Writing Smart Contracts

Dionysis Zindros

Smart Contracts Day
Athens, March 2017

We talked all about what smart contracts are...

...but what does a *real* smart contract look like?

- Let’s talk about writing actual Smart Contracts with code
- Using the first ever blockchain: Bitcoin

In this talk, we’ll get slightly more technical :)

Outline

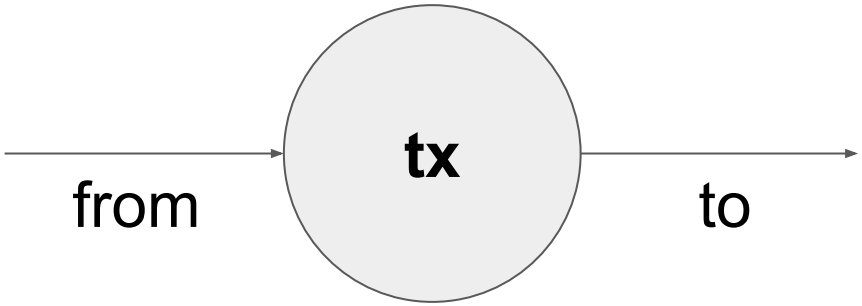
- The bitcoin blockchain
- The transaction graph
- Inputs and outputs
- Unspent transaction outputs
- Digital signatures
- Creating and resolving encumbrances
- The stack-based bitcoin computer
- A first contract: Paying someone some money

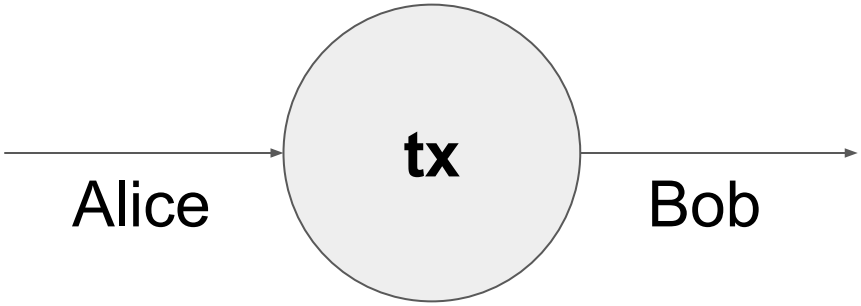
Enter bitcoin

- Very simple “smart contract” capabilities
- Expressibility very limited
  - Can express ideas such as a “payment contract”
  - We’ll go through that in this talk
- Good introductory example to practical smart contracts
- Significantly extended by blockchains created after Bitcoin
  - e.g. Ethereum, Cardano
  - Darryl will talk more about this in a bit

Transactions

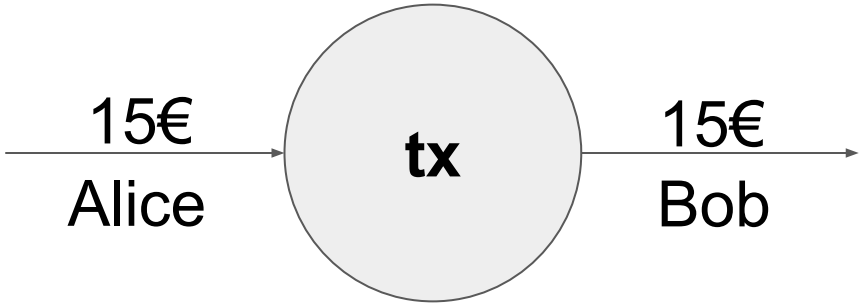
- Bitcoin’s basic structure: A **transaction** (tx)
- A transaction transfers money from an old owner to a new one

