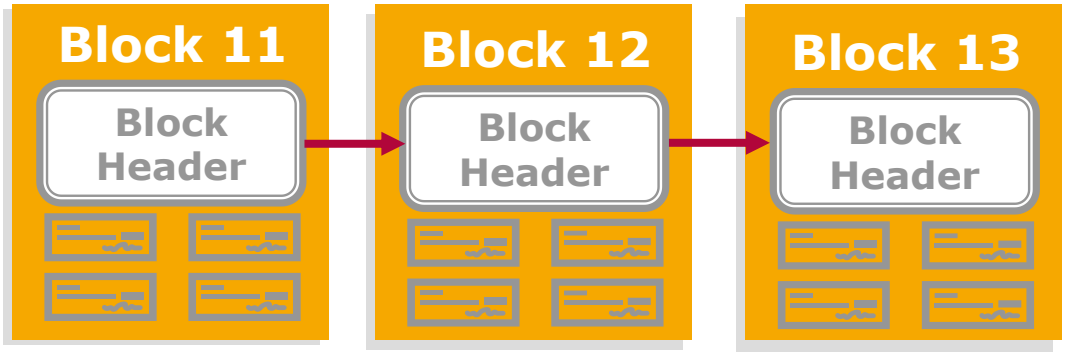
Bitcoin: How to Make it More Flexible? Stubbing off Branches of the Tree

- **Old blocks** can then be **compacted** by **stubbing off branches** of the tree. The **interior hashes** do **not need to be stored**
- In order to check whether **TX3** was included in the respective block, the **h2** and **h01** are sufficient for verification (calculating the Merkle Root and comparing it with the root stored in the block header)



Bitcoin: How to Make it More Flexible? Block Header

- This means that we no longer have to take the **hash of the entire block** (with all transactions, hash of the previous block and nonce) as a reference for the following blocks
- Instead, we **separate** the information data from the transactions in the so-called **block header**
- With other words, the **Merkle Root** together with a **hash of the previous block**, **nonce** and **some other** additional information are stored in the **block header** and **its hash** is used as a reference to the block in the next block header
- The information in the block header provides a **unique summary** of the entire block



Bitcoin: How to Make it More Flexible? Simplified Payment Verification (1/3)

- Basically, in our system **all nodes** are **"created equal"** and can be both **service users** and **service providers**
- If we look at the **size** of the bitcoin blockchain (February 2021: above **320 GB**), it is clear that not every user has **enough resources** for storage and verification
- Thus, in a bitcoin system there are two types of users: normal user (as described so far) also called **full node** and light users also called **lightweight nodes**

Full Node

Lightweight Node

Bitcoin: How to Make it More Flexible? Simplified Payment Verification (2/3)

- **Full nodes** save the **entire blockchain** and are involved in the **verification process** (blocks and transactions)
- **Lightweight nodes** only save a **copy of the block headers** of the longest proof-of-work chain, which they can get by querying full nodes until they are convinced, they have the longest chain
- Lightweight nodes **can't check the transaction for themself**, but by linking it to a place in the chain, they can see that a **full node has accepted it**, and blocks added after it further confirm the **network has accepted it**

