Introduction to public-key cryptography

Part 3: the Bitcoin protocol

Jean-Sébastien Coron

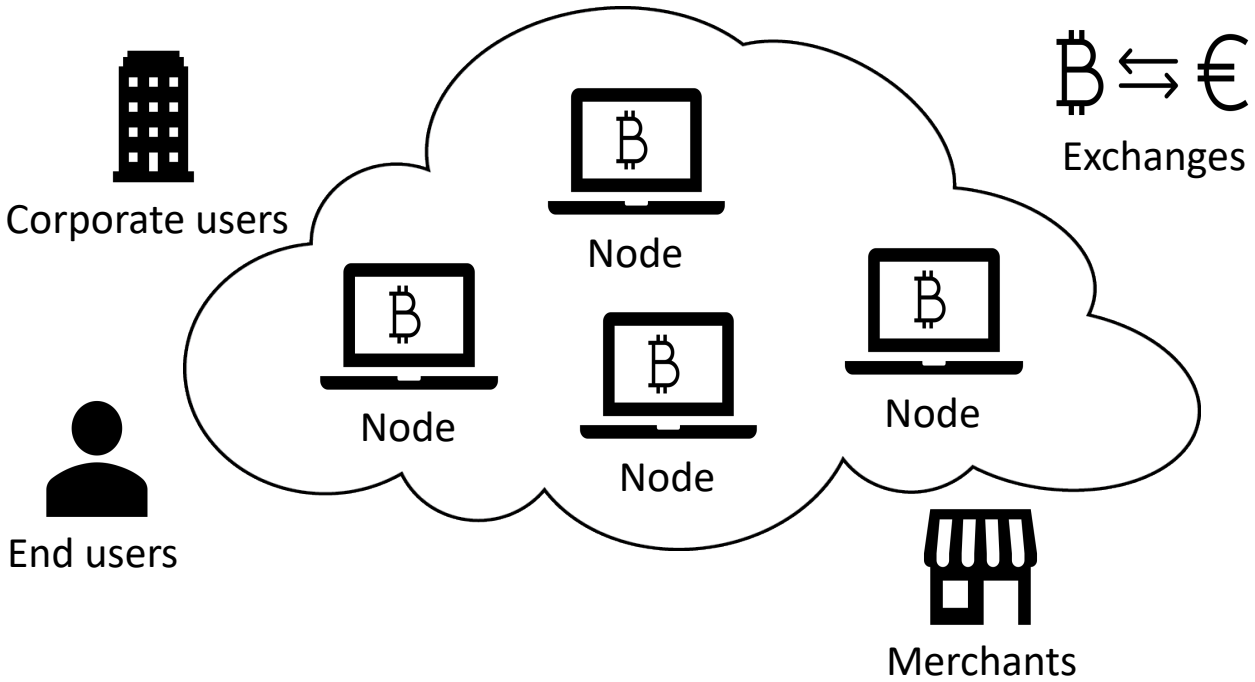
University of Luxembourg

Outline

- Part 1: basic constructions
  - History
  - Classical cryptography: block-ciphers, hash functions
  - Public-key cryptography: RSA encryption and RSA signatures, DH key exchange
- Part 2: applications of public-key cryptography
  - Security models
  - How to encrypt and sign securely with RSA. OAEP and PSS.
  - Public-key infrastructure. Certificates, HTTPS protocol.
  - Bitcoin and the cryptographic blockchain. (this lecture)

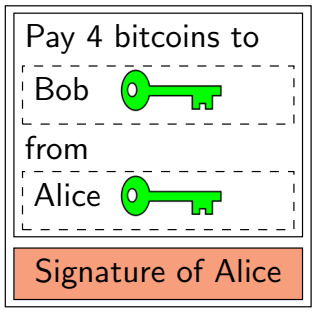
Bitcoin

- Decentralized payment system, invented by Satoshi Nakamoto in 2008.
  - "Bitcoin: a peer-to-peer electronic cash system"
- Network of thousands of computers.
  - No trusted authority. Pseudonymous.
  - Permissionless: anybody can run a Bitcoin node.



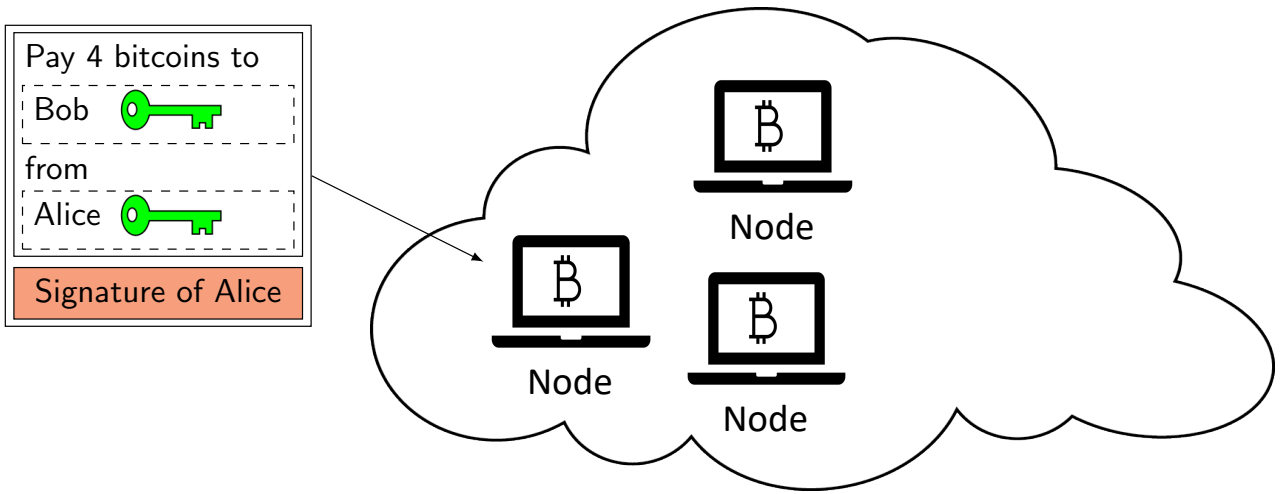
Payment of Bitcoin (simplified)

- Assume that Alice wants to transfer 4 bitcoins to Bob
  - Alice prepares the transaction
    - The amount to transfer (4 bitcoins), the address of the recipient, her digital signature
  - Alice sends the transaction to the network
    - The network verifies the transaction: Alice has 4 bitcoins, signature is valid.
  - Transaction is included in the blockchain
    - Alice's transaction is appended to the transaction history, creating a new *block*.



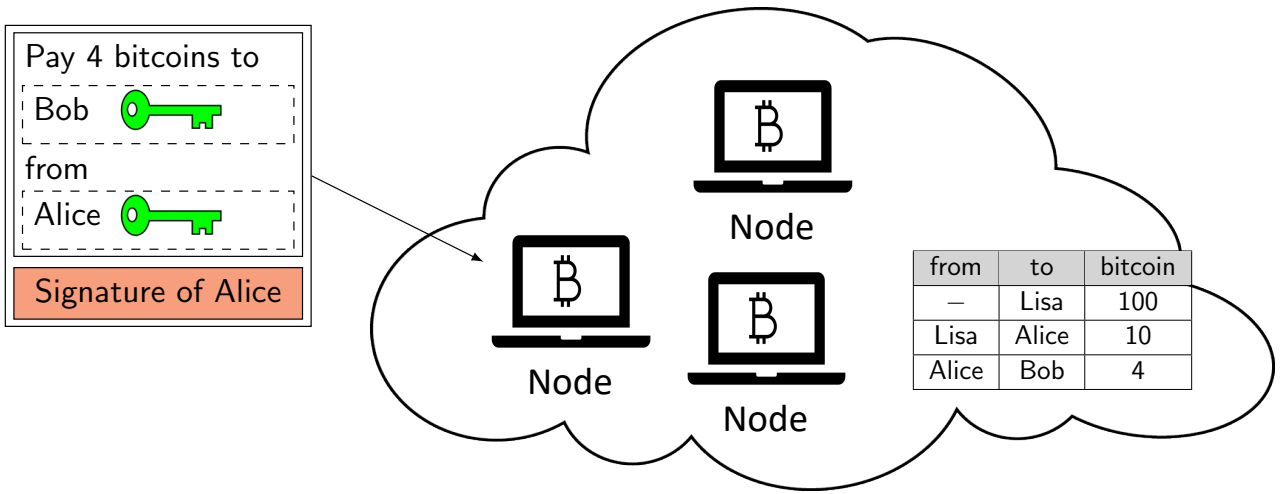
Payment of Bitcoin (simplified)

- Assume that Alice wants to transfer 4 bitcoins to Bob
  - Alice prepares the transaction
    - The amount to transfer (4 bitcoins), the address of the recipient, her digital signature
  - Alice sends the transaction to the network
    - The network verifies the transaction: Alice has 4 bitcoins, signature is valid.
  - Transaction is included in the blockchain
    - Alice's transaction is appended to the transaction history, creating a new *block*.



