Bitcoin Scripts using Node.JS

Kobi Gurkan

2016-06-11

Outline

1. Overview
   - Me
   - Goals
   - Bitcoin
   - Scripting language
2. Scripts in Bitcoin
   - Simple example
   - Standard Scripts
   - Special...

Topic

1. Overview
   - Me
   - Goals
   - Bitcoin
   - Scripting language
2. Scripts in Bitcoin
   - Simple example
   - Standard Scripts
   - Special...

- Chief Scientist at Ownership - anti-counterfeiting, supply chain and certification
- Founded Shield128 - Blockchain security company
- Founded EPOK
- Math geek
- Blockchain and security enthusiast


I GUESS YOU COULD SAY
I'M PRETTY FAMILIAR
WITH THE BITCOIN

Topic

1. Overview
   - Me
   - Goals
   - Bitcoin
   - Scripting language
2. Scripts in Bitcoin
   - Simple example
   - Standard Scripts
   - Special...

- Understand the Bitcoin scripting language
- Review Bitcoin scripts, standard and non-standard
- Learn to interpret, craft and test Bitcoin scripts using Node.JS

Topic

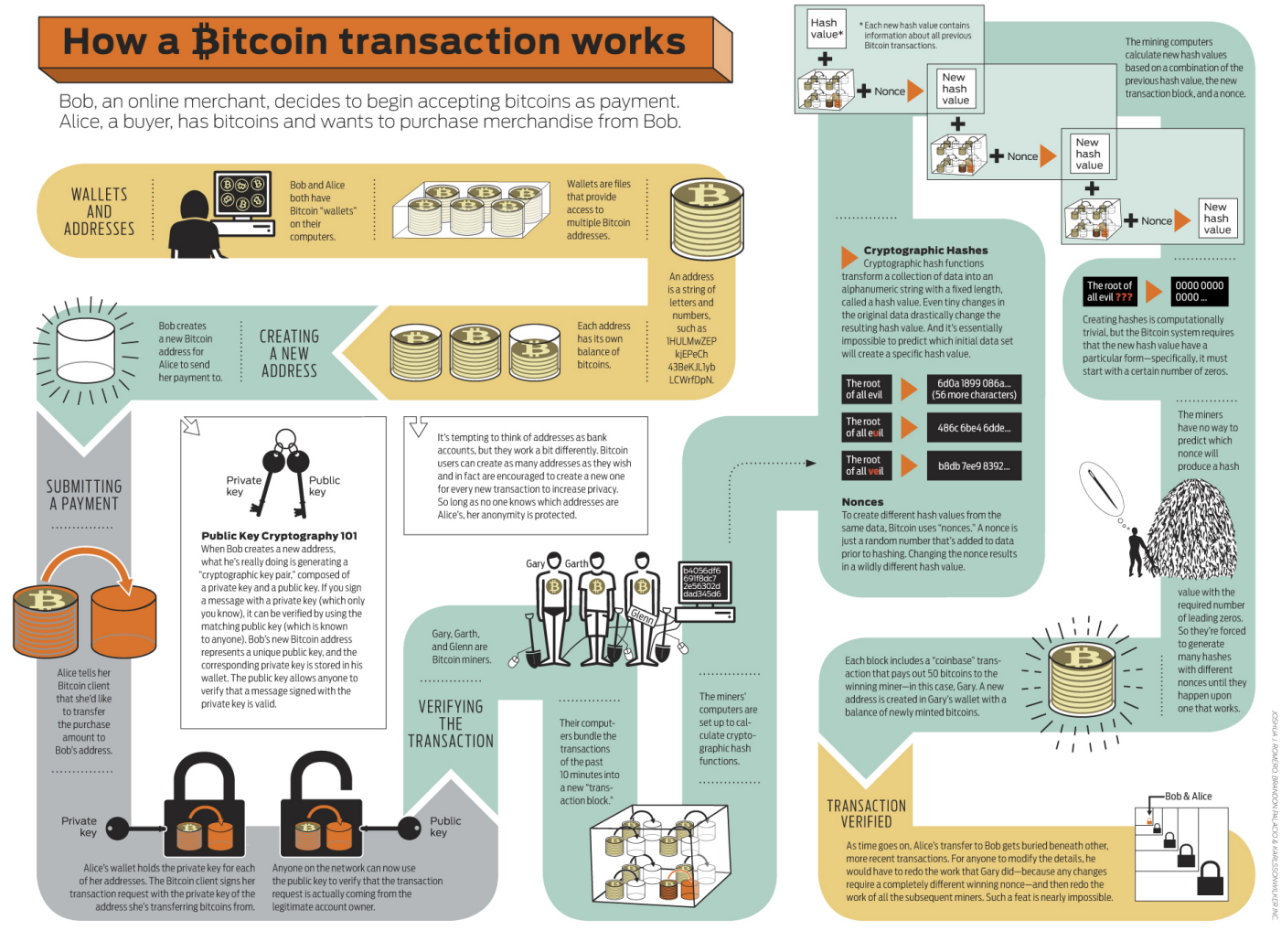
1. Overview
   - Me
   - Goals
   - Bitcoin
   - Scripting language
2. Scripts in Bitcoin
   - Simple example
   - Standard Scripts
   - Special...