Full Node

Lightweight Node

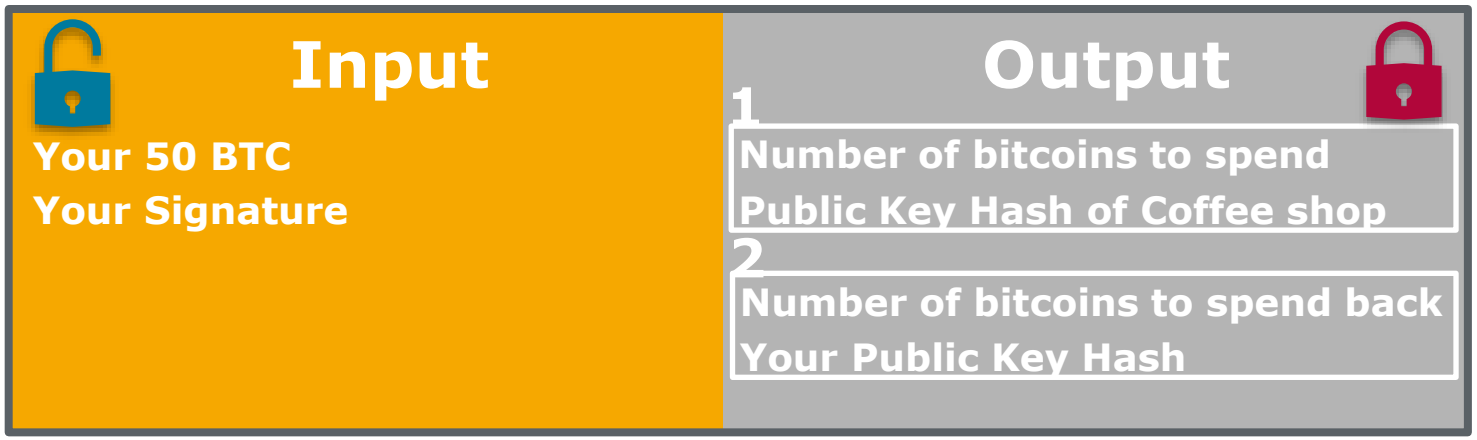
Bitcoin: How to Make it More Flexible? Simplified Payment Verification (3/3)

- The **full nodes** are the **backbone** of the Bitcoin system
- They allow the system to **grow** and at the same time **remain secure** and **decentralized**
- **Lightweight** nodes do not have any **block contents** (transactions), **they have to trust the full nodes** that the blocks and transactions are created in accordance with the rules and do not contain any doubled spending

Bitcoin: How to Make it More Flexible? Scripts

The last tool that we need to present here, and that is important for the presentation of further material, are **scripts**

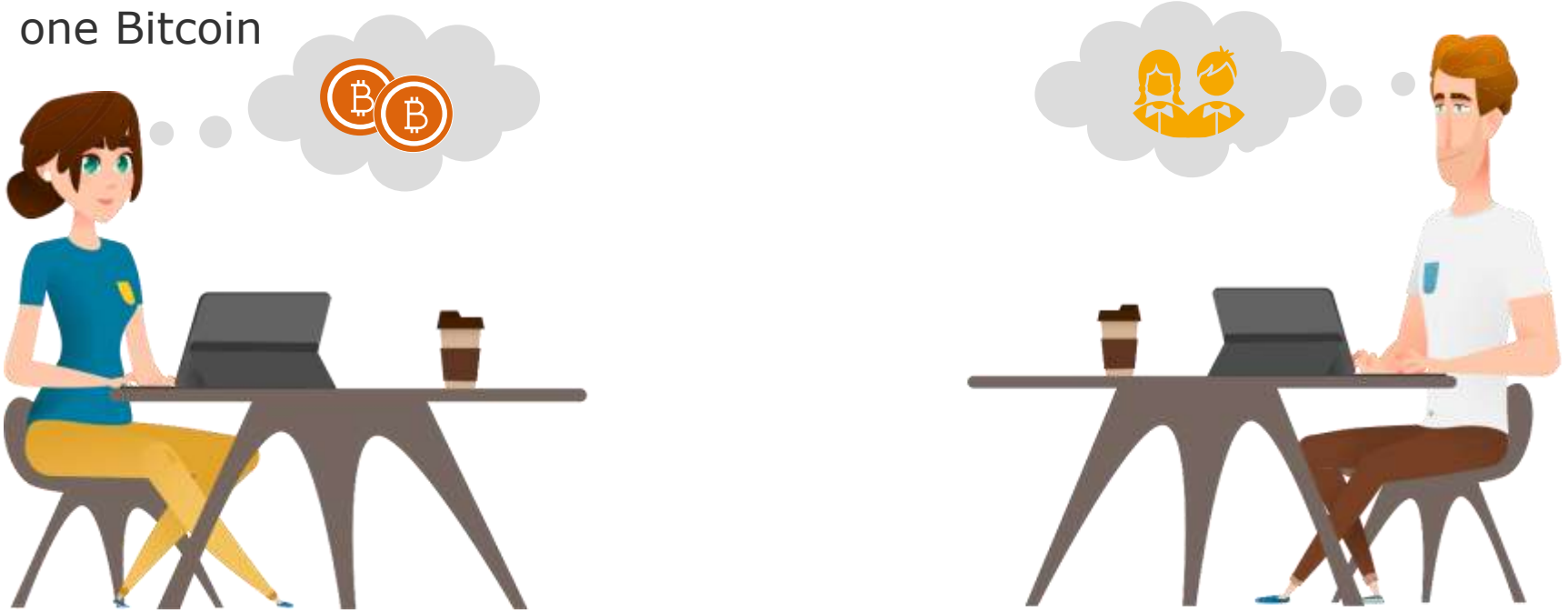
- Scripts are like "**mini programs**" used to **lock outputs (ScriptPubKey)** and **unlock inputs (ScriptSig)**
- In place of the hash of the recipient's public key in the output comes the **hash from the locking script**
- And in place of your signature in the input comes the **unlocking script with the necessary data** (signatures, public keys)