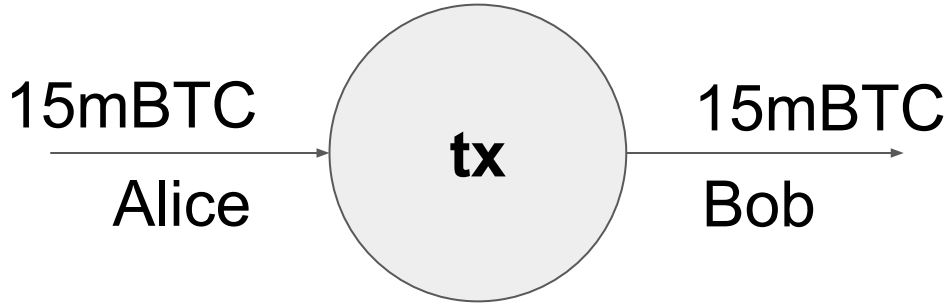


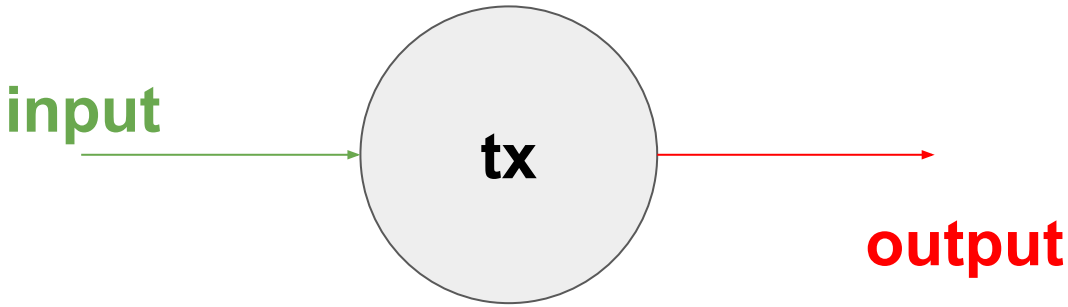

tx
Alice
Bob

Transaction edges

- I’ll illustrate a transaction as a **node (circle)**
- It has **incoming** and **outgoing edges (lines)**
- The incoming edge shows **who pays**
- The outgoing edge shows **who is paid**
- The nodes **do not illustrate** owners, but transactions
- **The edges have owners**
- Each edge has a **weight** (number associated with it) which is its nominal monetary value

