Digital Signatures And Bitcoin

Nicolas T. Courtois

- **U**niversity **C**ollege **L**ondon, UK

Introducing Bitcoin

Bitcoin In A Nutshell

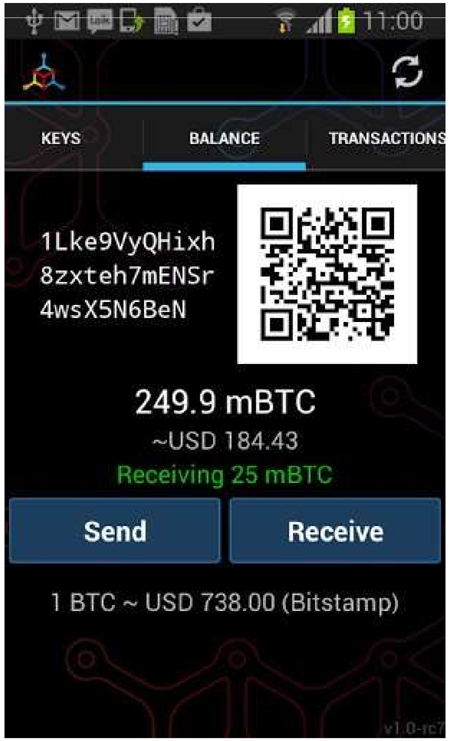
- bitocoins are cryptographic tokens
  - stored by people on their PCs or mobile phones
- ownership is achieved through digital signatures:
  - you have a certain cryptographic key, you have the money.
  - publicly verifiable, only one entity can sign

- consensus-driven, a distributed system which has no central authority
  - but I will not claim it is decentralized, this is simply not true!
  - a major innovation is that financial transactions CAN be executed and policed without trusted authorities. Bitcoin is a sort of financial cooperative or a distributed business.
- based on self-interest:
  - a group of some 100 K people called bitcoin miners own the bitcoin “infrastructure” which has costed about 0.5-1 billion dollars (estimation)
  - they make money from newly created bitcoins and fees
  - at the same time they approve and check the transactions.
  - a distributed electronic notary system

Two Key Concepts

- initially money are attributed through **Proof Of Work (POW)** to one public key A
  - to earn bitcoins one has to “work” (hashing) and consume energy (pay for electricity)
  - Now in order to cheat one needs to work even much more (be more powerful than the whole network), more precisely:
- money transfer from public key A to public key B:
  - **like signing a transfer in front of one notary which confirms the signature**,
  - multiple confirmations: another notary will re-confirm it, then another, etc…
  - we do NOT need to assume that ALL these notaries are honest.
    - at the end it becomes too costly to cheat

In Practice



1Lke9VyQHixh
8zxteh7mENSr
4wsX5N6BeN

249.9 mBTC

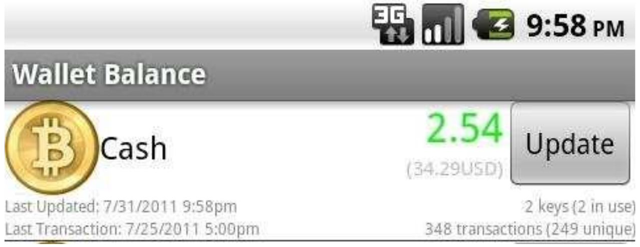
~USD 184.43

Receiving 25 mBTC

1 BTC ~ USD 738.00 (Bitstamp)

Wallets

- Wallet: file which stores your “money".
- A Bitcoin client App is also called a wallet



Wallet Balance

Cash 2.54 (34.29USD)

Last Updated: 7/31/2011 9:58pm 2 keys (2 in use)

Last Transaction: 7/25/2011 5:00pm 348 transactions (249 unique)

- Four types:
  1. Decent PC, full P2P node, stores ALL history - 14 Gbytes, trusts no one.
  2. Mobile apps: trust and rely on servers for DB and authenticity; but stores money locally.
  3. Cloud apps: all is stored in the cloud!
  4. Offline systems: protect your assets from cybercriminals

Digital Currency

Digital Currency

1. Sth. that we know… String of Bits.

 + additional layers of security:

2. Sth that we can do (capability): BETTER.
   - can be used many times without loss of confidentiality…
   - in bitcoin bank account = a certain private ECDSA key…

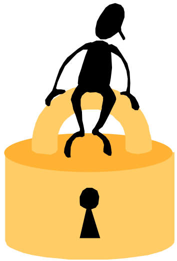
=>PK-based Currency,
an important modern application of Digital Signatures!