Payment of Bitcoin (simplified)

- Assume that Alice wants to transfer 4 bitcoins to Bob
  - Alice prepares the transaction
    - The amount to transfer (4 bitcoins), the address of the recipient, her digital signature
  - Alice sends the transaction to the network
    - The network verifies the transaction: Alice has 4 bitcoins, signature is valid.
  - Transaction is included in the blockchain
    - Alice's transaction is appended to the transaction history, creating a new *block*.



from	to	bitcoin
–	Lisa	100
Lisa	Alice	10
Alice	Bob	4

The Bitcoin ledger: simplified model

from	to	bitcoin
–	Lisa	100
Lisa	Alice	10
Alice	Bob	4

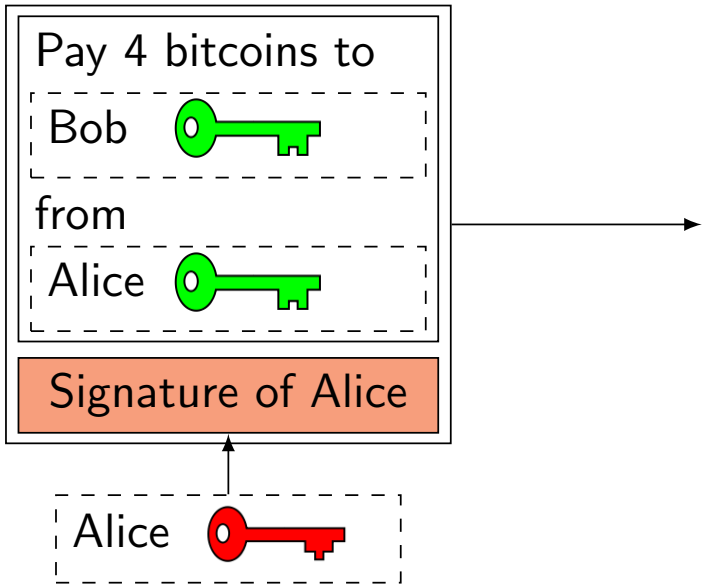
- New transactions are appended at the end of the spreadsheet
- Balance of Alice: sum of "to" minus sum of "from"
  - balance of Alice: 6 bitcoins
  - Alice can transfer 1 bitcoin to Charlie
  - new balance is 5 bitcoins

The Bitcoin ledger: simplified model

from	to	bitcoin
–	Lisa	100
Lisa	Alice	10
Alice	Bob	4
Alice	Charlie	1

- New transactions are appended at the end of the spreadsheet
- Balance of Alice: sum of "to" minus sum of "from"
  - balance of Alice: 6 bitcoins
  - Alice can transfer 1 bitcoin to Charlie
  - new balance is 5 bitcoins

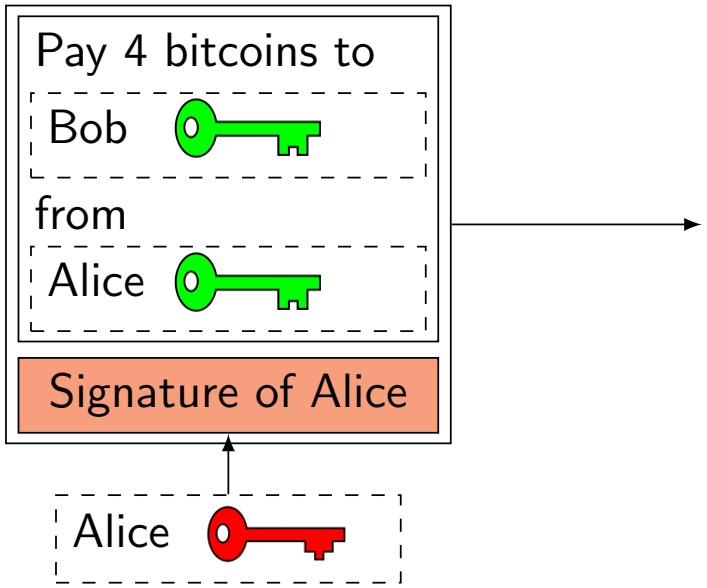
Certifying transactions with signatures



from	to	bitcoin
–	Lisa	100
Lisa	Alice	10
Alice	Bob	4

- Alice signs the transaction to pay 4 bitcoins to Bob
  - Bob is identified by his public-key
  - Anybody can verify the transaction using the public-key of Alice.
- If Alice looses her private-key, she cannot spend her bitcoins.

Certifying transactions with signatures



from	to	bitcoin
–	Lisa	100
Lisa	Alice	10
Alice	Bob	4

- Alice signs the transaction to pay 4 bitcoins to Bob
  - Bob is identified by his public-key
  - Anybody can verify the transaction using the public-key of Alice.
- If Alice looses her private-key, she cannot spend her bitcoins.

Pseudonymous payment

- Names are replaced by public-keys
  - Payments are made via public-keys instead of names.

from	to	bitcoin
–	Lisa	100
Lisa	Alice	10
Alice	Bob	4
Alice	Charlie	1

- One replaces the public-keys by their hash to save space: the *public-key hash* (PKH).
- One can use a fresh address for each payment to improve privacy.

Pseudonymous payment

- Names are replaced by public-keys
  - Payments are made via public-keys instead of names.

from	to	bitcoin
–	`45ab...1c32`	100
`45ab...1c32`	`8c24...fe5a`	10
`8c24...fe5a`	`7a53...b3ac`	4
`8c24...fe5a`	`3c6e...02a3`	1

- One replaces the public-keys by their hash to save space: the *public-key hash* (PKH).
- One can use a fresh address for each payment to improve privacy.

Preparing a Bitcoin transaction

Confirmed transactions

txid	1	sig	pk	5	PKH_1
txid	0	sig	pk		

txid	2	sig	pk

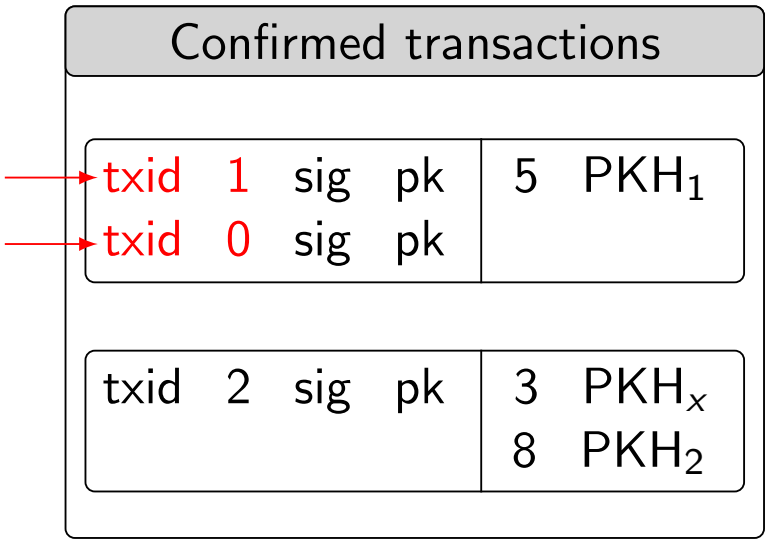
3	PKH_x
8	PKH_2

$txid_1$	0
$txid_2$	1

9	PKH_3
4	PKH_4

- A transaction can have multiple inputs
  - to spend bitcoins from previous transactions
  - reference txid and output index.
  - The txid is a transaction's identifier: double SHA-256 of its serialized data.
- A transaction can have multiple outputs
  - number of bitcoins and output PKH.

