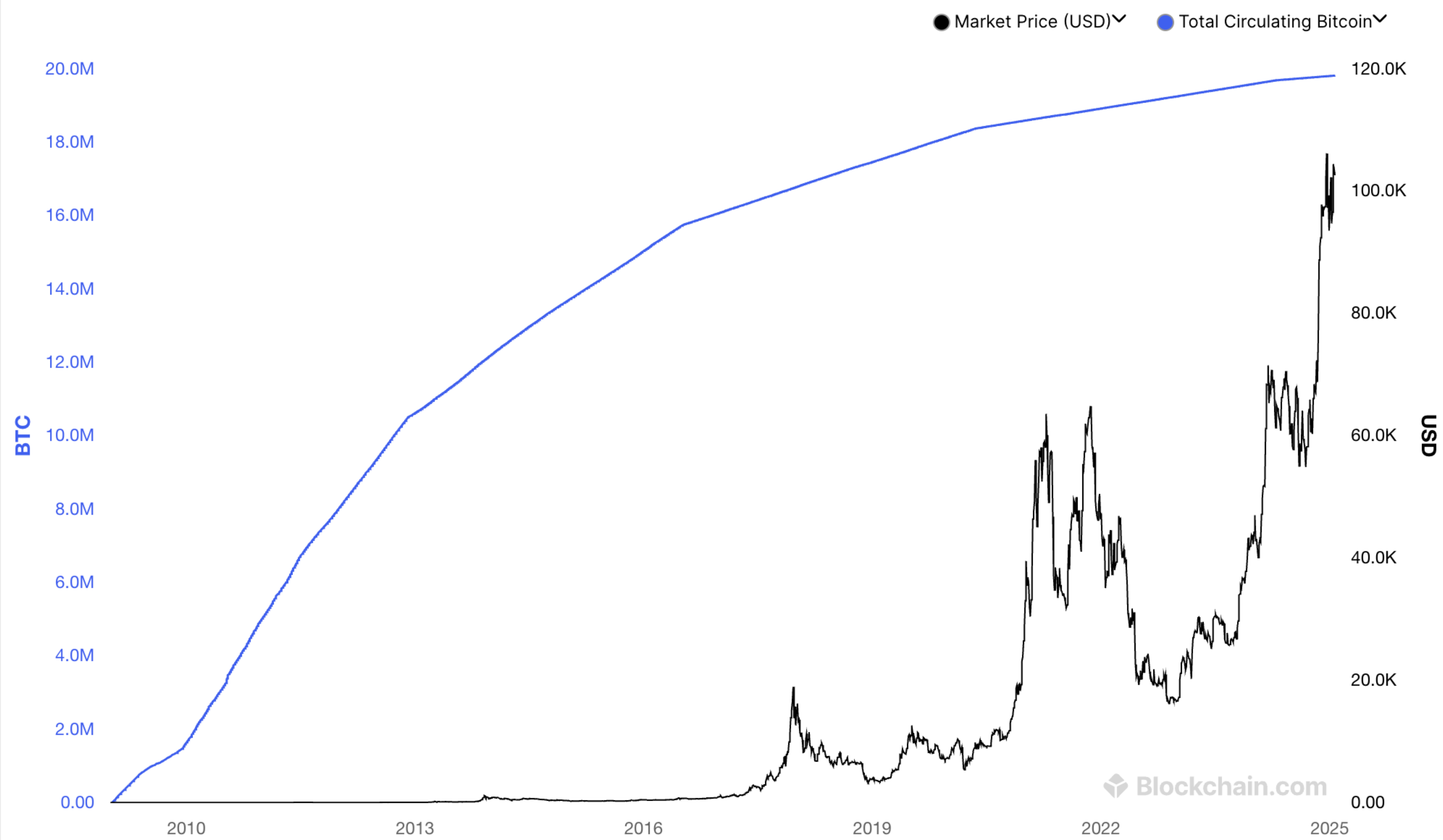
Bitcoin Mechanics

Thierry Sans

Mined Bitcoin and Price



Bitcoin in a Nutshell

- Uses Elliptic Curve Public Keys (secp256k1) and ECDSA signature algorithm
- UTXO Blockchain (Unspent Transaction Output)
- Consensus : Proof of Work
- Block time : ~10 minutes
- Bitcoin are created through block reward (a.k.a coinbase transaction)