BITCOIN
SAN DIEGO
KIND OF A BIG DEAL ROUND HERE
quickmeme.com

What is Bitcoin?

- Created in 2008 by Satoshi Nakamoto
- Uses elliptic curves - asymmetric encryption, public/private keys
- P2P
- Proof of work - solves Byzantine Generals Problem in decentralized way!
- For the first time, a decentralized system of money
- Ledger of transactions
- Everyone sees every transaction from the creation of the network - end of 2008

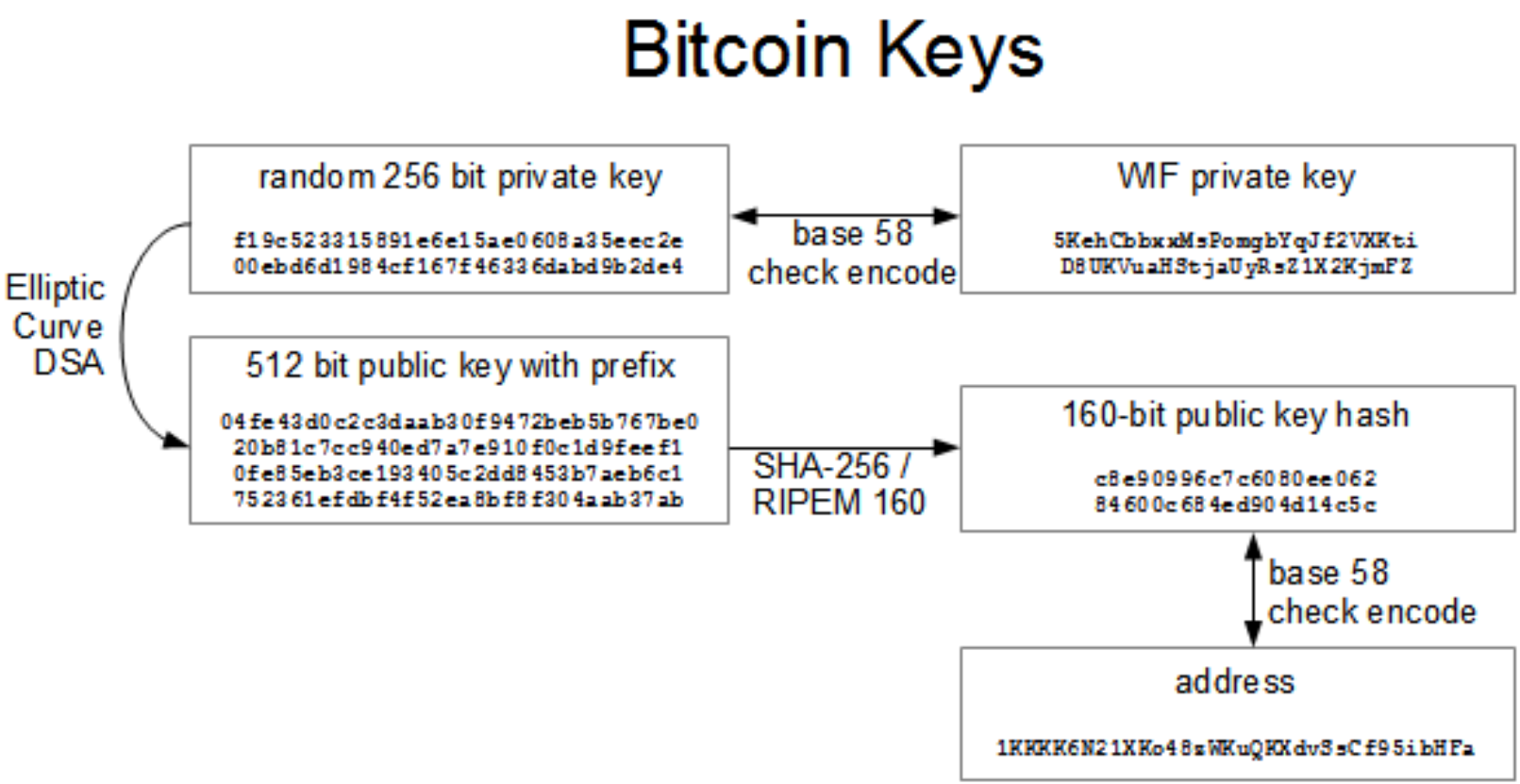
What is Bitcoin?