Main Problem:

This capability can be “spent twice”.

Avoiding this “Double Spending” is the main problem when designing a digital currency system.

Crypto



**Crypto Citations

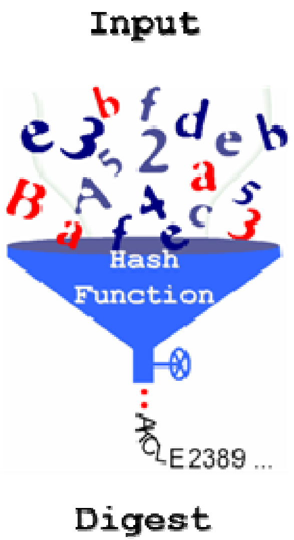
About Bitcoin:

- The accuracy of past transactions is guaranteed by cryptography, which is a special type of mathematics ☺

**Crypto Misconceptions

THIS IS WRONG:

- SHA-256 is a cipher and provides confidentiality.
  - – Not it is a hash function and provides integrity of everything [hard to modify./cheat]



- "Bitcoins are encrypted": WRONG
  - – ONLY if you encrypt your wallet, not everybody does.
  - – Also can use SSL in P2P connections…
    - communications are encrypted if you use TOR

Block Chain

Append-Only Logs

One well-known method to implement money
[pre-dates bitcoin according to George Danezis slides]:

A high-integrity, high-authenticity ”append only log”.

Sufficient to implement money in theory.

- Start by marking who has what money.
- Enter a log entry for each transfer.

Solutions differ in the method to get this ”append only log”

Bitcoin Mining

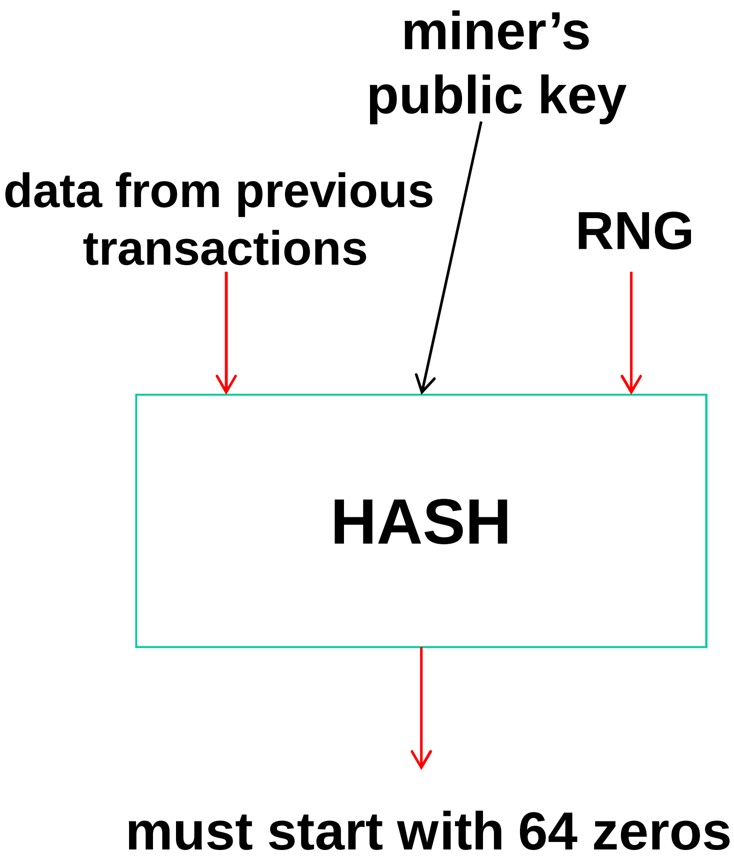
- Minting: creation of new currency.
- Confirmation+re-confirmation of older transactions

Random Oracle – like mechanism

Ownership:

- "policed by majority of miners":
- only the owner can transfer

[a part of] 25 BTC produced.