Bitcoin Properties

You need to convince 51% of the mining power to

- remove a block **(safety)**
- prevent a transaction to be confirmed **(liveness)**

Anatomy of a Bitcoin Transaction

Field	Size (in bytes)	Description
Version	4	Transaction version number
Input Count	1-9	Number of inputs in the transaction
Inputs	41+ (per input)	List of transaction inputs (UTXOs being spent)
Output Count	1-9	Number of outputs in the transaction
Output	34+ (per output)	List of transaction outputs (recipients and amounts)
Locktime	4	Specifies when the transaction can be included in a block
Witness (signatures)	variable	Witness data for SegWit transactions

Building the blockchain

1. Transactions are broadcasted on the network
2. The nodes verify each transaction before adding them to their local mempool of unconfirmed transactions
3. Miners select transactions from the mempool (maximizing the Maximal) Extractable Value MEV) and start mining the next block
4. Once a nonce matching the current difficulty is found, the miner broadcast the block on the network
5. The nodes verify each block before adding it to the blockchain. The transactions are confirmed and the mempool is updated

➡ New transactions cannot use input UTXO of transactions that have not been confirmed yet

✓ Data is always consistent for a given chain and no double spending attack for a given chain (however there might be competing chains with double spent UTXOs)

Mining awards

➡ Miners verify/broadcast blocks transactions and broadcast and are rewarded for that work

- **Coinbase transaction** (first transaction in the block) Currently 3.125 BTC - Block reward halves every four years The only way BTC is created (max 21M BTC in total that will be reached 2140)
- and/or **Transaction Fees** (Total Input Value – Total Output Value)

Transaction Fee

How to calculate the fee

- Determine the current fee rate per byte
  (depends on network traffic)
- Estimate the size of the transaction in bytes (inputs + outputs)
- Multiply the transaction size by the fee rate to get the total fee

➡ Adding a fee is not an obligation but necessary in practice for the transaction to be picked

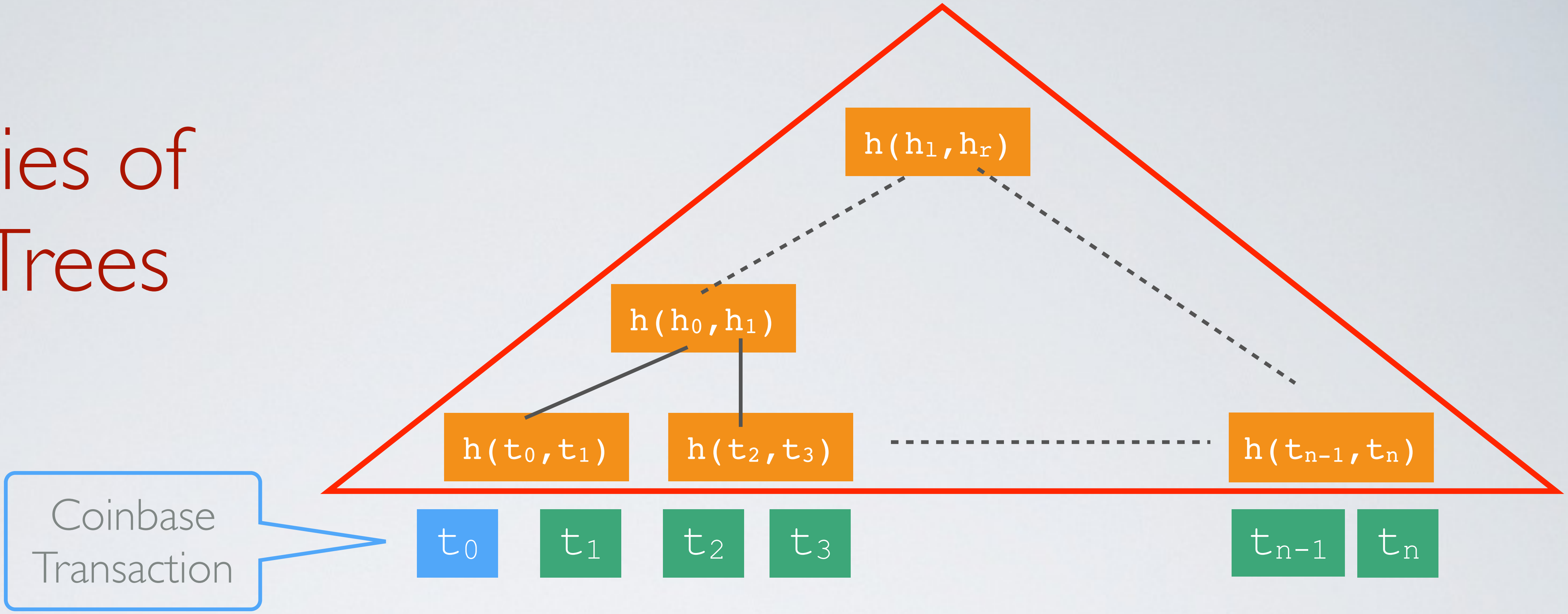
◉ A transaction with an underpaid fee may remain unconfirmed for a long time, especially during network congestion