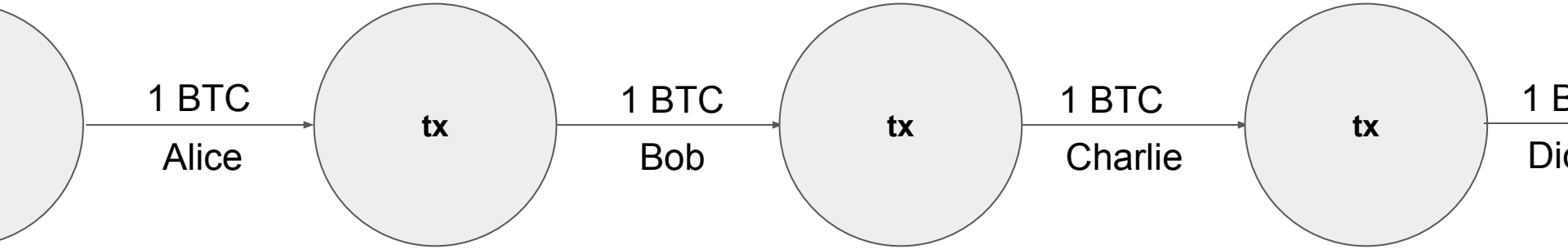

15€
Alice
tx
15€
Bob


15mBTC
Alice
tx
15mBTC
Bob


input
tx
output

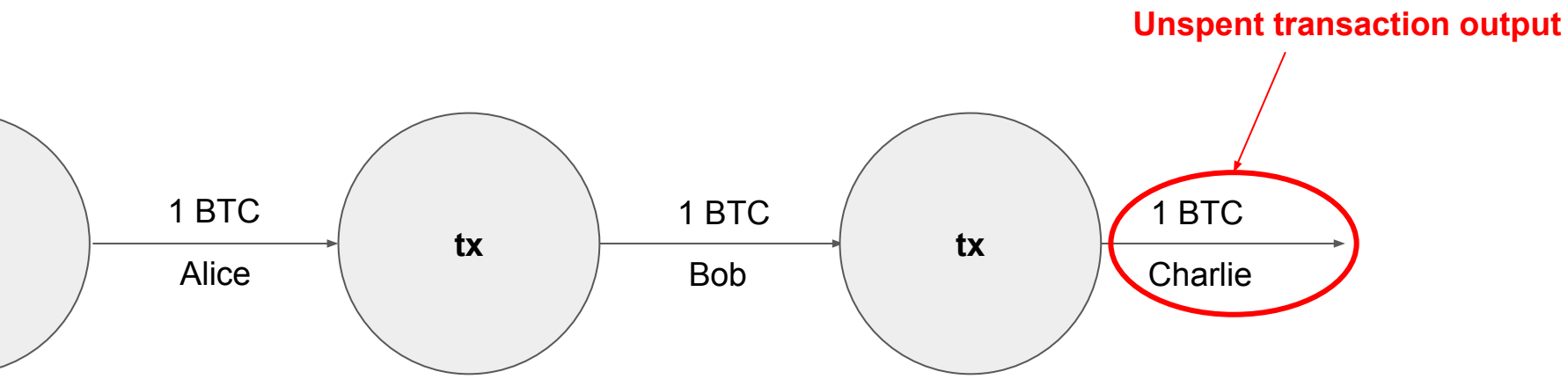
The transaction graph

- A graph is a group of edges connecting various nodes together
- Payments are made by **connecting** transaction nodes
- Money is a **chain of transactions**

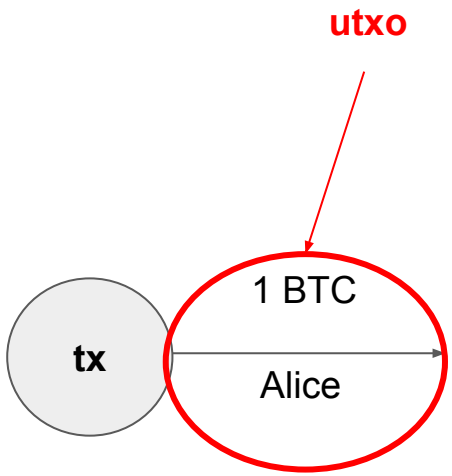


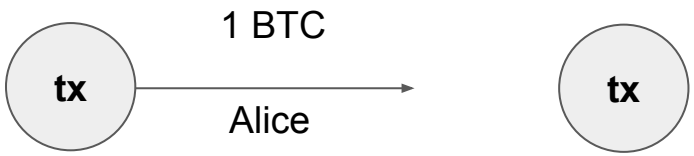
Unspent money

- Money that can be spent is **unspent money**
- It is the **dangling outgoing edges** of transactions (utxo)