What is Bitcoin?

- What is a Bitcoin transaction? Transfer of value from A to B. Or that's what they want you to believe...
- But first, let's review some Bitcoin constructs.

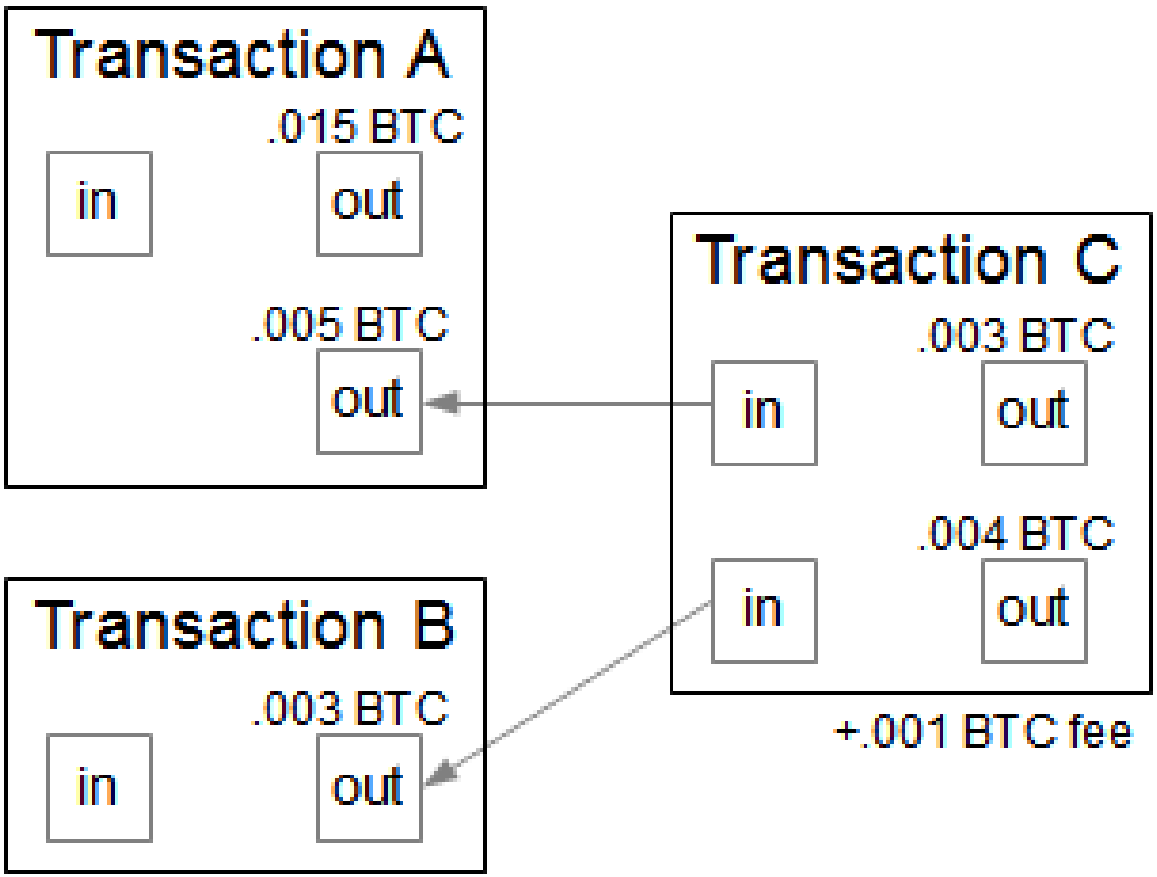
Keys


Bitcoin Keys
random 256 bit private key
f19c523315891e6e15ae0608a35eec2e
00ebd6d1984cf167f46336dabd9b2de4
base 58
check encode
WIF private key
5KehCbbxxMsPomgbYqJf2VXKti
D8UKVuaHStjaUyRsZ1X2KjmFZ
Elliptic
Curve
DSA
512 bit public key with prefix
04fe43d0c2c3daab30f9472beb5b767be0
20b81c7cc940ed7a7e910f0c1d9feef1
0fe85eb3ce193405c2dd8453b7aeb6c1
752361efdbf4f52ea8bf8f304aab37ab
SHA-256 /
RIPEM 160
160-bit public key hash
c8e90996c7c6080ee062
84600c684ed904d14c5c
base 58
check encode
address
1KKKK6N21XKo48zWKuQKXdvSsCf95ibHFa

