Bitcoin scripts

University of Athens
Christos Nasikas, Dionysis Zindros

Smart Contracts

- First proposed by Nick Szabo in 1994
- A computerized transaction protocol that executes the terms of a contract
- A set of promises, specified in digital form, including protocols within which the parties perform on these promises.

Smart Contracts

- Define the rules and penalties around an agreement and automatically enforce those obligations
- Many kinds of contractual clauses may be made partially or fully self-executing, self-enforcing, or both
- Minimize the need for trusted intermediaries
- On blockchain: General purpose computation

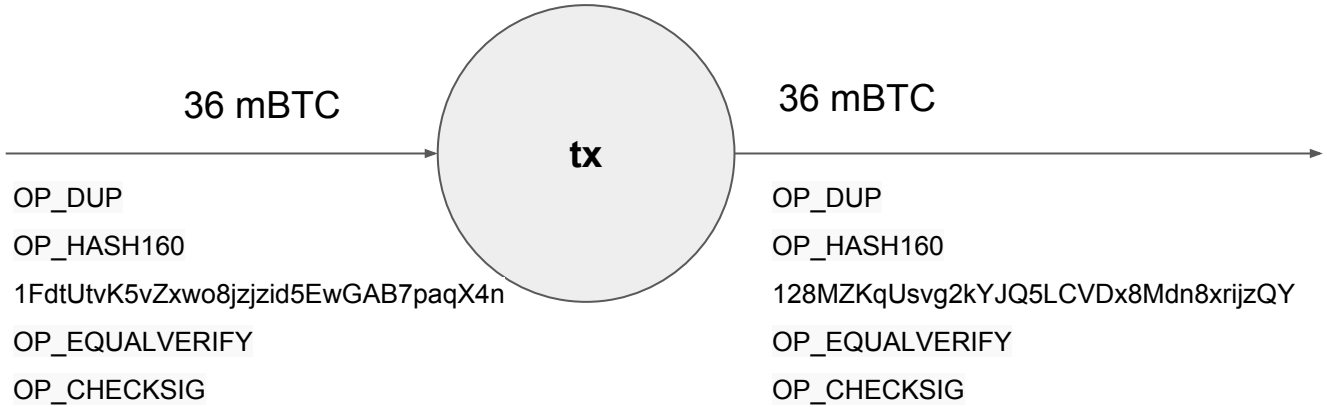
Bitcoin script: The original smart contracts

- 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
- This program is called a **scriptPubKey**
- This is the program **the verifier runs**
- This allows us to express more **complicated ownerships**

Bitcoin script



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 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**

Bitcoin script commands

- Built for Bitcoin (inspired by Forth)
- Simple, compact
- Support for cryptography
- Stack-based
- No looping (Not Turing-complete!)
- Time/memory usage bound by program size

Bitcoin script commands

256 opcodes total (15 disabled, 75 reserved)

- Arithmetic
- If/then
- Logic/data handling
- Hashes
- Signature verification
- Multi-signature verification

Bitcoin script commands

- **OP_DUP**:
  Duplicates the top element of the stack and put it on the top
- **OP_HASH160**:
  Replaces the top element of the stack x with RIPEMD160(SHA256(x))
- **OP_HASH256:**
  Replaces the top element of the stack x with SHA256(x)
- **OP_EQUAL**:
  Replaces the top two elements of the stack x and y with 1 if x==y and with 0 otherwise

Commands of Bitcoin script

- **OP_VERIFY**:
  Removes the top element of the stack. If the element is 1 the program continues. Otherwise the program fails and the execution stops
- **OP_CHECKSIG**:
  It takes from the stack a public key and a signature. It checks that the signature has been made on the new transaction and with that particular public key.
- **Constant:**
  Adds a constant at the top of the stack

Pay-to-pubkey (p2pk)

- The simplest smart contract
- And the first ever written
- Expresses the notion that some money *rightfully belongs to* an owner

Pay-to-pubkey

scriptPubKey:

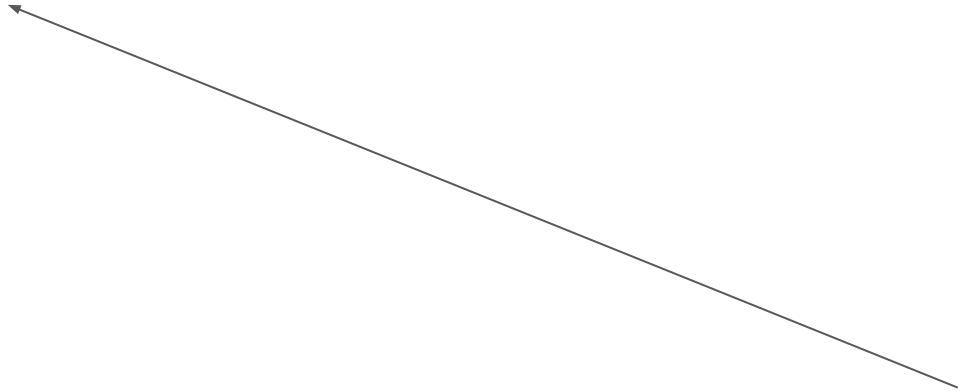
045a5f526dfe5d5995bf95f12
OP_CHECKSIG

scriptSig:

υπογραφή σ

Pay-to-pubkey

σ



scriptPubKey:

045a5f526dfe5d5995bf95f12
OP_CHECKSIG

scriptSig:

υπογραφή σ

Pay-to-pubkey

σ

scriptPubKey:

→ 045a5f526dfe5d5995bf95f12
OP_CHECKSIG

scriptSig:

υπογραφή σ

Pay-to-pubkey

045a5f526dfe5d5995bf95f12

σ

scriptPubKey:

045a5f526dfe5d5995bf95f12
OP_CHECKSIG

scriptSig:

υπογραφή σ

Pay-to-pubkey

045a5f526dfe5d5995bf95f12

σ

scriptPubKey:

045a5f526dfe5d5995bf95f12

⟶ OP_CHECKSIG

scriptSig:

υπογραφή σ

Pay-to-pubkey

The transaction completed successfully

scriptPubKey:

045a5f526dfe5d5995bf95f12
OP_CHECKSIG

scriptSig:

υπογραφή σ

Pay-to-pubkey-hash

scriptPubKey:

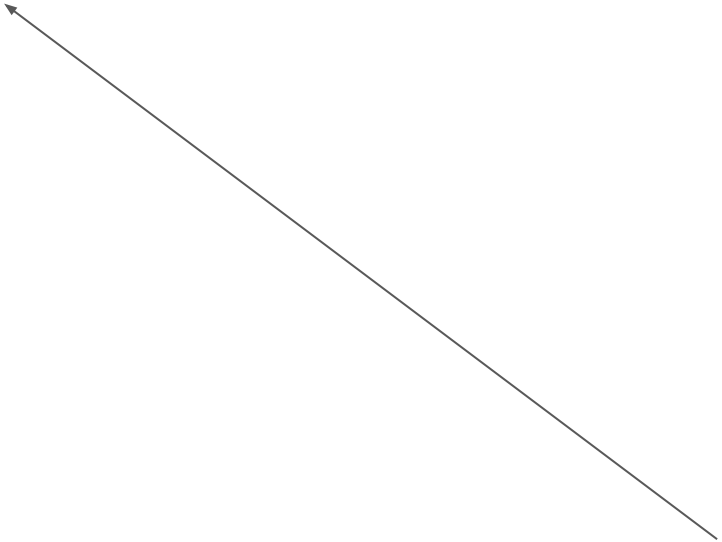
OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

pubKey
υπογραφή σ



scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

pubKey
υπογραφή σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

pubKey
pubKey
υπογραφή σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

pubKey
pubKey
υπογραφή σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

H(pubKey)
pubKey
υπογραφή σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

H(pubKey)
pubKey
υπογραφή σ

scriptPubKey:

OP_DUP
OP_HASH160
→ **1FdtUtvK5vZxwo8jzjzid5Ew**
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

1FdtUtvK5vZxwo8jzjzid5Ew
H(pubKey)
pubKey
υπογραφή σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

1FdtUtvK5vZxwo8jzjzid5Ew
H(pubKey)
pubKey
υπογραφή σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
→ OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