Bitcoin: How to Make it More Flexible?
Scripts – Example (1/4)

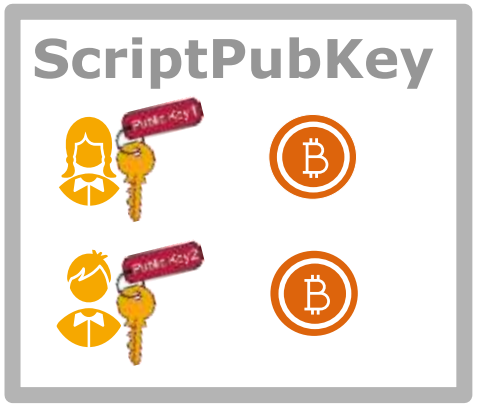
Scripts offer us **more flexibility** in **how** (under which conditions) the respective bitcoins **may be spent**. To visualize this, let's consider an **example**

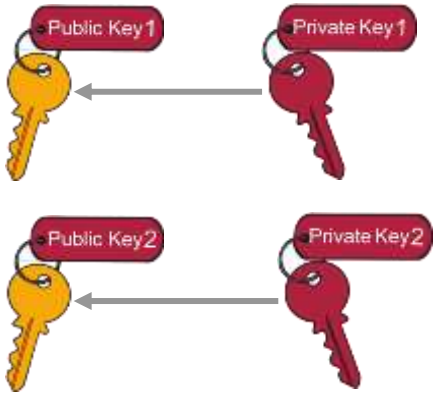
- Let’s imagine that **Alice** wants to “transfer” two bitcoins to **Bob**
- Bob plans to give these bitcoins to his **children**, each receiving one Bitcoin



Bitcoin: How to Make it More Flexible? Scripts – Example (2/4)