Block Chain

Def:

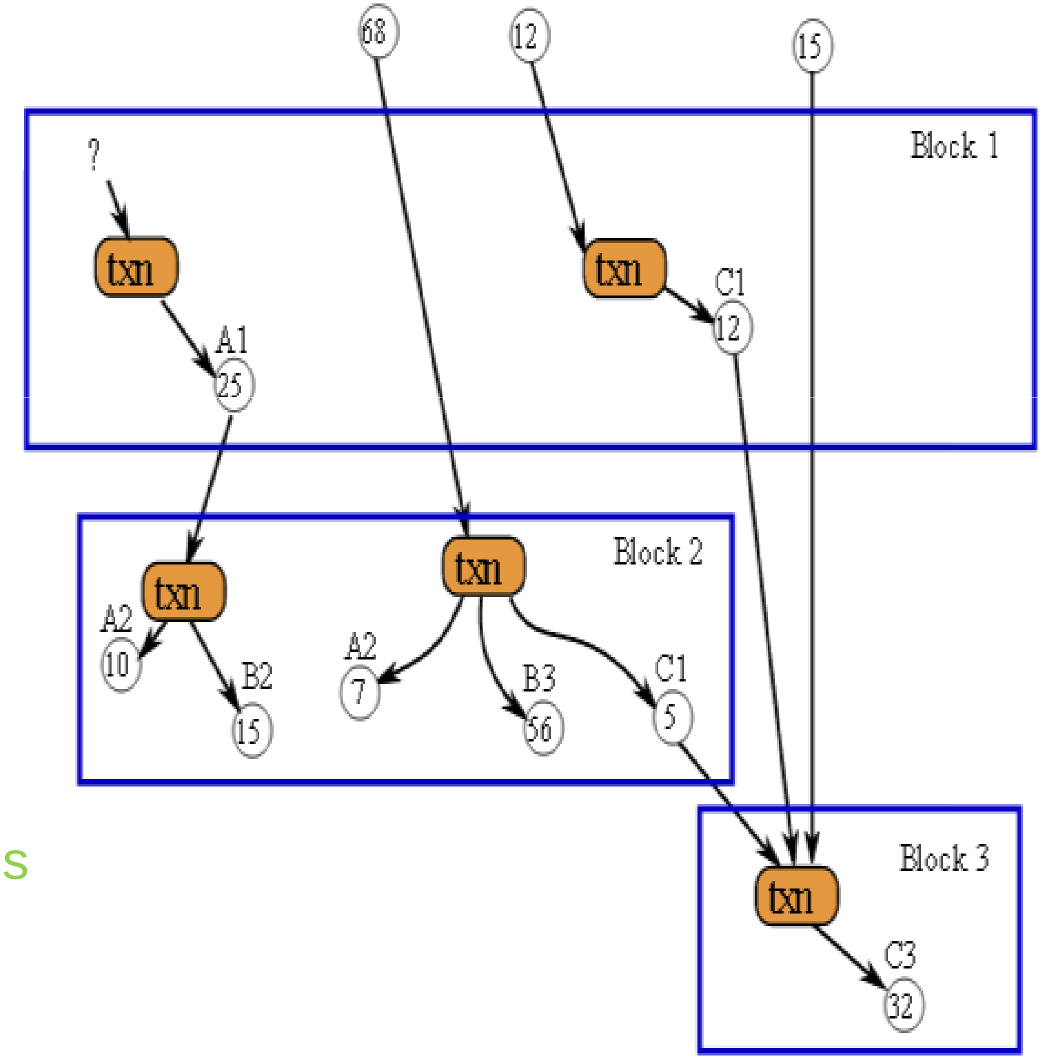
A transaction database
shared by everyone.

Also a ledger.