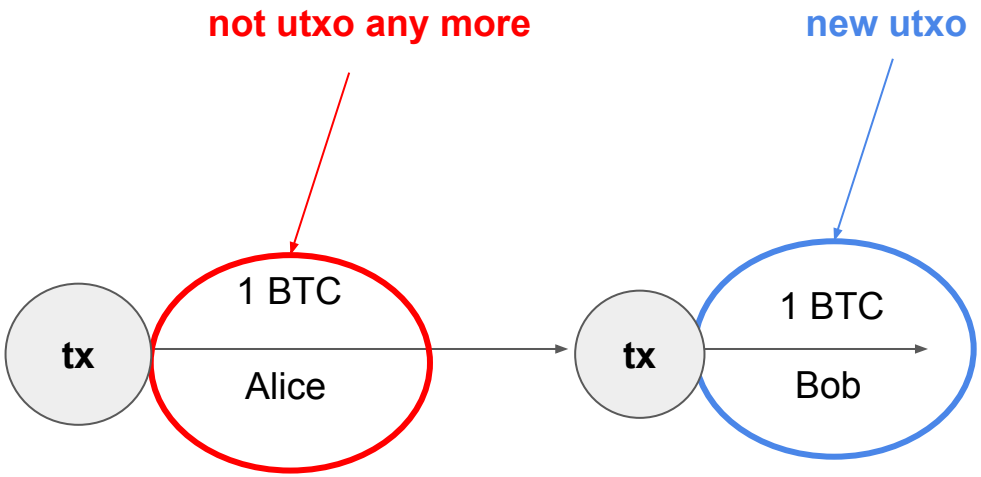



tx
1 BTC
Alice


utxo
1 BTC
tx
Alice

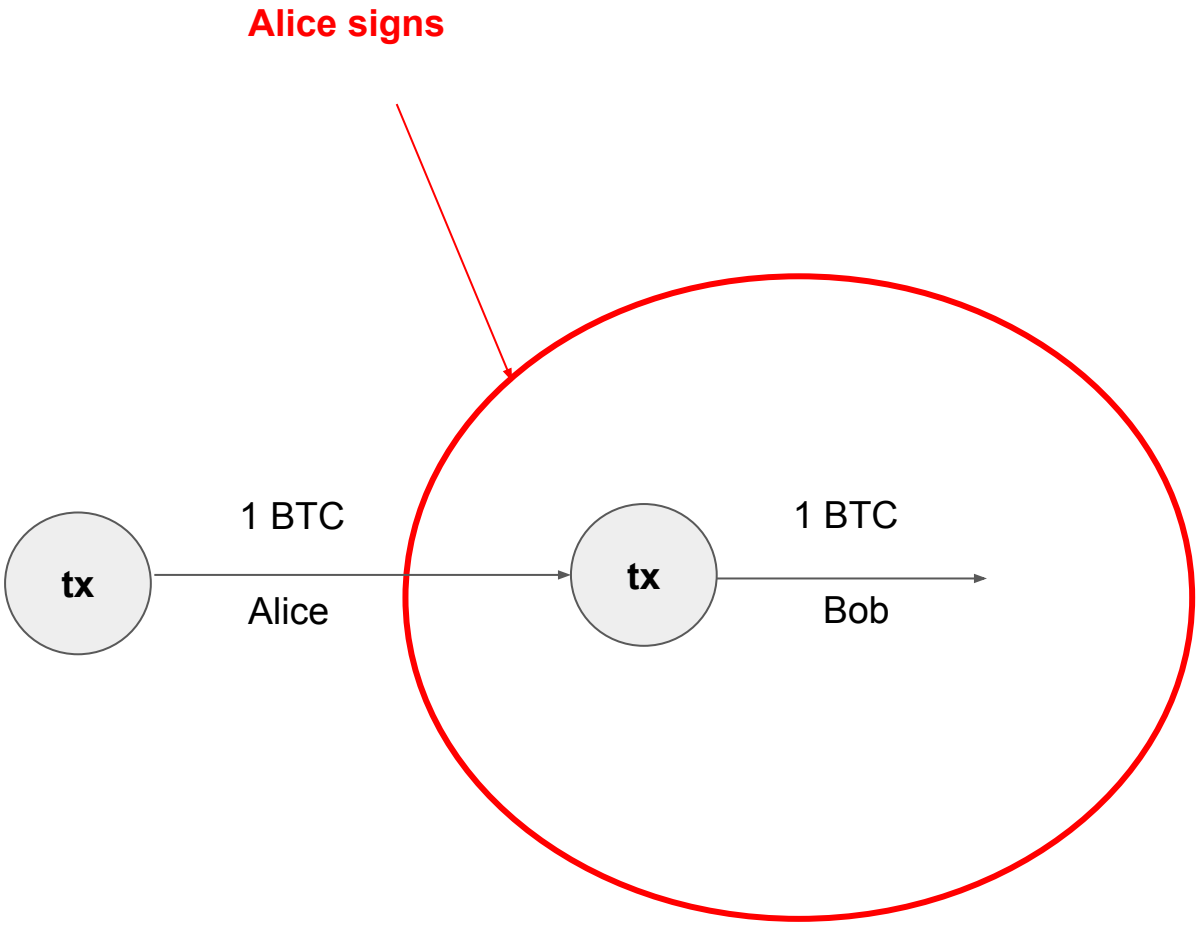

1 BTC
tx
Alice
tx

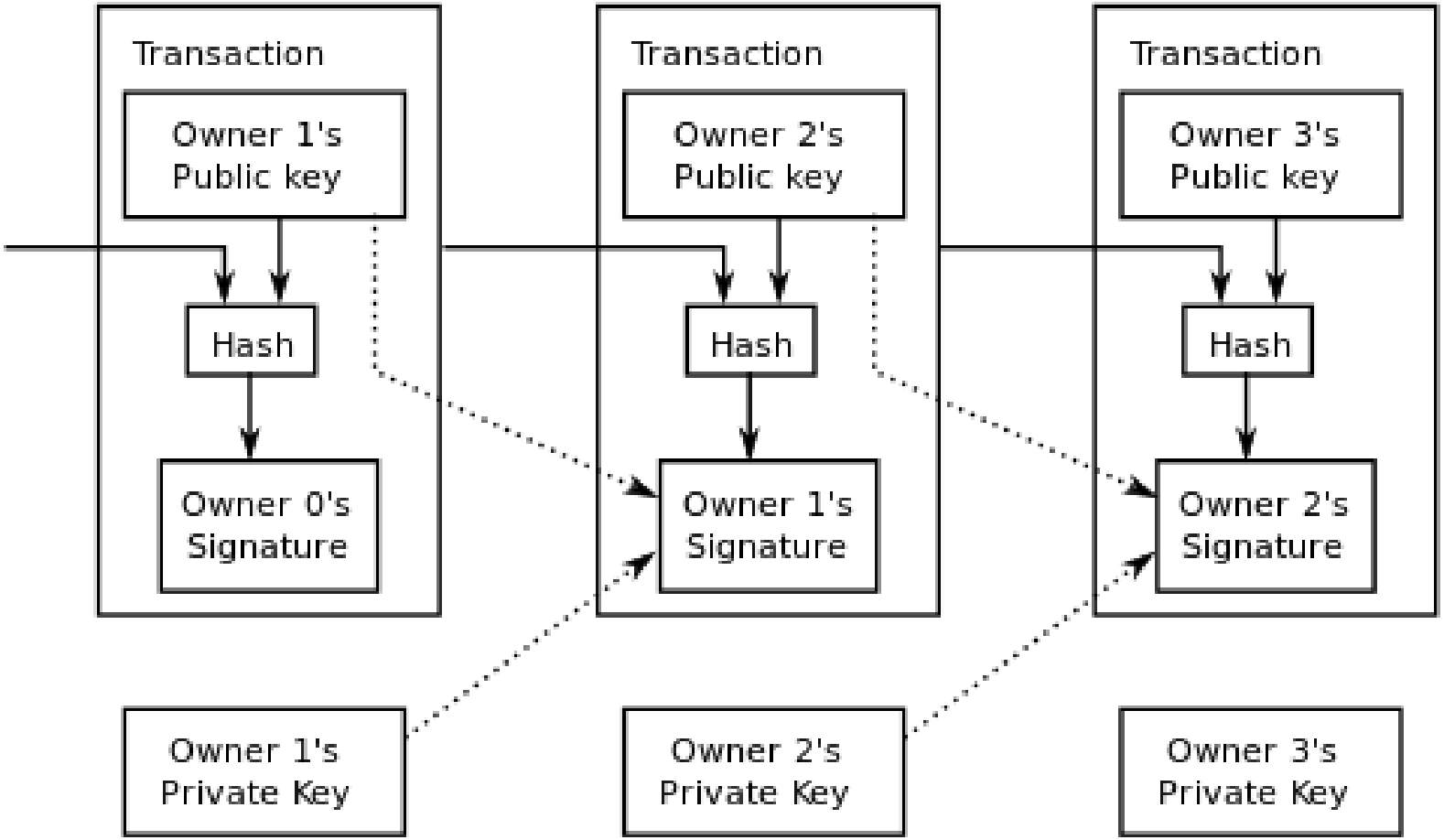

1 BTC
tx
Alice
1 BTC
tx
Bob


not utxo any more
new utxo
1 BTC
Alice
tx
tx
1 BTC
Bob

Proof of ownership

- **Digitally sign the UTXO that I want to spend with the new tx details**
- This ensures I’m the true owner of the UTXO
- The new transaction must include the tx