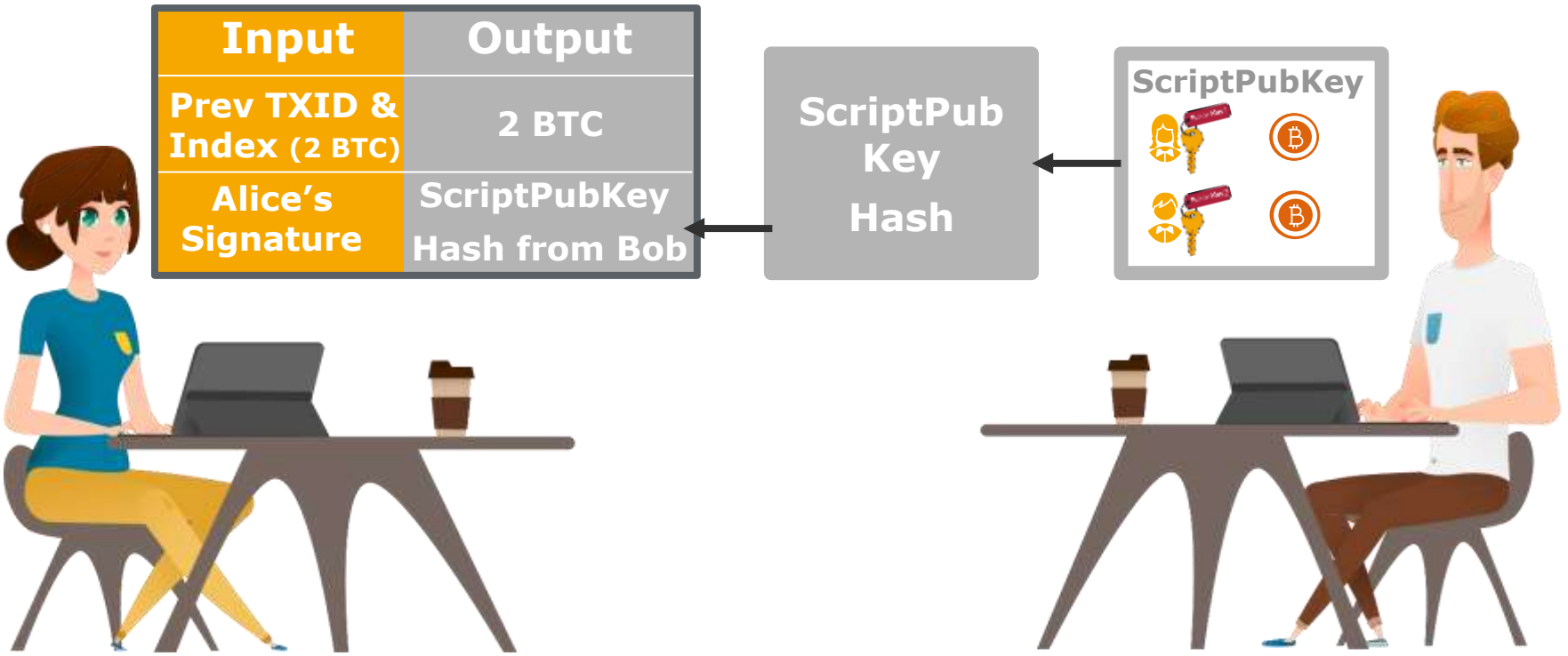
- Bob creates **two private keys** and generates a **public key for each**
- Then he creates a **script** which states that his **daughter Bea** (first public key) will be allowed to use **half of the bitcoins**
- It also states that his **son Bill** (second public key) gets the **second half**





Bitcoin: How to Make it More Flexible? Scripts – Example (3/4)

- Finally, Bob takes the **script with the public keys** and **creates a hash value**
- This information appears in the **ScriptPubKey** in the **output of Alice's** transaction



Bitcoin: How to Make it More Flexible? Scripts – Example (4/4)

- **Bea** and **Bill** can thus use their **private keys** to **"spend" their bitcoins**
- To do this, each of them has to create a **transaction whose ScriptSig** contains the following information: signatures (generated by means of their private keys) and the script with the public key used there.

Input	Output
Prev TXID	1 BTC
ScriptSig	Public Key Hash x

Input	Output
Prev TXID	1 BTC
ScriptSig	Public Key Hash x

Bitcoin: How to Make it More Flexible? Summary

- Once the **UTXO** is buried under enough blocks, the **spent transactions before it** can be **discarded** to save disk space
- To facilitate this **without breaking the block's hash**, transactions are hashed in a **Merkle Tree**
- **Merkle Root** together with a **hash of the previous block**, **nonce** and **some other** additional information are stored in the **block header** and **its hash** is used as a reference to the block in the next block header
- **Lightweight nodes** only save a **copy of the block headers** of the longest proof-of-work chain, which they can get by querying full nodes until they are convinced, they have the longest chain
- **Scripts** offer **more flexibility** in **how** (under which conditions) the respective bitcoins **may be spent**

Recommended Literature and References

Recommended literature:

- For more **technical information** on the subject of bitcoin and **examples of coding**, we would recommend the following exciting and very easy-to-understand explanation By Greg Walker https://learnmeabitcoin.com/technical/
- For more detailed information about bitcoin **in the context of the big picture of blockchain**, we would recommend our book "Blockchain: Hype or Innovation"

References:

- S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, (2008)
- Learn me a bitcoin by Greg Walker, https://learnmeabitcoin.com/technical/