Keys

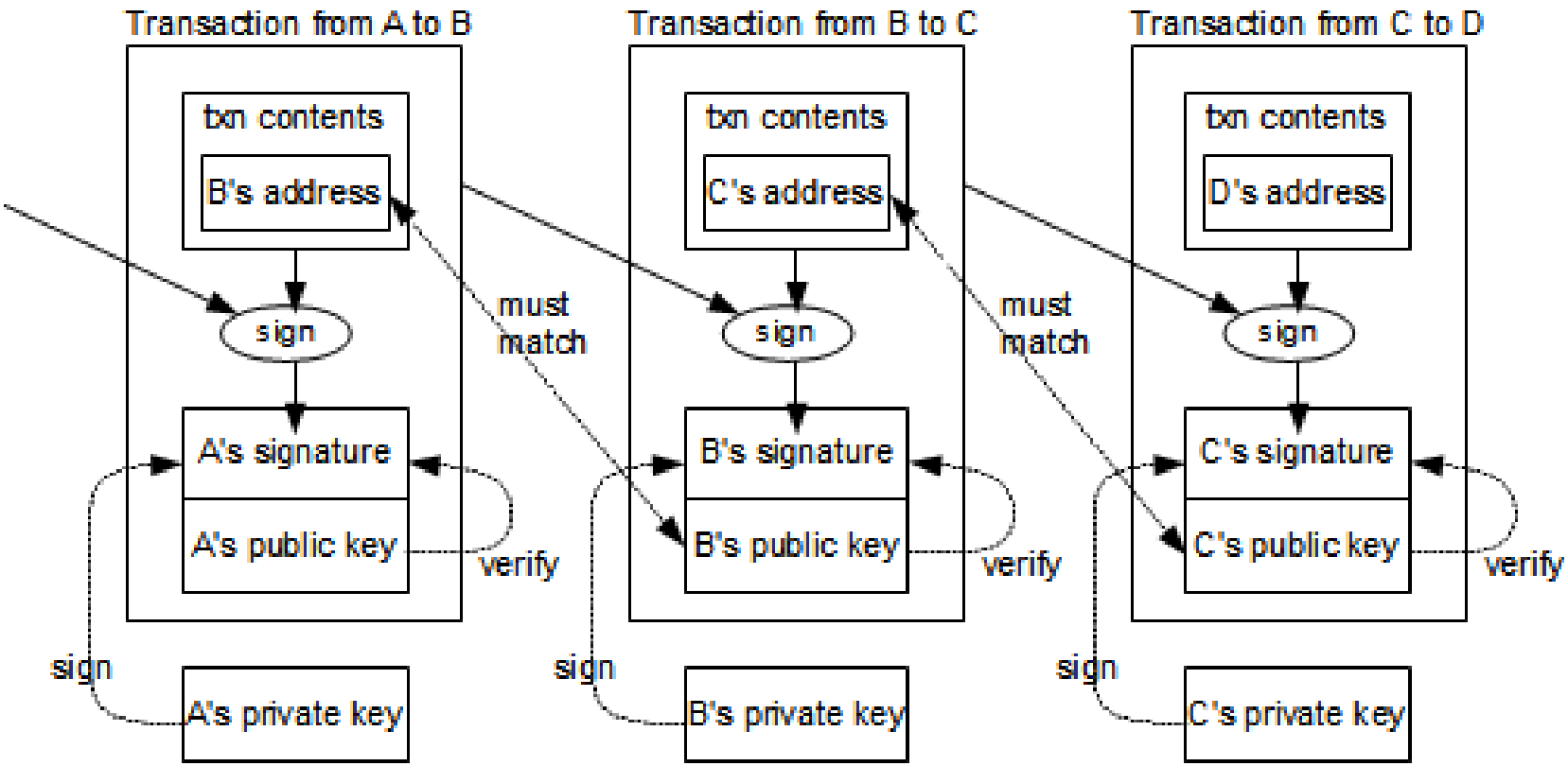
- Bitcoin private key - random 256-bit number
- Public key - derived from private key directly
- Address
  - Version = 1 byte of 0 (zero); on the test network, this is 1 byte of 111
  - Key hash = Version concatenated with RIPEMD-160(SHA-256(public key))
  - Checksum = 1st 4 bytes of SHA-256(SHA-256(Key hash))
  - Bitcoin Address = Base58Encode(Key hash concatenated with Checksum)
- You prove ownership of an address by signing using your private key - ECDSA

Transactions



Transactions

- Every transaction has inputs and outputs
- Output - credit 100 bitcoins to someone under conditions - scriptPubKey
- Input - spend 100 bitcoins from a previous output by fulfilling the condition - scriptSig
- That is, every input spends a previous output
- Where are the initial Bitcoins to spend from? As we saw in the diagram, mining
- Because the ledger is public - i.e. you can see the move of Bitcoins from the miner up to the moment you received them



Source: http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html


ONE DOES NOT SIMPLY
UNDERSTAND WHAT IS BITCOIN
imgflip.com

Topic

1. Overview
   - Me
   - Goals
   - Bitcoin
   - Scripting language
2. Scripts in Bitcoin
   - Simple example
   - Standard Scripts
   - Special. . .

So in reality, you can do much more.

- 117 Opcodes (some disabled, some placeholders)
- 1 byte
- Simple language
- No loops (=> no infinite loops)
- Not turing-complete, intentionally

- Stack starts empty
- scriptSig is evaluated
- scriptPubKey is evaluated
- If top of stack is non-zero then the transaction is valid

Topic

1. Overview
   - Me
   - Goals
   - Bitcoin
   - Scripting language
2. Scripts in Bitcoin
   - Simple example
   - Standard Scripts
   - Special...