- This way I ensure I give permission to the **new owner** and my signature **cannot be forged** towards a wrong owner with just copying it


Alice signs
tx
1 BTC
Alice
tx
1 BTC
Bob

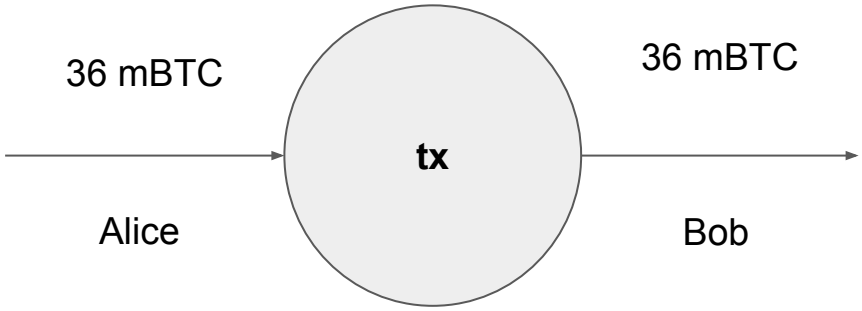

Transaction
Owner 1's
Public key
Hash
Owner 0's
Signature
Transaction
Owner 2's
Public key
Hash
Owner 1's
Signature
Transaction
Owner 3's
Public key
Hash
Owner 2's
Signature
Owner 1's
Private Key
Owner 2's
Private Key
Owner 3's
Private Key