1

pubKey
υπογραφή σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

pubKey
υπογραφή σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

pubKey
υπογραφή σ

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
→ OP_CHECKSIG

scriptSig:

pubKey
υπογραφή σ

Pay-to-pubkey-hash

1

The transaction completed successfully

scriptPubKey:

OP_DUP
OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew
OP_EQUALVERIFY
OP_CHECKSIG

scriptSig:

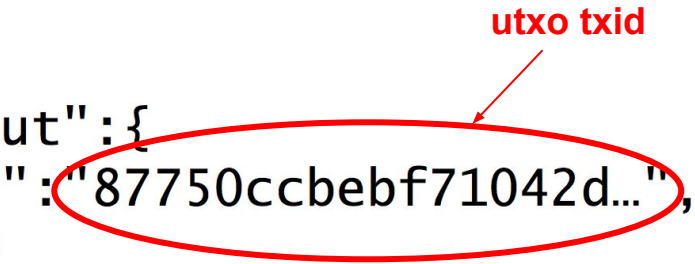
pubKey
υπογραφή σ

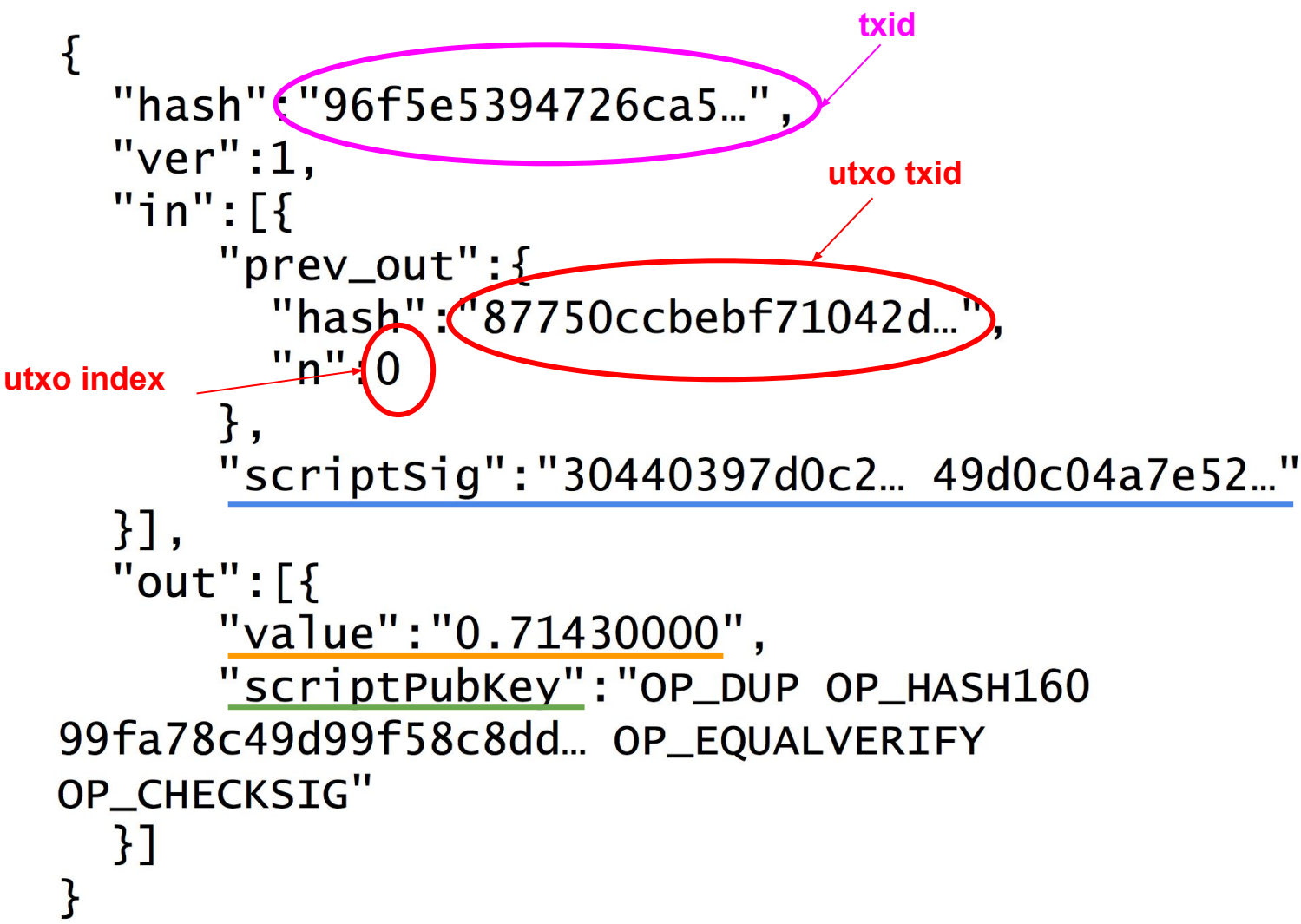
Pay-to-pubkey-hash

- Η πλειονότητα των πληρωμών στο bitcoin σήμερα είναι Pay-to-pubkey-hash
- Το Pay-to-pubkey χρησιμοποιήθηκε στην αρχή
- Πλεονέκτημα του pay-to-pubkey-hash:
- Αμύνεται στο “σπάσιμο” της EC κρυπτογραφίας
- Τα δημόσια κλειδιά δεν αποκαλύπτονται έως ότου έρθει η ώρα να ξοδέψουμε
- Όταν ξοδέψουμε, ο αντίπαλος μπορεί να προσπαθήσει να βρει το ιδιωτικό κλειδί και να κάνει double spend
- Έχει 10 λεπτά έως ότου γίνουμε confirm σε block
- Αν η EC κρυπτογραφία σπάει, δύσκολα σπάει σε 10 λεπτά

```
{
  "hash":"96f5e5394726ca5…",
  "ver":1,
  "in":[{
      "prev_out":{
        "hash":"87750ccbebf71042d…",
        "n":0
      },
      "scriptSig":"30440397d0c2… 49d0c04a7e52…"
  }],
  "out":[{
      "value":"0.71430000",
      "scriptPubKey":"OP_DUP OP_HASH160
99fa78c49d99f58c8dd… OP_EQUALVERIFY
OP_CHECKSIG"