Stack	scriptSig	scriptPubKey
	5 2	OP_SUB

Stack	scriptSig	scriptPubKey
5	2	OP_SUB

Stack	scriptSig	scriptPubKey
5 2		OP_SUB

Stack	scriptSig	scriptPubKey
3		

- This is actually valid, because the top of the stack is non-zero.
- We could also check the result to verify that we know math.

Stack	scriptSig	scriptPubKey
	5 2	OP_SUB 3 OP_EQUAL

Stack	scriptSig	scriptPubKey
5	2	OP_SUB 3 OP_EQUAL

Stack	scriptSig	scriptPubKey
5 2		OP_SUB 3 OP_EQUAL

Stack	scriptSig	scriptPubKey
3		3 OP_EQUAL

Stack	scriptSig	scriptPubKey
3 3		OP_EQUAL

Stack	scriptSig	scriptPubKey
1 (True)		

- Transaction valid too, and we indeed know to subtract!

- This was easy, but might be hard to do by hand for more complex scripts.
- Luckily, BitPay did an awesome job by developing bitcore.

```
> npm install bitcore
> node
```

```
var bitcore = require('bitcore');
var interpreter = bitcore.Script.Interpreter();
interpreter.set({script: 'OP_2 OP_1 OP_SUB'})
interpreter.evaluate()
```

- Let's see an interactive step session!
- `https://asciinema.org/a/bkznx7286g09jleitqdu641xm`

- OK, but on to more honesty. The transaction was not valid.
- There are only a few standard scripts.
- But don't despair - arbitrary scripts are still allowed. I'll explain later.

Topic

1. Overview
   - Me
   - Goals
   - Bitcoin
   - Scripting language
2. Scripts in Bitcoin
   - Simple example
   - Standard Scripts
   - Special. . .

P2PK

scriptSig	scriptPubKey
<sig>	<pubkey> OP_CHECKSIG

- `https://asciinema.org/a/82dplmciis241l3slxdee5h09`
- Not widely used anymore because public key is... public.

P2PKH

scriptSig	scriptPubKey
<sig> <pubkey>	OP_DUP OP_HASH160 OP_EQUALVERIFY OP_CHECKSIG

- Most used transaction - credit 100 Bitcoins to a hash of a public key.
- Prove you own the address by signing the entire transaction - OP_CHECKSIG.

Multisig

scriptSig	scriptPubKey
OP_0	<m>
<A sig>	<A pubkey>
[B sig]	<B pubkey>
[C sig...]	<C pubkey...>
	<n>
	OP_CHECKMULTISIG

- Divide ownership of Bitcoins
- Not widely used. Multisig is implemented as script hash...

Null Data

scriptSig	scriptPubKey
	OP_RETURN <0 to 80 bytes of data>

- https://asciinema.org/a/0llb5rtakanx3to35lvcdul39
- Can put arbitrary data on Blockchain!
- https://twitter.com/op_return_ack/status/705968000688455684
- Open Assets (colored coins) - https://github.com/OpenAssets/open-assets-protocol/blob/master/specification.mediawiki

P2SH

scriptSig	scriptPubKey
<additional opcodes...> <redeemScript>	OP_HASH160 <Hash160(redeemScript)> OP_EQUAL

- Supports arbitrary scripts!

P2SH-based 2-of-3 Multisig

scriptSig	redeemScript	scriptPubKey
OP_0 <A sig> [C sig] <redeemScript>	<OP_2> <A pubkey> <B pubkey> <C pubkey...> <OP_3> OP_CHECKMULTISIG	OP_HASH160 <PubKeyHash> <Hash160(redeemScript)> OP_EQUAL

- Widely used form of Multisig

Topic

1. Overview
   - Me
   - Goals
   - Bitcoin
   - Scripting language
2. Scripts in Bitcoin
   - Simple example
   - Standard Scripts
   - Special...

Transaction Puzzle

scriptSig	scriptPubKey
<data>	OP_SHA256 <given_hash> OP_EQUAL

- `https://live.blockcypher.com/btc/tx/9969603dca74d14d29d1d5f56b94c7872551607f8c2d6837ab9715c60721`
- `https://asciinema.org/a/aj92zdz3m61kjkqfg0nghxiq7`
- Or as P2SH: `https://asciinema.org/a/5sumtxdthpgza5f8z8s1t8jzb`
- But this is insecure because anyone can see the solution, since everything is public...