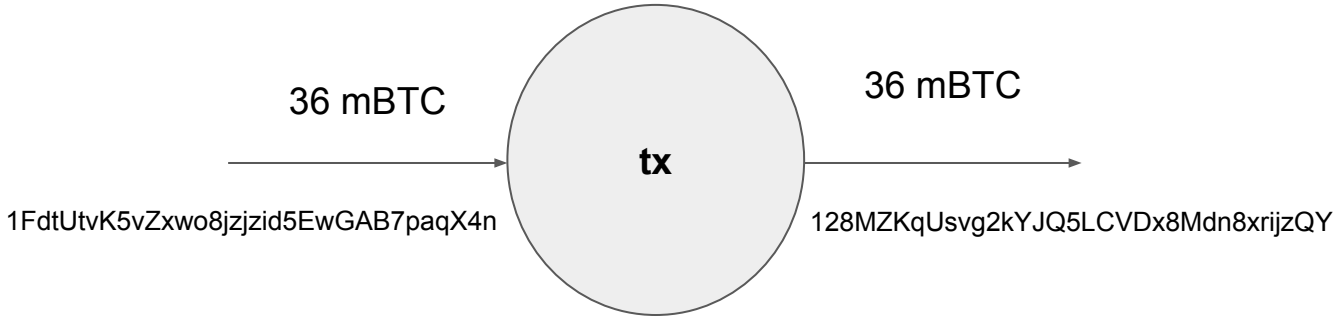
Bitcoin script: The original smart contracts

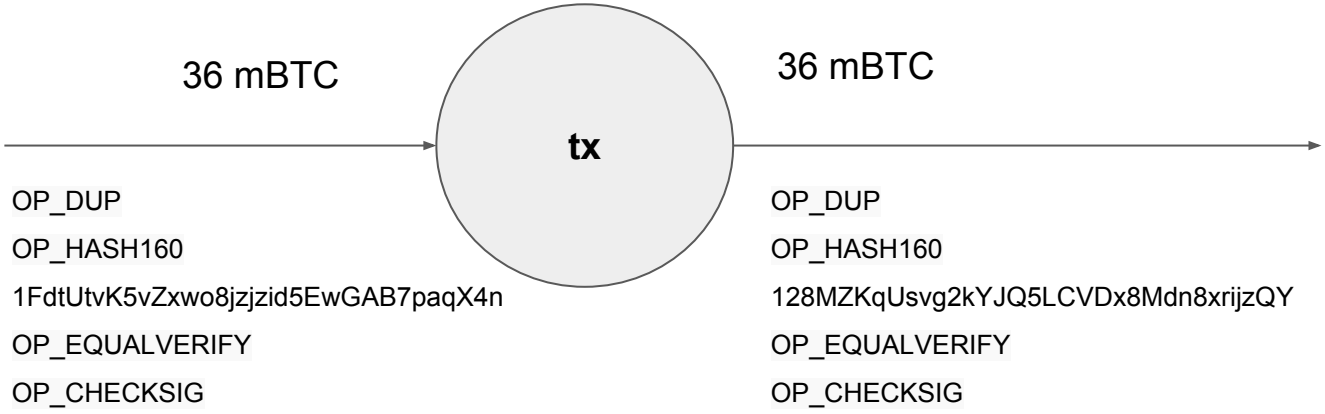
- People talk about Smart Contracts in Ethereum
- The original Smart Contract language is bitcoin!
- Bitcoin provides a language for **expressing simple smart contracts**
- What can it express?
  - Alice owns some money
  - Alice and Bob own money together
  - Micropayments - continuous transfer of value

Bitcoin script: Encumbrances

- The owner of an edge on the bitcoin tx graph is **not** just bitcoin address!
- It is a **computer program** which decides whether the edge can be spent
- It is written bitcoin script
- A dangling edge is **an encumbrence**
- This program is called a **scriptPubKey**
- This is the program **the verifier runs**
- This allows us to express more **complicated ownerships**


36 mBTC
36 mBTC
tx
Alice
Bob


36 mBTC
tx
36 mBTC
1FdtUtvK5vZxwo8jzjzid5EwGAB7paqX4n
128MZKqUsvg2kYJQ5LCVDx8Mdn8xrijzQY


36 mBTC
tx
36 mBTC
OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5EwGAB7paqX4n
OP_EQUALVERIFY
OP_CHECKSIG
OP_DUP
OP_HASH160
128MZKqUsvg2kYJQ5LCVDx8Mdn8xrijzQY
OP_EQUALVERIFY
OP_CHECKSIG