  }]
}
```

```
{
  "hash":"96f5e5394726ca5…",
  "ver":1,
  "in":[{
      "prev_out":{
        "hash":"87750ccbebf71042d…",
        "n":0
      },
      "scriptSig":"30440397d0c2… 49d0c04a7e52…"
  }],
  "out":[{
      "value":"0.71430000",
      "scriptPubKey":"OP_DUP OP_HASH160
99fa78c49d99f58c8dd… OP_EQUALVERIFY
OP_CHECKSIG"
  }]
}
```


txid
utxo txid
utxo index
{
"hash":"96f5e5394726ca5…",
"ver":1,
"in":[{
"prev_out":{
"hash":"87750ccbebf71042d…",
"n":0
},
"scriptSig":"30440397d0c2… 49d0c04a7e52…"
}],
"out":[{
"value":"0.71430000",
"scriptPubKey":"OP_DUP OP_HASH160
99fa78c49d99f58c8dd… OP_EQUALVERIFY
OP_CHECKSIG"
}]
}

Ποιος μπορεί να ξοδέψει αυτό το script?

scriptPubKey:

OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew...
OP_EQUAL

Ποιος μπορεί να ξοδέψει αυτό το script?

scriptPubKey:

OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew...
OP_EQUAL

Λείπει το OP_CHECKSIG! **Οποιοσδήποτε** μπορεί να ξοδέψει αρκεί να ξέρει το public key που αντιστοιχεί στη διεύθυνση.

Ποιος μπορεί να ξοδέψει αυτό το script?

scriptPubKey:

OP_HASH160
1FdtUtvK5vZxwo8jzjzid5Ew...
OP_EQUAL

Οποιοσδήποτε μπορεί να κάνει **double spend** διότι το public key δημοσιεύεται ως scriptSig στην πρώτη απόπειρα ξοδέματος!

A more complicated contract

OP_2DUP
OP_HASH160
BOB_HASH_CONST
OP_EQUALVERIFY
OP_DUP
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