Secure Transaction Puzzle

scriptSig	scriptPubKey
<sig>	OP_SHA256
<pubkey>	<given_hash>
<data>	OP_EQUALVERIFY
	OP_DUP
	OP_HASH160
	<pubkeyHash>
	OP_EQUALVERIFY
	OP_CHECKSIG

- This is non-standard, of course. But, we can put this script as a P2SH script.
- `https://asciinema.org/a/bvk8nockh7dq715qp8avhiw8h`

Zero Knowledge Contingent Payment

scriptSig	scriptPubKey
	OP_SHA256 <Y> OP_EQUAL OP_IF <Seller pubkey> OP_ELSE <block_height + 100> OP_CHECKLOCKTIMEVERIFY OP_DROP <Buyer pubkey> OP_ENDIF OP_CHECKSIG

- `https://bitcoincore.org/en/2016/02/26/zero-knowledge-contingent-payments-announcement/`
- Deposit - `https://live.blockcypher.com/btc/tx/8e5df5f792ac4e98cca87f10aba7947337684a5a0a7333ab897fb9c9d616ba9e`
- Spend - `https://live.blockcypher.com/btc/tx/200554139d1e3fe6e499f6ffb0b6e01e706eb8c897293a7f6a26d25e39623fae/`

OP_CHECKLOCKTIMEVERIFY

scriptSig	scriptPubKey
	OP_IF <now + 3 months> OP_CHECKLOCKTIMEVERIFY OP_DROP <Lenny's pubkey> OP_CHECKSIGVERIFY 1 OP_ELSE 2 OP_ENDIF <Alice's pubkey> <Bob's pubkey> 2 OP_CHECKMULTISIG

- Allows to lock Bitcoins until a specific date!
- `https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki`

- At any time:

scriptSig	scriptPubKey
0 <Alice's signature> <Bob's signature> 0	

- After 3 months:

scriptSig	scriptPubKey
0 <Alice/Bob's signature> <Lenny's signature> 1	

Puzzle time!

```
var bitcore = require('bitcore');
var request = require('request');

var utxo = {
    'txId' : '0c5d827a24b0822abbff73f1adbfd2be7efaf4e368b4baac7e280865eb13f497',
    'outputIndex' : 1,
    'script' : 'a91431f2ae5b333c56a4f01df6209382ec8f892e4f3687',
    'satoshis' : 1000000
};


//you can see it here:
//https://live.blockcypher.com/btc/tx/0c5d827a24b0822abbff73f1adbfd2be7efaf4e368b4baac7e280865eb13f497

var address = 'YOUR_ADDRESS';
var tx = new bitcore.Transaction().from(utxo).to(address, 900000);
var redeemScript = bitcore.Script.fromASM('\
    OP_SHA256 \
    1bc273366856a5fb0bfd0611f57b7d4ae8e709dbe30fa207729f74716fd4e877\
    OP_EQUAL');
//this is your task! what is the total number of Bitcoins that will
//ever be created, in hex format (don't forget the leading zero!)?
var solutionScript = '';
var scriptSig = bitcore.Script.fromASM(solutionScript + ' ' + redeemScript);
tx.inputs[0].setScript(scriptSig);
var rawTx = tx.toString('hex');
```

```
var pushTx = {
    tx: rawTx
};

//for simplicity, we'll use the awesome BlockCypher API.
//You could also submit this raw transaction using Bitcoin Core.
request({
    url: 'https://api.blockcypher.com/v1/btc/main/txs/push',
    method: 'POST',
    json: true,
    body: pushTx
}, function (err, response, body) {
    if (err) {
        return console.log('Error: ', err);
    }
    console.log(body);
});
```

Questions?

- kobigurk@gmail.com
- kobigurk.com
- @kobigurk

We are hiring!