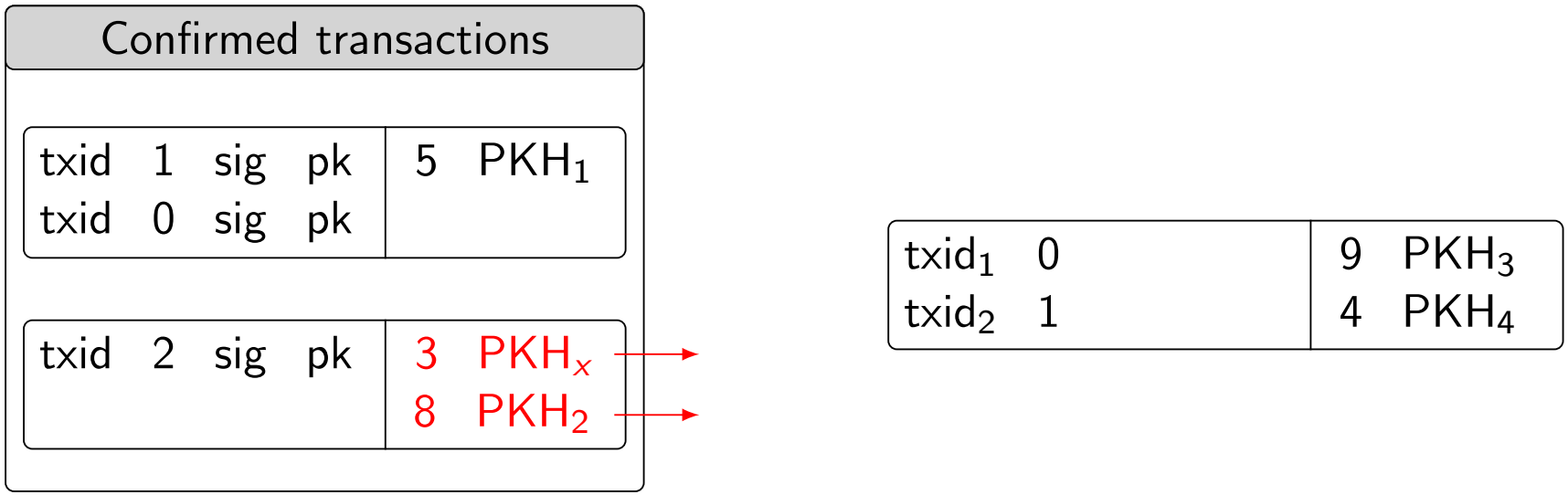
Preparing a Bitcoin transaction



txid_1	0	9	PKH_3
txid_2	1	4	PKH_4

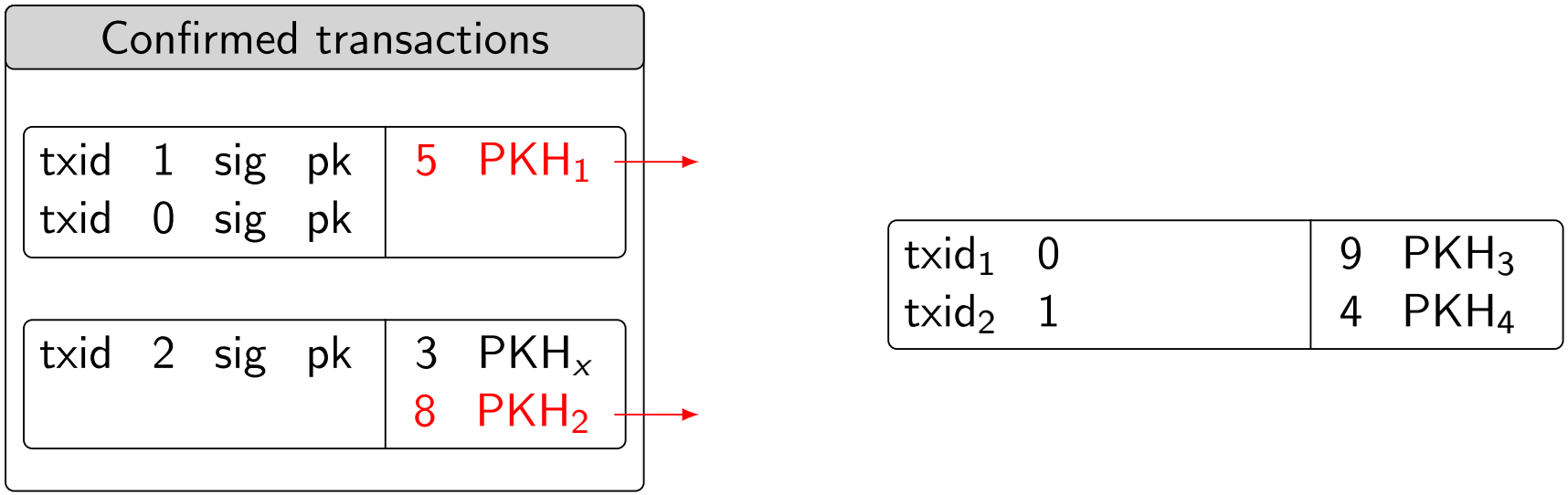
- A transaction can have multiple inputs
  - to spend bitcoins from previous transactions
  - reference txid and output index.
  - The txid is a transaction's identifier: double SHA-256 of its serialized data.
- A transaction can have multiple outputs
  - number of bitcoins and output PKH.

Preparing a Bitcoin transaction



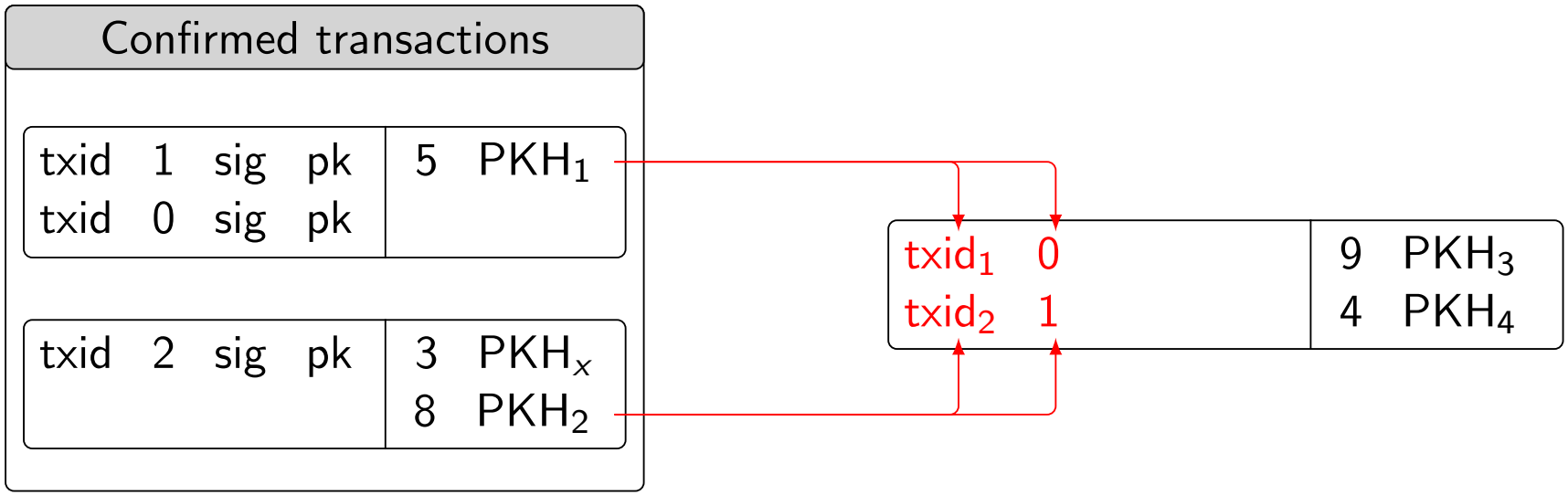
- A transaction can have multiple inputs
  - to spend bitcoins from previous transactions
  - reference txid and output index.
  - The txid is a transaction's identifier: double SHA-256 of its serialized data.
- A transaction can have multiple outputs
  - number of bitcoins and output PKH.