✓ In practice, fee can be used to cancel/modify an unconfirmed transaction
(by emitting a new transaction for the same utxo with higher fee)

Anatomy of a Bitcoin Block (80 bytes)

Field	Size (in bytes)	Description
Version	4	Block version number
Previous Block Hash	32	Hash of the previous block header
Merkle Root	32	Hash summarizing all transactions in the block
Timestamp	4	UNIX epoch time when the block was mined
Difficulty Target	4	Compact representation of the current mining difficulty
Nonce	4	Number adjusted by miners to meet the difficulty target

Properties of Merkle Trees



Why using a Merkle-Tree to record all transactions in the block rather than a "flat list" such as $\mathtt{H(T_0 + T_1 + ... + T_n)}$?

✓ Because it is easier to prove than a given transaction belongs to a block

- Using flat list, need all **n** transactions to build the proof
- Using "Merkle Tree, just $\log_2(n)$ intermediate hashes to build the proof

Block Size Limit

Before 2017 : 1 MB (~2000 transactions/block)

After 2017 and the *Segregated Witness upgrade* (BIP-141)

➡ The signatures (a.k.a witness) are stored outside of the main block structure allowing transaction to use less space

✓ Each block has a maximum weight of 4 million weight units (new metric introduced) which is roughly 4 MB

Propagation Time

According to the paper "*Information propagation in the bitcoin network*" by Decker and Wattenhofer (2013):

> *The **median time** until a node receives a block is **6.5 seconds** whereas **the mean** is at **12.6 seconds**.*
>
> *The long tail of the distribution means that even **after 40 seconds there still are 5% of nodes that have not yet received the block***

Bitcoin Script

The language

Input and Output addresses are actually scripts

- Stack-based language (simplistic)
- Cryptography primitives
- No loop (not Turing complete ... but no halting problem)

See all instructions
https://wiki.bitcoinsv.io/index.php/Opcodes_used_in_Bitcoin_Script

The mechanics

`scriptPubKey` (transaction output)

the locking script that specified the condition that needs to be fulfilled to use the UTXO

`scriptSig` (transaction input)

the unlocking script (transaction input) provided by the user who wants to use the UTXO

➡ The program `scriptSig|scriptPubKey` must return `true` for a transaction to be valid

Op Codes

Op Code	Description
OP_CHECKSIG	Takes (i.e pull from stack) public key and signature and returns (i.e push) **TRUE** or **FALSE** if the signature match
OP_DUP	Duplicates the element at the top of the stack
OP_EQUAL	Takes two inputs and returns **TRUE** or **FALSE** if the two values are equal
OP_VERIFY	Marks the transaction as valid if the top of the stack is **TRUE**
OP_EQUALVERIFY	Same as **OP_EQUAL** combine with **OP_VERIFY**
OP_CHECKMULTISIG	Similar to **OP_CHECKSIG** but checking multiple in a row
OP_HASH256	Takes an input and returns it hash
OP_MAX	Takes two values and returns the biggest one
OP_RETURN	Abort and fail