Bitcoin script

- The script runs on a **stack machine**
- It contains **simple serial** commands without loops
- It runs on **every network computer** when a **utxo** is spent
- The output of the execution is 0 or 1
- This is part of transaction validation
- If the output is 1, the input is valid and can be spent
- Otherwise the input is not valid
- And the tx is not valid

Bitcoin script

- When a tx spends a UTXO, the creator of the tx has to prove that the script outputs 1 successfully
  - i.e. that the output edge is spent fairly
- For this purpose, it supplies some **parameters** for the scriptPubKey program (the program = the encumbrance) so that **when the scriptPubKey program runs with these parameters, it outputs 1**
- The execution parameters of scriptPubKey are called **scriptSig**
- These parameters are given as part of **the new tx** which the old UTXO is connected to

Bitcoin script execution

1. We put **the scriptSig parameters** on the stack
2. We run the **commands of scriptPubKey** one by one
3. Each of these commands can **change** the stack
4. We check if the stack ends up with just a 0 or 1 in the end for **failure** or **success**

Pay-to-pubkey (p2pk)

- The simplest smart contract
- And the first ever written
- Expresses the notion that some money *rightfully belongs to* an owner
- Similar to a physical bank check
- Except it doesn’t need a central trusted third party like bank or government
- Security is provable
- Unlike “security by call-the-cops” of traditional checks
- So it can work pseudonymously

Pay-to-pubkey

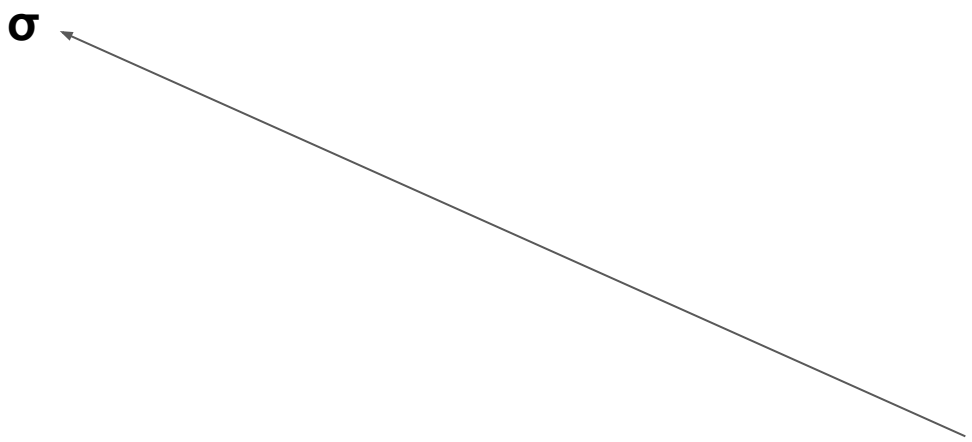
scriptPubKey:

045a5f526dfe5d5995bf95f12
OP_CHECKSIG

scriptSig:

signature σ

Pay-to-pubkey



scriptPubKey:

045a5f526dfe5d5995bf95f12
OP_CHECKSIG

scriptSig:

signature σ

Pay-to-pubkey

σ

scriptPubKey:

→ 045a5f526dfe5d5995bf95f12
OP_CHECKSIG

scriptSig:

signature σ

Pay-to-pubkey

045a5f526dfe5d5995bf95f12

σ

scriptPubKey:

045a5f526dfe5d5995bf95f12
OP_CHECKSIG

scriptSig:

signature σ

Pay-to-pubkey

045a5f526dfe5d5995bf95f12

σ

scriptPubKey:

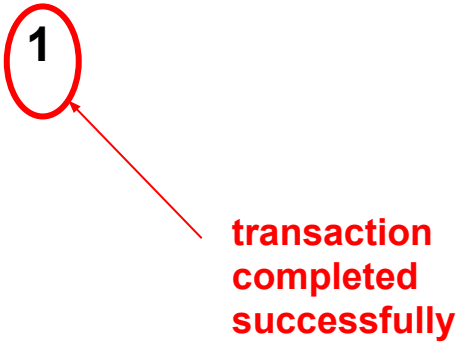
045a5f526dfe5d5995bf95f12

→ OP_CHECKSIG

scriptSig:

signature σ

Pay-to-pubkey



scriptPubKey:

045a5f526dfe5d5995bf95f12
OP_CHECKSIG

scriptSig:

signature σ

Pay-to-pubkey-hash (p2pkh)