Preparing a Bitcoin transaction



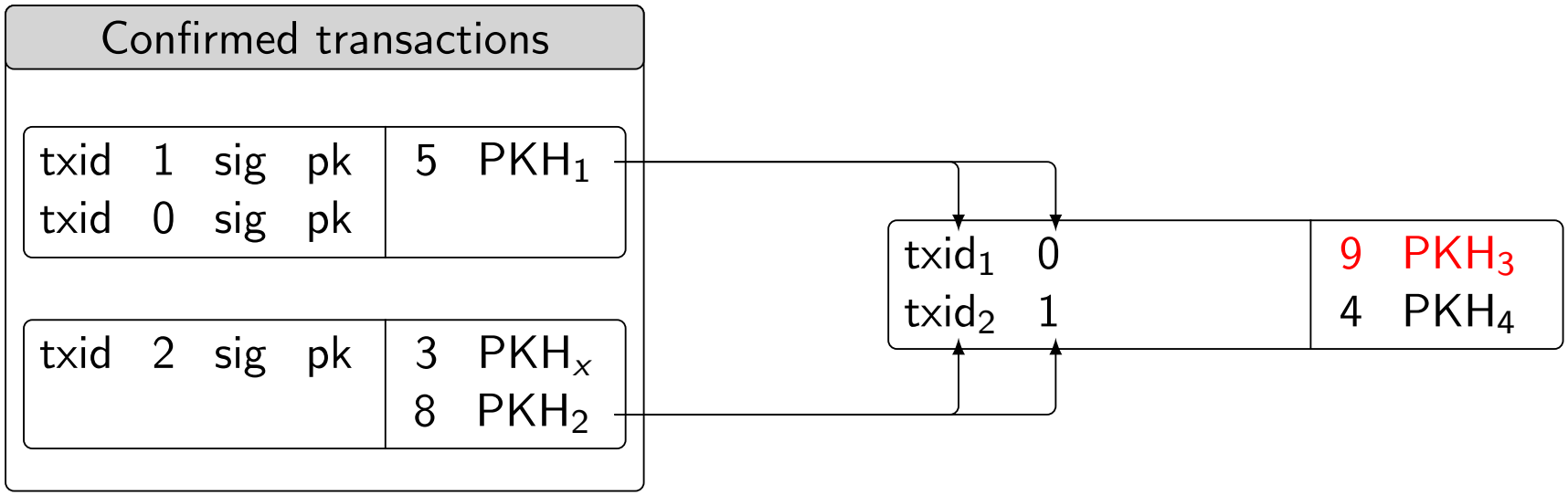
- Unspent transaction outputs (UTXOs)
  - coins received by Alice that she can spend ($5 + 8 = 13$ Ƀ)
  - by referencing the transaction (txid) and the output index (idx) in the transaction.
- Alice pays 9 Ƀ to Bob's address PKH_3 (output index 0)
  - and 4 Ƀ to herself (change) in a fresh address PKH_4.
  - An entire UTXO is consumed, so the 4 Ƀ change must be created as a new output.

Preparing a Bitcoin transaction



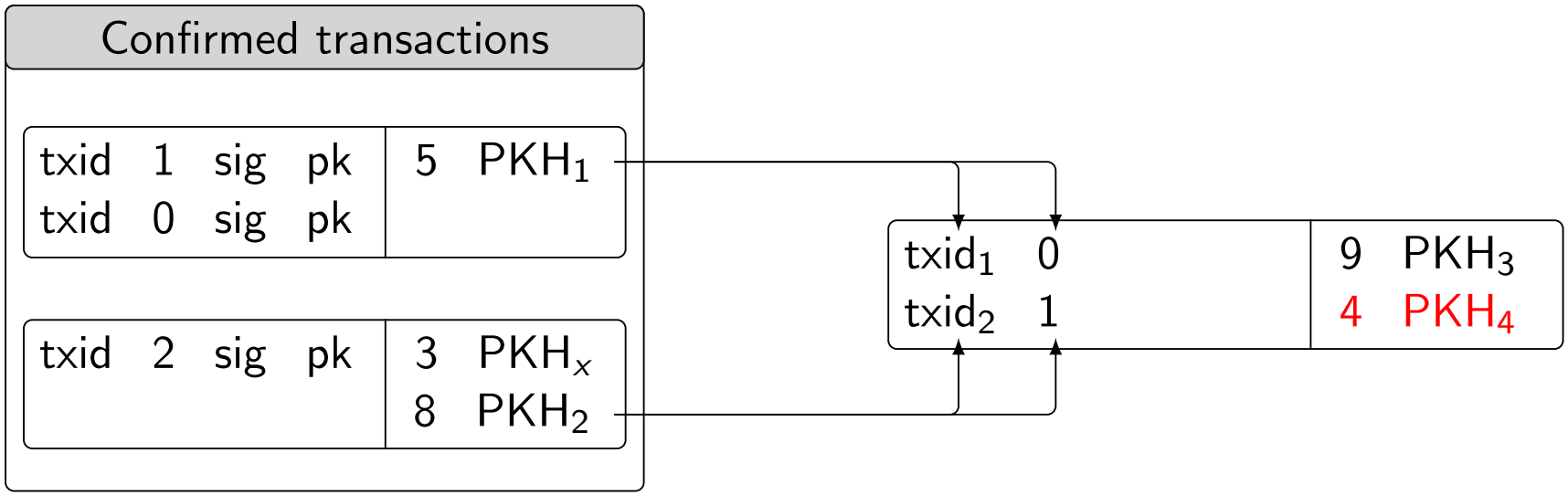
- Unspent transaction outputs (UTXOs)
  - coins received by Alice that she can spend ($5 + 8 = 13$ Ƀ)
  - by referencing the transaction (txid) and the output index (idx) in the transaction.
- Alice pays 9 Ƀ to Bob's address PKH_3 (output index 0)
  - and 4 Ƀ to herself (change) in a fresh address PKH_4.
  - An entire UTXO is consumed, so the 4 Ƀ change must be created as a new output.

Preparing a Bitcoin transaction



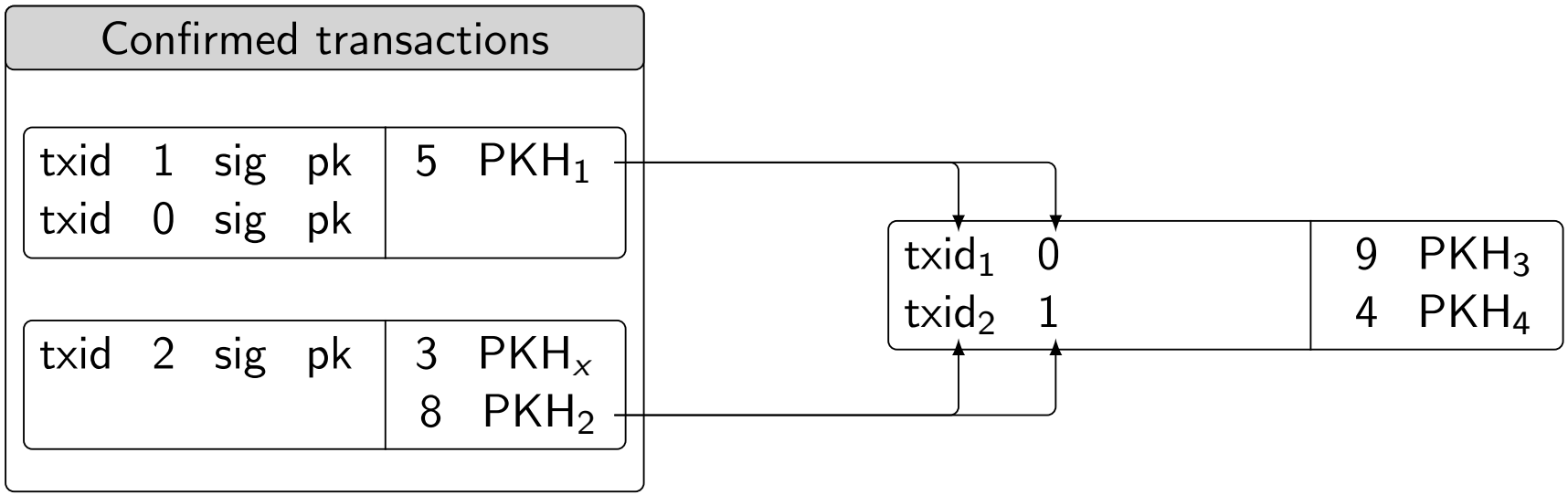
- Unspent transaction outputs (UTXOs)
  - coins received by Alice that she can spend ($5 + 8 = 13$ Ƀ)
  - by referencing the transaction (txid) and the output index (idx) in the transaction.
- Alice pays 9 Ƀ to Bob's address PKH_3 (output index 0)
  - and 4 Ƀ to herself (change) in a fresh address PKH_4.
  - An entire UTXO is consumed, so the 4 Ƀ change must be created as a new output.