Pay to Public Key Hash (P2PKH)

The payer sends bitcoin to another address

```
scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
scriptSig:    <sig> <pubKey>
```

✓ The UTXO does not reveal Alice's public key

Pay to Script Hash (P2SH)

The payer can specify a redeeming script (BIP16)

```
scriptPubKey: OP_HASH160 <redemptionScriptHash> OP_EQUAL
scriptSig:     [<sign>...<sig>] <redeemScript>
```

✓ Can specify complex conditions for when UTXO can be spent

Example of P2SH : Multi-Signature

Spending a UTXO requires 2-out-of-3 signatures

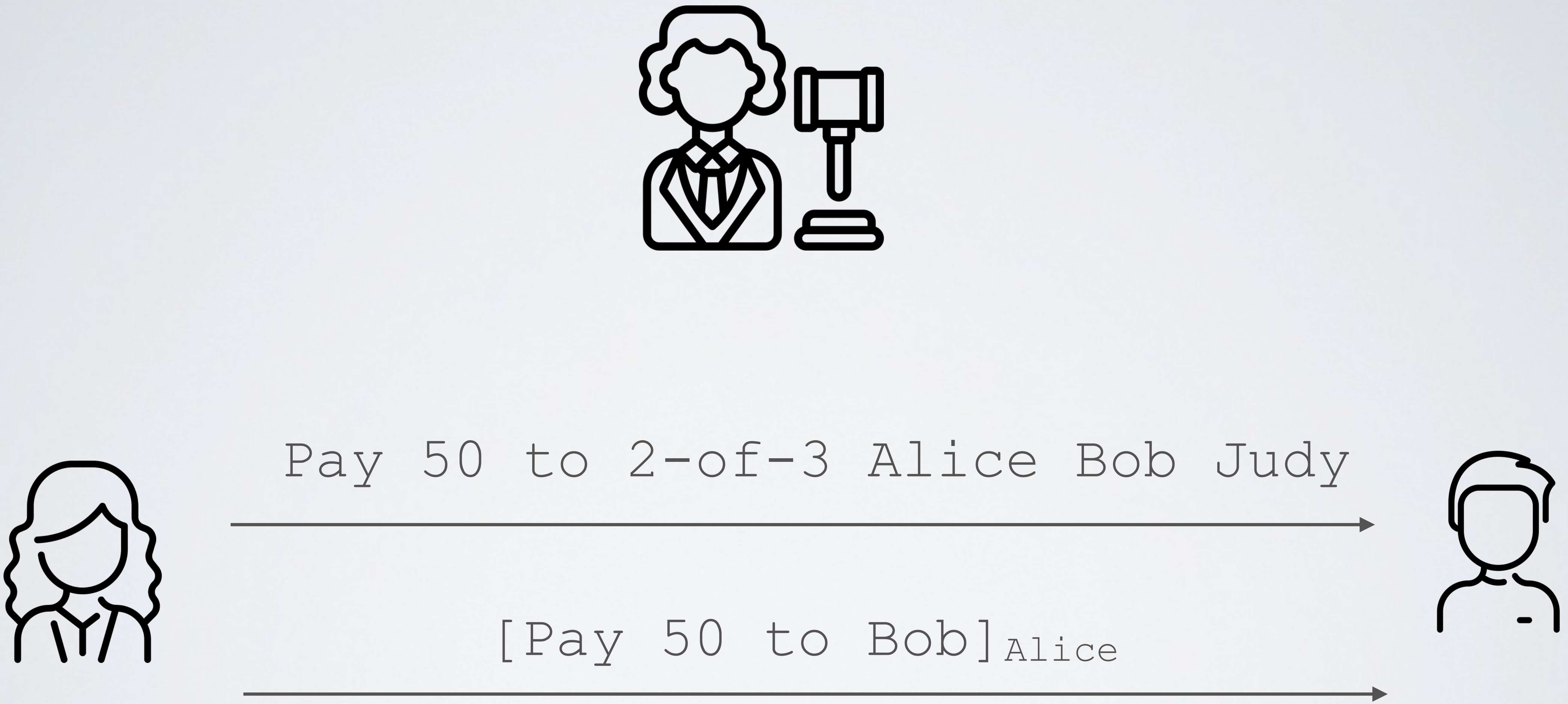
```
scriptPubKey: <2> <pubKey1> <pubKey2> <pubKey3> <3> <OP_CHECKMULISIG>
scriptSig:    <0> <sig1> <sig3> <redeemScript>
```