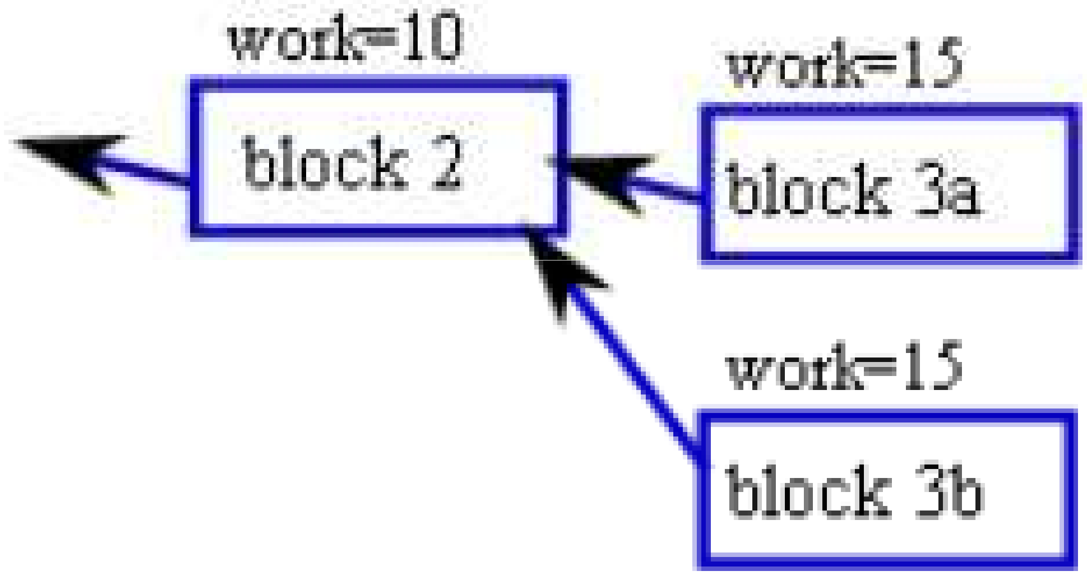
Every transaction
since ever is public.

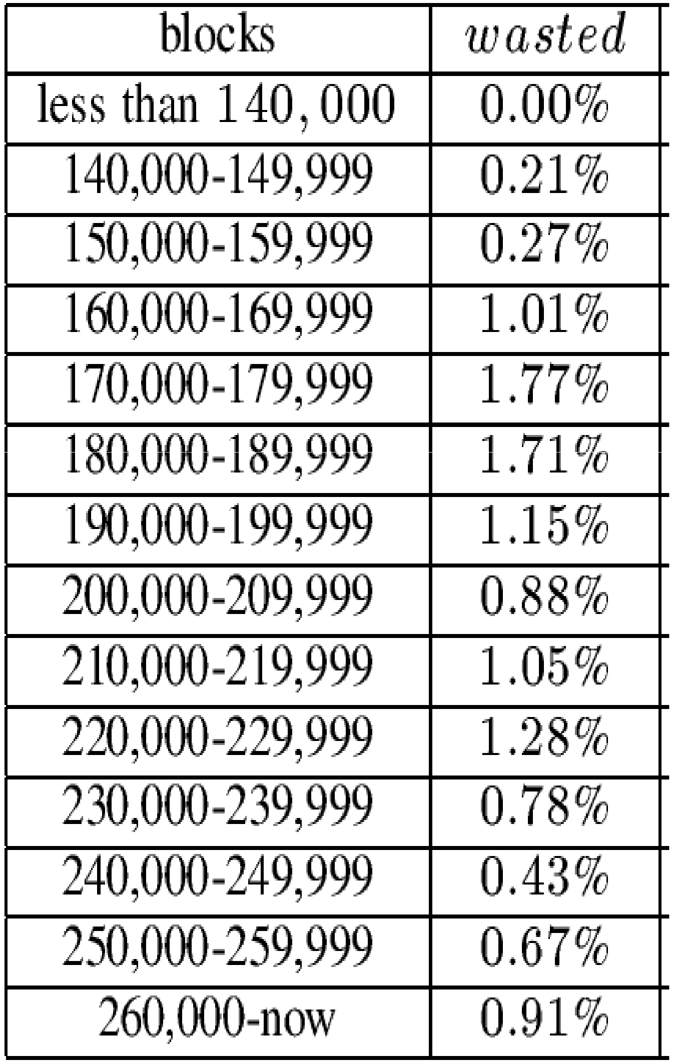
Each bitcoin “piece” is
a union of things uniquely traced
to their origin in time