Preparing a Bitcoin transaction



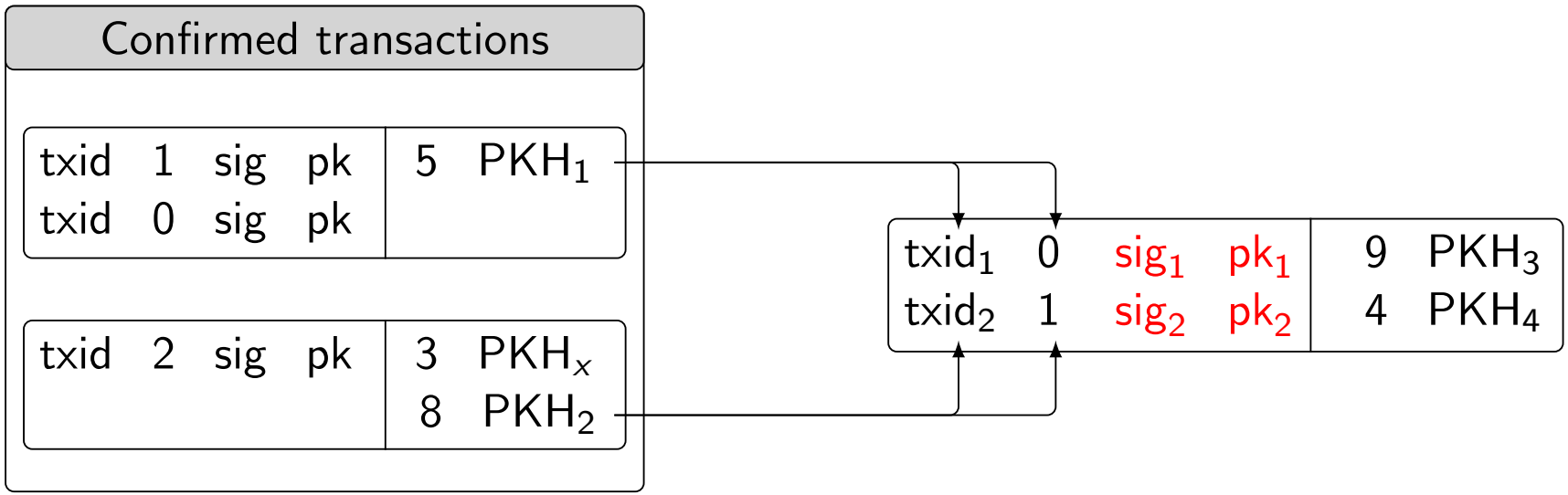
- Unspent transaction outputs (UTXOs)
  - coins received by Alice that she can spend ($5 + 8 = 13$ Ƀ)
  - by referencing the transaction (txid) and the output index (idx) in the transaction.
- Alice pays 9 Ƀ to Bob's address PKH_3 (output index 0)
  - and 4 Ƀ to herself (change) in a fresh address PKH_4.
  - An entire UTXO is consumed, so the 4 Ƀ change must be created as a new output.

Signing a transaction



- Alice knows the private-keys corresponding to the addresses PKH_1 and PKH_2
  - She can "unlock" the two UTXOs by signing the transaction.
  - She inserts public-keys pk_1 and pk_2 so that people can verify.
  - Each signature commits the entire transaction.
- Anyone can verify the transaction
  - For each input, check $HASH160(pk_i) = PKH_i$ of the spent output, and verify sig_i under pk_i.
  - Also check that the referenced output is still unspent.

Signing a transaction



- Alice knows the private-keys corresponding to the addresses PKH_1 and PKH_2
  - She can "unlock" the two UTXOs by signing the transaction.
  - She inserts public-keys pk_1 and pk_2 so that people can verify.
  - Each signature commits the entire transaction.
- Anyone can verify the transaction
  - For each input, check $\mathrm{HASH160}(\mathrm{pk}_i) = \mathrm{PKH}_i$ of the spent output, and verify sig_i under pk_i.
  - Also check that the referenced output is still unspent.

Account-based vs value-based system

- Account-based system

from	to	bitcoin
PKH_y	PKH_1	5
PKH_z	PKH_x	3
PKH_z	PKH_2	8

  - Each PKH is an account
  - One must keep the balance of each account

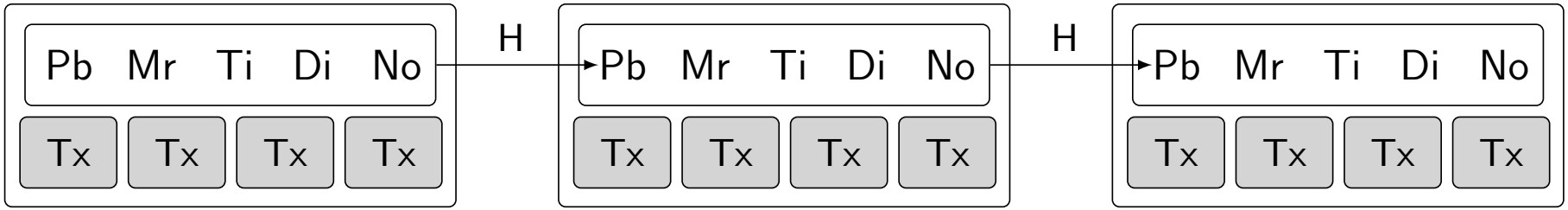
- Value-based system (Bitcoin)

txid	idx	output
txid_1	0	5 PKH_1
txid_2	0	3 PKH_x
txid_2	1	8 PKH_2

  - Each transaction output is a "coin" that can be spent only once.
  - One must keep track of the UTXO set

The blockchain

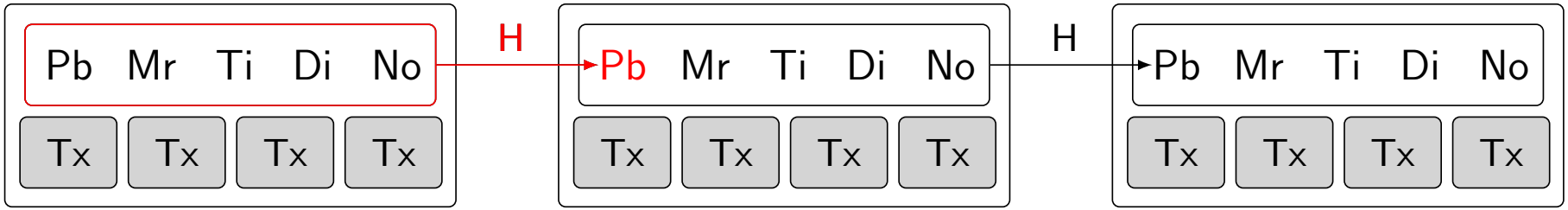
- The blockchain is a sequence of blocks of transactions
  - connected through cryptographic hashes



- The block header contains
  - the hash of the previous block header (prev_block)
  - the combined hash of the transactions (merkle_root)
  - a timestamp, the target difficulty, and the nonce.
- Security of the blockchain
  - Given the hash in the last block header, one cannot modify any previous block, unless one finds a collisision in the hash function.
  - one cannot modify any transaction in a previous block

The blockchain

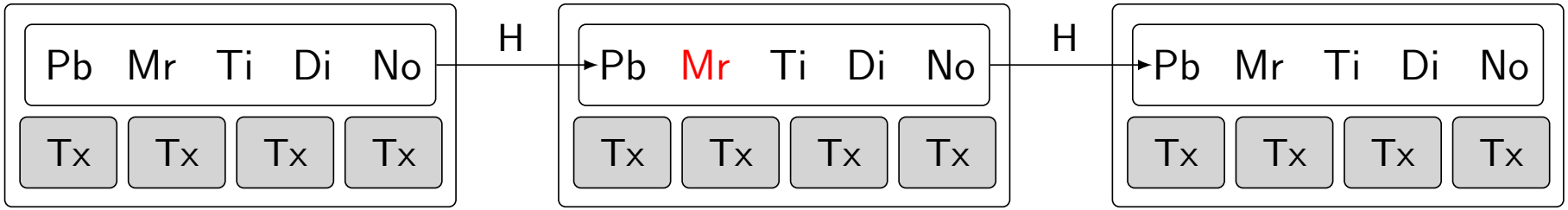
- The blockchain is a sequence of blocks of transactions
  - connected through cryptographic hashes



- The block header contains
  - the hash of the previous block header (prev_block)
  - the combined hash of the transactions (merkle_root)
  - a timestamp, the target difficulty, and the nonce.
- Security of the blockchain
  - Given the hash in the last block header, one cannot modify any previous block, unless one finds a collisision in the hash function.
  - one cannot modify any transaction in a previous block