Null Data

This script is used to store arbitrary data (limit 40 bytes)

```
scriptPubKey: <OP_RETURN> <DATA>
```

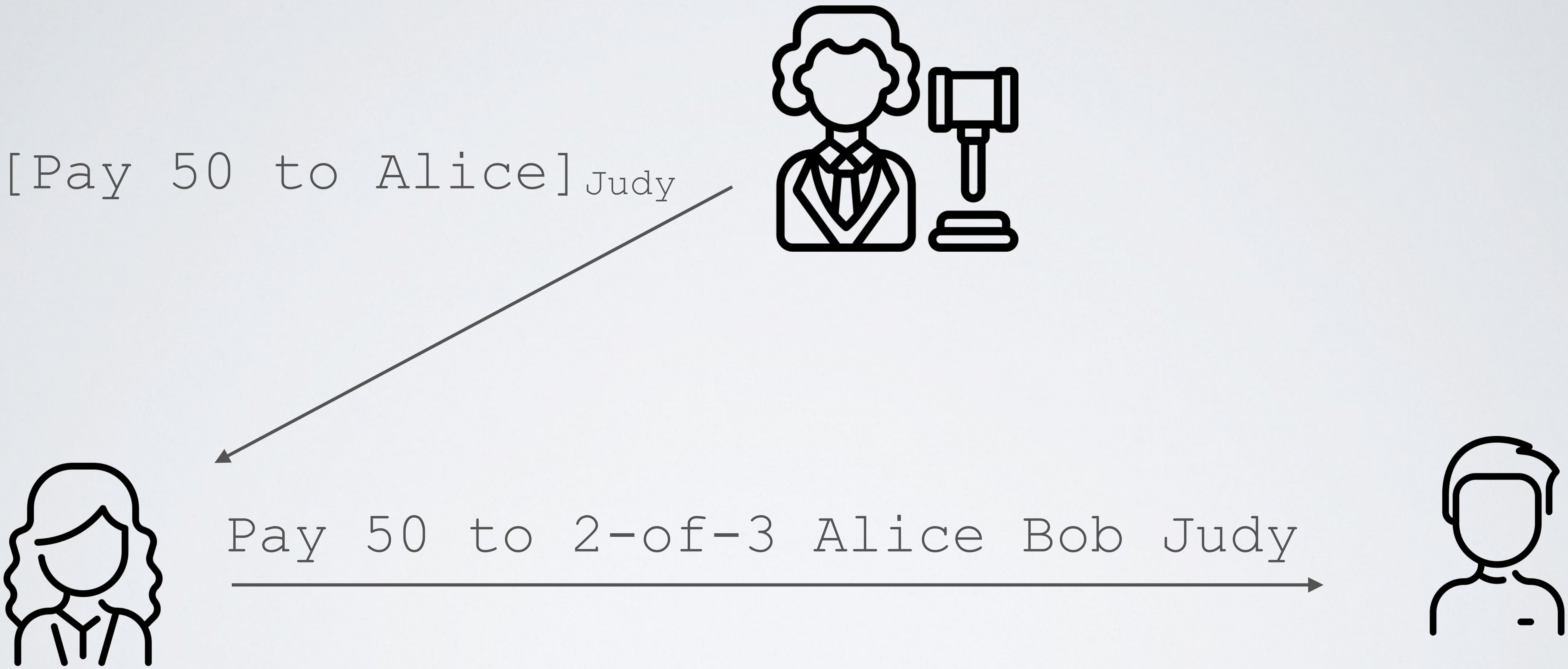
Escrow Transactions



Pay 50 to 2-of-3 Alice Bob Judy

[Pay 50 to Bob]$_{\texttt{Alice}}$

Escrow dispute



[Pay 50 to Alice]$_{\text{Judy}}$

Pay 50 to 2-of-3 Alice Bob Judy