- The way payments are done in bitcoin today
- Again a contract that ensures someone *owns money*

Pay-to-pubkey-hash

scriptPubKey:

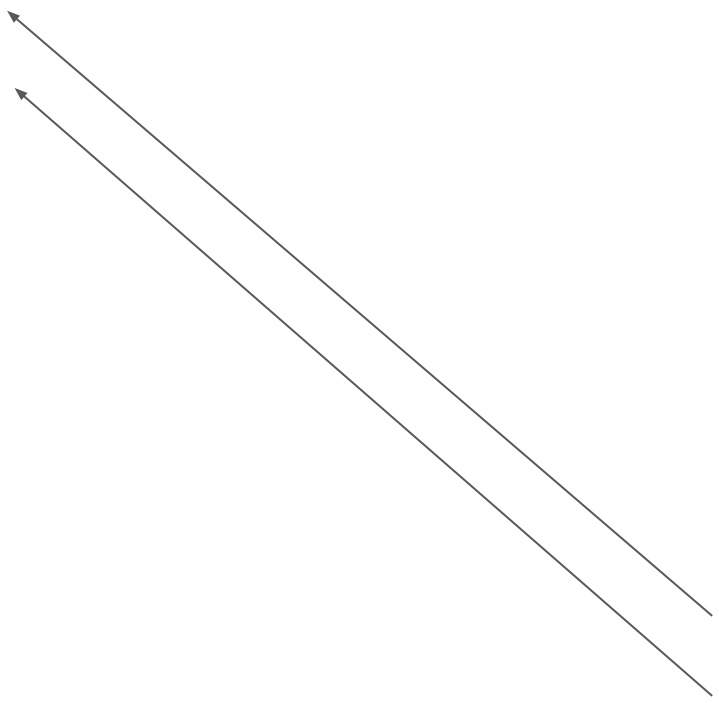
OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
signature σ

Pay-to-pubkey-hash

pubKey
signature σ



scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
signature σ

Pay-to-pubkey-hash

pubKey
signature σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
signature σ

Pay-to-pubkey-hash

pubKey
pubKey
signature σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
signature σ

Pay-to-pubkey-hash

pubKey
pubKey
signature σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
signature σ

Pay-to-pubkey-hash

H(pubKey)
pubKey
signature σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
signature σ

Pay-to-pubkey-hash

H(pubKey)
pubKey
signature σ

scriptPubKey:

OP_DUP
OP_HASH160
→ **1FdtUtvK5vZxwo8jzjzid5Ew**
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
signature σ

Pay-to-pubkey-hash

1FdtUtvK5vZxwo8jzjzid5Ew
H(pubKey)
pubKey
signature σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
signature σ

Pay-to-pubkey-hash

1FdtUtvK5vZxwo8jzjzid5Ew
H(pubKey)
pubKey
signature σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
signature σ

Pay-to-pubkey-hash

pubKey
signature σ

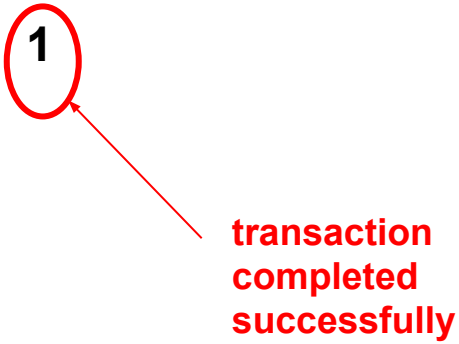
scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
→ OP_CHECKSIG

scriptSig:

pubKey
signature σ

Pay-to-pubkey-hash



scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
signature σ

Pay-to-pubkey-hash

- Most payments in bitcoin today are **Pay-to-pubkey-hash**
- Pay-to-pubkey was used at the beginning of bitcoin

A more complicated contract

OP_2DUP
OP_HASH160
BOB_HASH_CONST
OP_EQUALVERIFY
OP_HASH160
ALICE_HASH_CONST
OP_EQUALVERIFY
OP_SIZE
OP_NIP
16
OP_NUMEQUAL
OP_SWAP
OP_SIZE
OP_NIP
16
OP_NUMEQUAL
OP_NUMEQUAL
OP_IF
ALICE_PUB_KEY
OP_ELSE
BOB_PUB_KEY
OP_END_IF
OP_CHECKSIG

How can one argue about the security of these?

- These scripts are complicated and unreadable
- How can we know they do what we want?