The blockchain

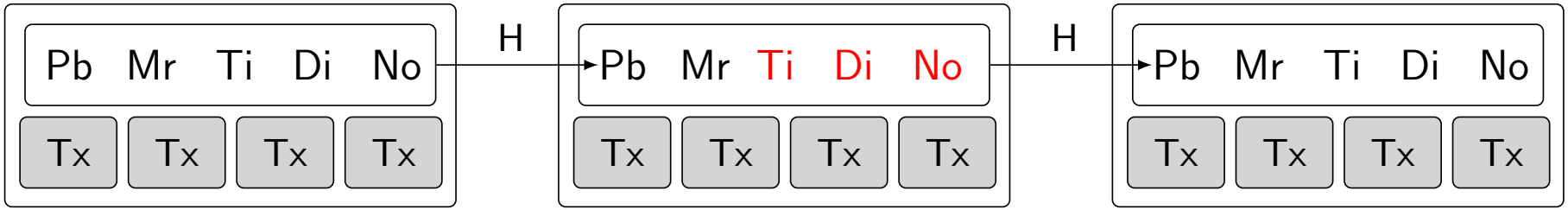
- The blockchain is a sequence of blocks of transactions
  - connected through cryptographic hashes



- The block header contains
  - the hash of the previous block header (prev_block)
  - the combined hash of the transactions (merkle_root)
  - a timestamp, the target difficulty, and the nonce.
- Security of the blockchain
  - Given the hash in the last block header, one cannot modify any previous block, unless one finds a collisision in the hash function.
  - one cannot modify any transaction in a previous block

The blockchain

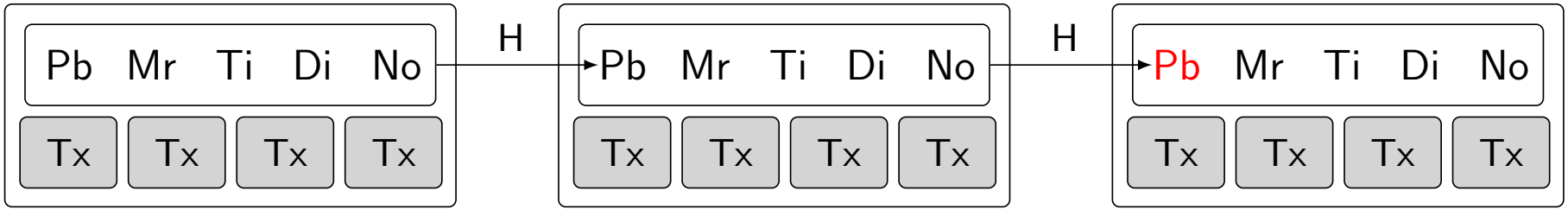
- The blockchain is a sequence of blocks of transactions
  - connected through cryptographic hashes



- The block header contains
  - the hash of the previous block header (prev_block)
  - the combined hash of the transactions (merkle_root)
  - a timestamp, the target difficulty, and the nonce.
- Security of the blockchain
  - Given the hash in the last block header, one cannot modify any previous block, unless one finds a collisision in the hash function.
  - one cannot modify any transaction in a previous block

The blockchain

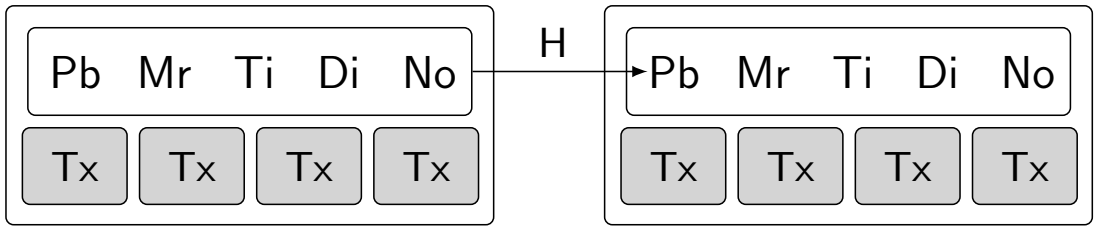
- The blockchain is a sequence of blocks of transactions
  - connected through cryptographic hashes



- The block header contains
  - the hash of the previous block header (prev_block)
  - the combined hash of the transactions (merkle_root)
  - a timestamp, the target difficulty, and the nonce.
- Security of the blockchain
  - Given the hash in the last block header, one cannot modify any previous block, unless one finds a collisision in the hash function.
  - one cannot modify any transaction in a previous block

Proof of work

- Miners compete to create the next block
  - Proof of work: requires to compute a huge number of cryptographic hashes
  - Miners are rewarded by the block subsidy and transaction fees.



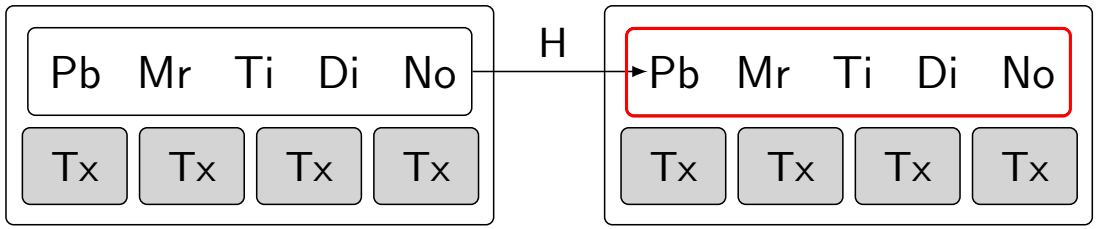
- Proof of work
  - The hash of the block header must be below the target

```
block_id:  0000000000000000094002a8b505cd2509bcbfe9f5da0a3d7ccd209887c134
target :   0000000000000000170cfe0000000000000000000000000000000000000000
```

  - by adjusting the 32-bit nonce, or changing other data.
  - Anybody can verify the proof of work by hashing the block header.

Proof of work

- Miners compete to create the next block
  - Proof of work: requires to compute a huge number of cryptographic hashes
  - Miners are rewarded by the block subsidy and transaction fees.



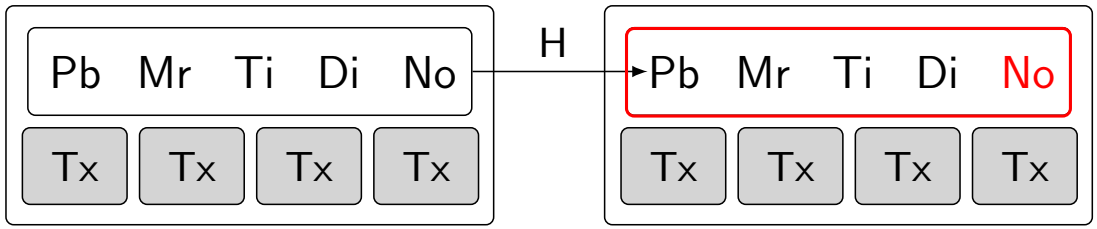
- Proof of work
  - The hash of the block header must be below the target

```
block_id:  000000000000000000094002a8b505cd2509bcbfe9f5da0a3d7ccd209887c134
target :   0000000000000000170cfe0000000000000000000000000000000000000000
```

  - by adjusting the 32-bit nonce, or changing other data.
  - Anybody can verify the proof of work by hashing the block header.

Proof of work

- Miners compete to create the next block
  - Proof of work: requires to compute a huge number of cryptographic hashes
  - Miners are rewarded by the block subsidy and transaction fees.



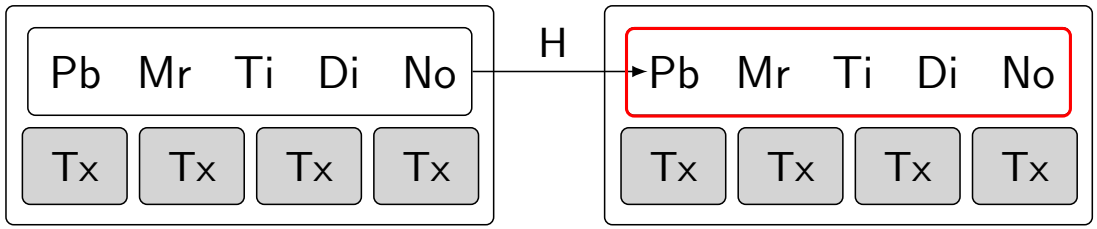
- Proof of work
  - The hash of the block header must be below the target

```
block_id:  000000000000000000094002a8b505cd2509bcbfe9f5da0a3d7ccd209887c134
target :   0000000000000000170cfe0000000000000000000000000000000000000000
```

  - by adjusting the 32-bit nonce, or changing other data.
  - Anybody can verify the proof of work by hashing the block header.

Proof of work

- Miners compete to create the next block
  - Proof of work: requires to compute a huge number of cryptographic hashes
  - Miners are rewarded by the block subsidy and transaction fees.



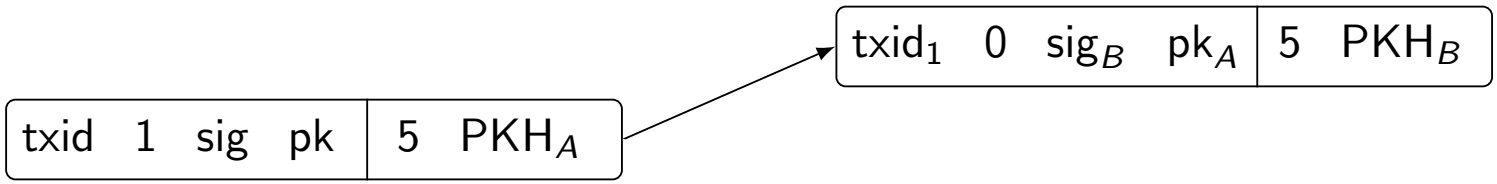
- Proof of work
  - The hash of the block header must be below the target

```
block_id:  000000000000000000094002a8b505cd2509bcbfe9f5da0a3d7ccd209887c134
target :   00000000000000000170cfe0000000000000000000000000000000000000000
```

  - by adjusting the 32-bit nonce, or changing other data.
  - Anybody can verify the proof of work by hashing the block header.

Preventing double spending

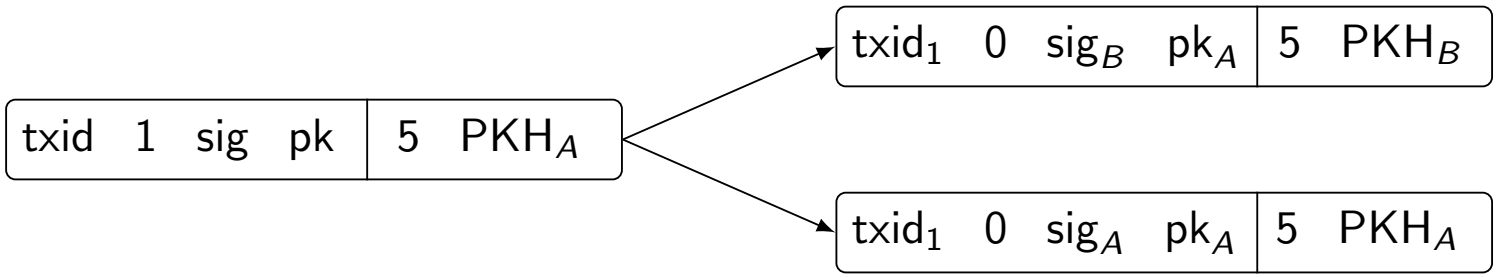
- Double spending
  - Alice prepares a transaction of 5 Ƀ to Bob in exchange of his car



  - She also prepares a double-spend transaction to herself.
  - Each is valid relative to the same prior UTXO state, but they cannot both appear in one valid chain.
  - Signatures alone do not resolve conflicting payments.

Preventing double spending

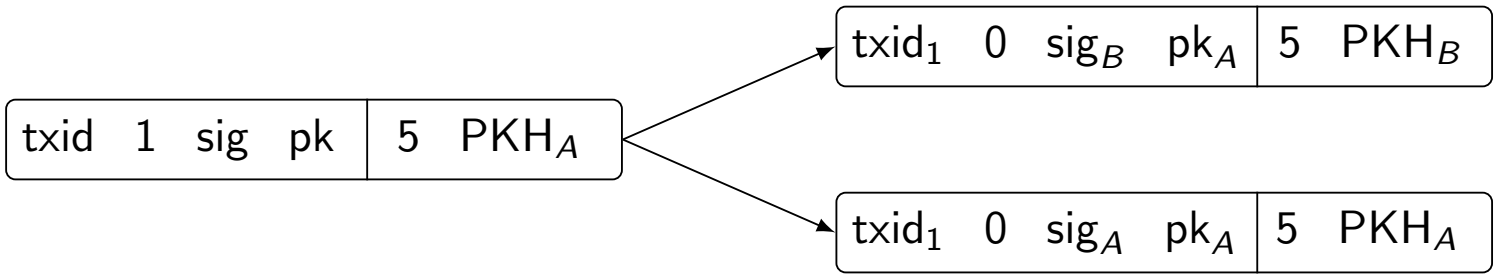
- Double spending
  - Alice prepares a transaction of 5 Ƀ to Bob in exchange of his car



  - She also prepares a double-spend transaction to herself.
  - Each is valid relative to the same prior UTXO state, but they cannot both appear in one valid chain.
  - Signatures alone do not resolve conflicting payments.

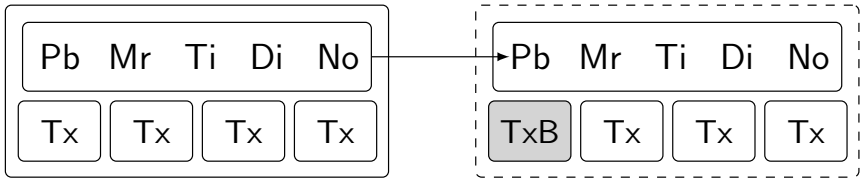
Preventing double spending

- Double spending
  - Alice prepares a transaction of 5 Ƀ to Bob in exchange of his car



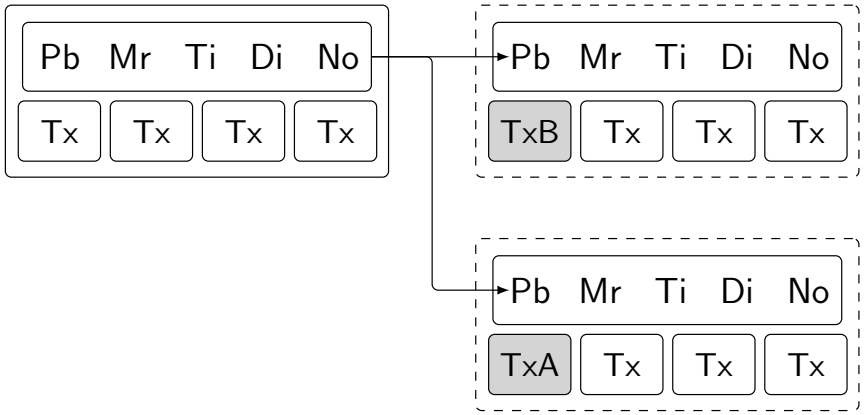
  - She also prepares a double-spend transaction to herself.
  - Each is valid relative to the same prior UTXO state, but they cannot both appear in one valid chain.
  - Signatures alone do not resolve conflicting payments.

Preventing double spending



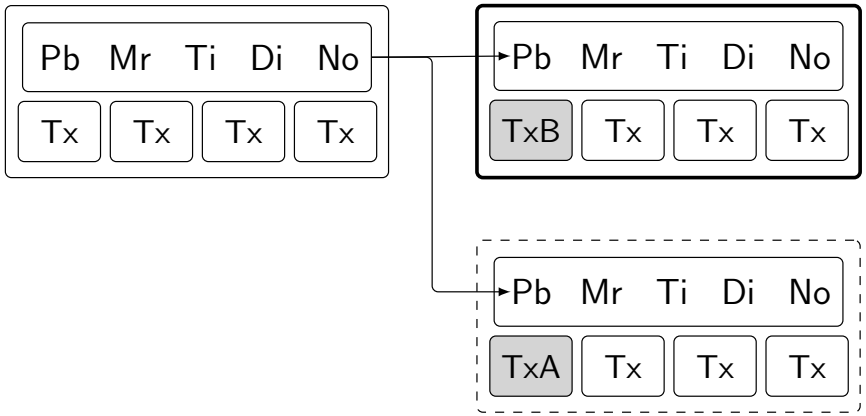
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Preventing double spending



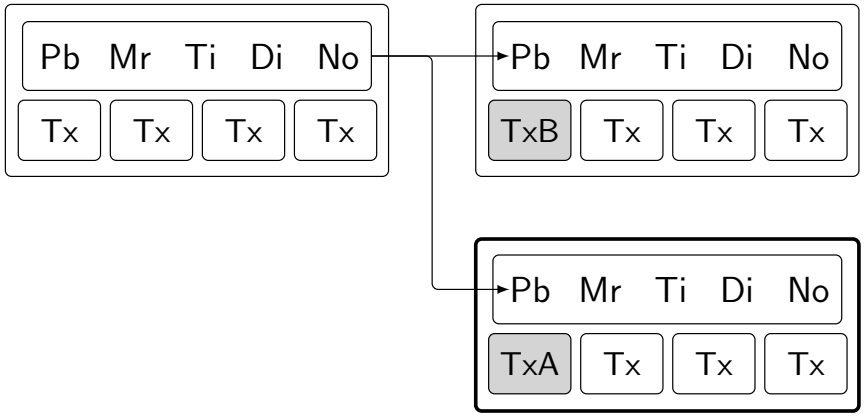
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Preventing double spending



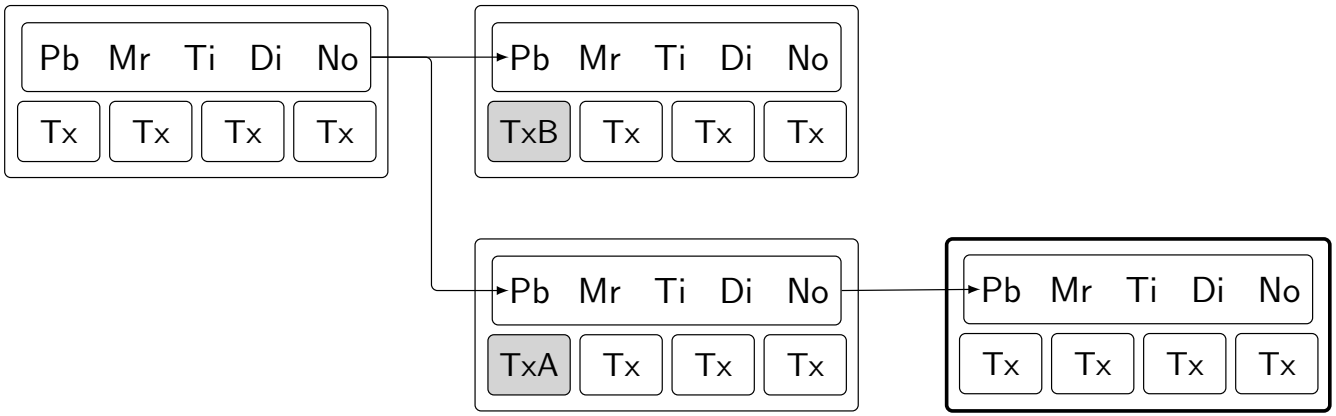
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Preventing double spending



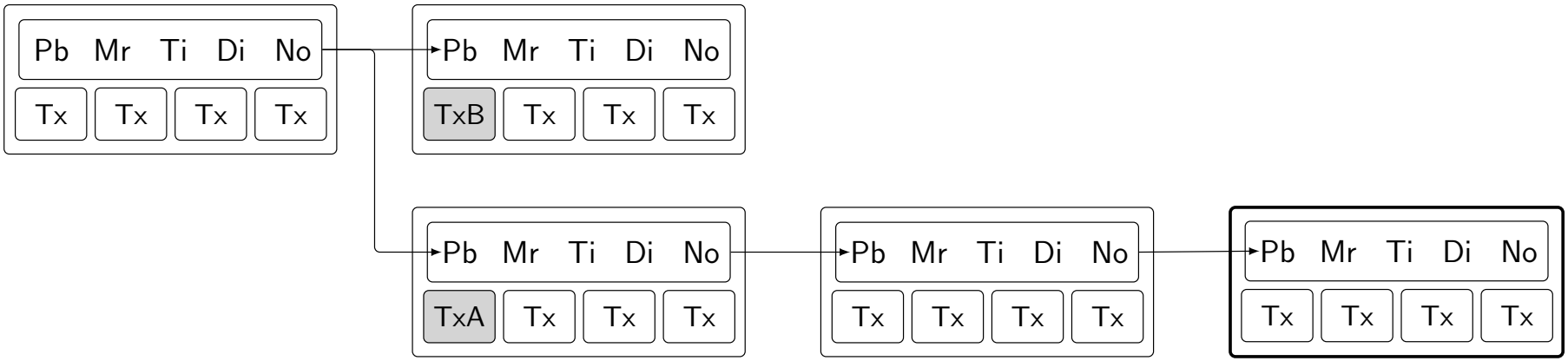
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Preventing double spending



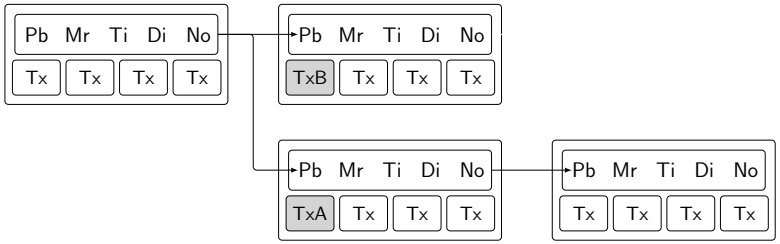
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Preventing double spending



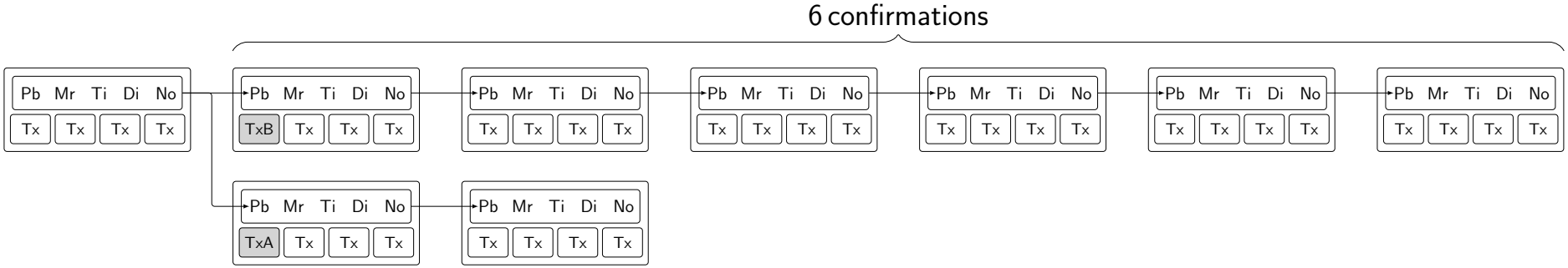
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Reducing double-spend risk



- Waiting for more confirmations reduces the probability and economic feasibility of a reversal.
  - 6 confirmations = the block containing TxB plus 5 successors.
  - In the usual minority-attacker model, reversal probability decreases with confirmation depth.
  - A sustained majority of hash power undermines this protection.

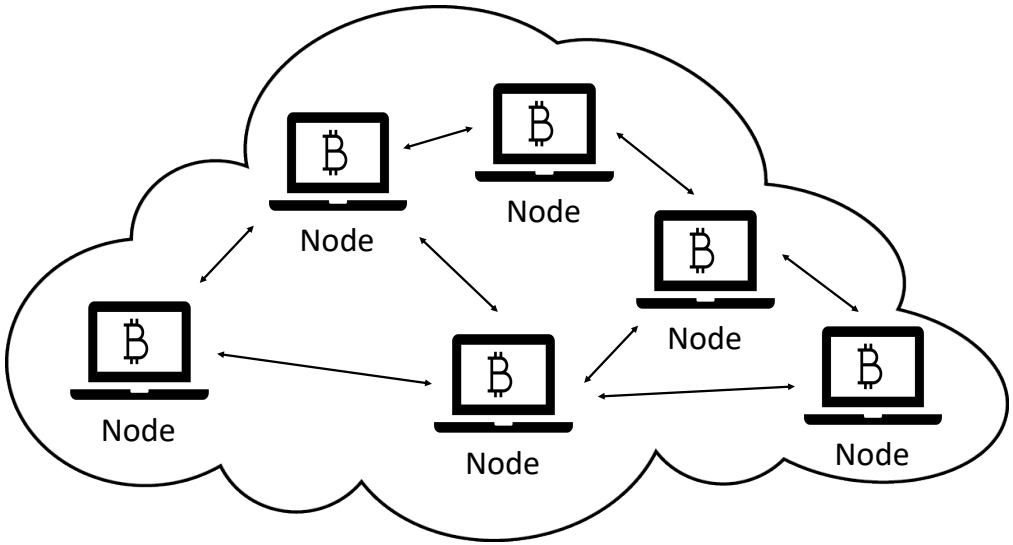
Reducing double-spend risk



- Waiting for more confirmations reduces the probability and economic feasibility of a reversal.
  - 6 confirmations = the block containing TxB plus 5 successors.
  - In the usual minority-attacker model, reversal probability decreases with confirmation depth.
  - A sustained majority of hash power undermines this protection.

Conclusion

- Bitcoin is a decentralized payment protocol
  - Transactions and blocks are relayed through a peer-to-peer network
  - No central source of authority needed



- Other concepts in Bitcoin
  - Difficulty adjustment: a block mined roughly every 10 mins.
  - Payment script: more flexibility.
  - Lightweight wallet: fast verification using Merkle tree.