(cf. same as for several banknotes
due to SN)



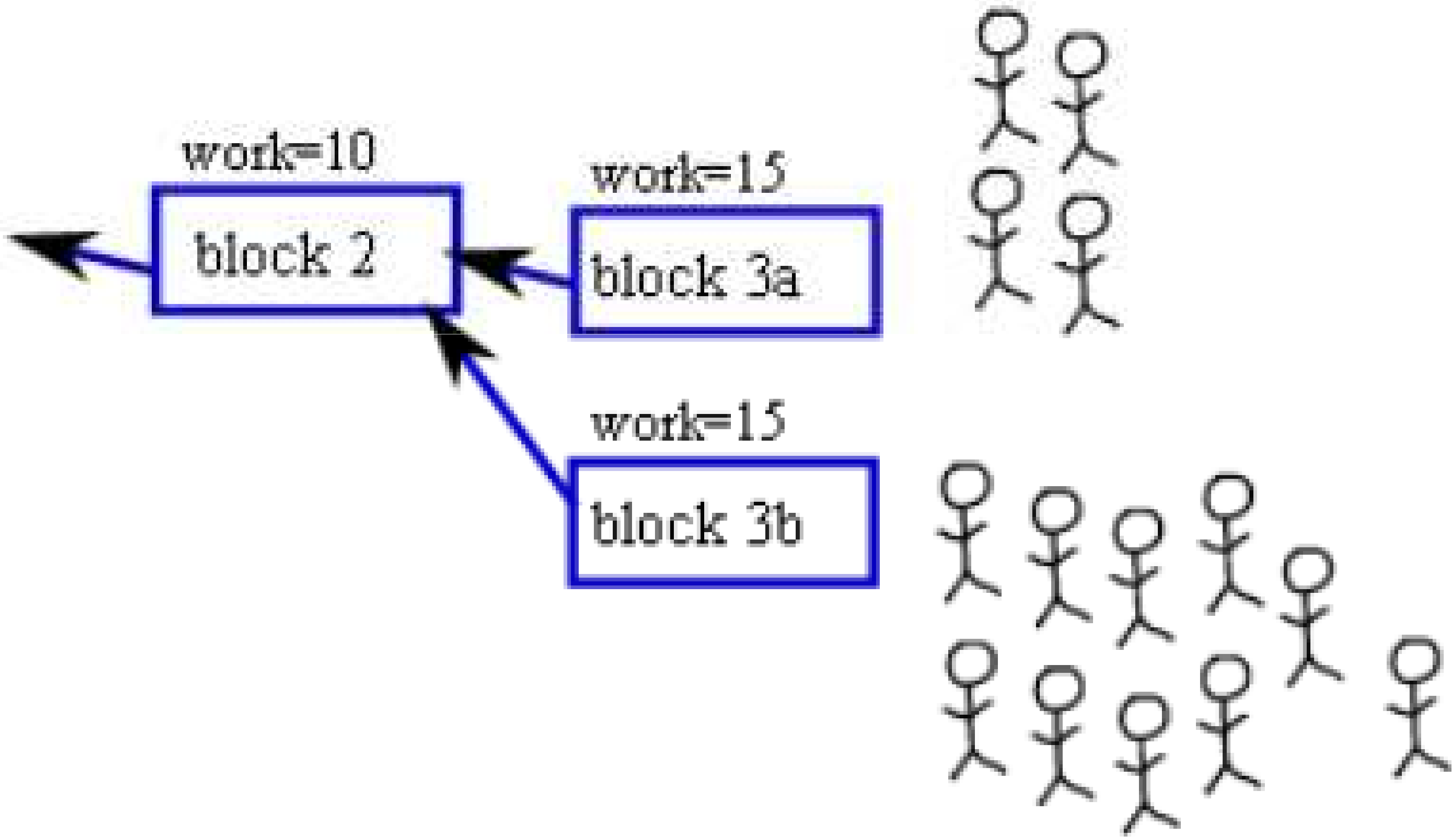
Fork – Hard To Avoid, 1% of the time





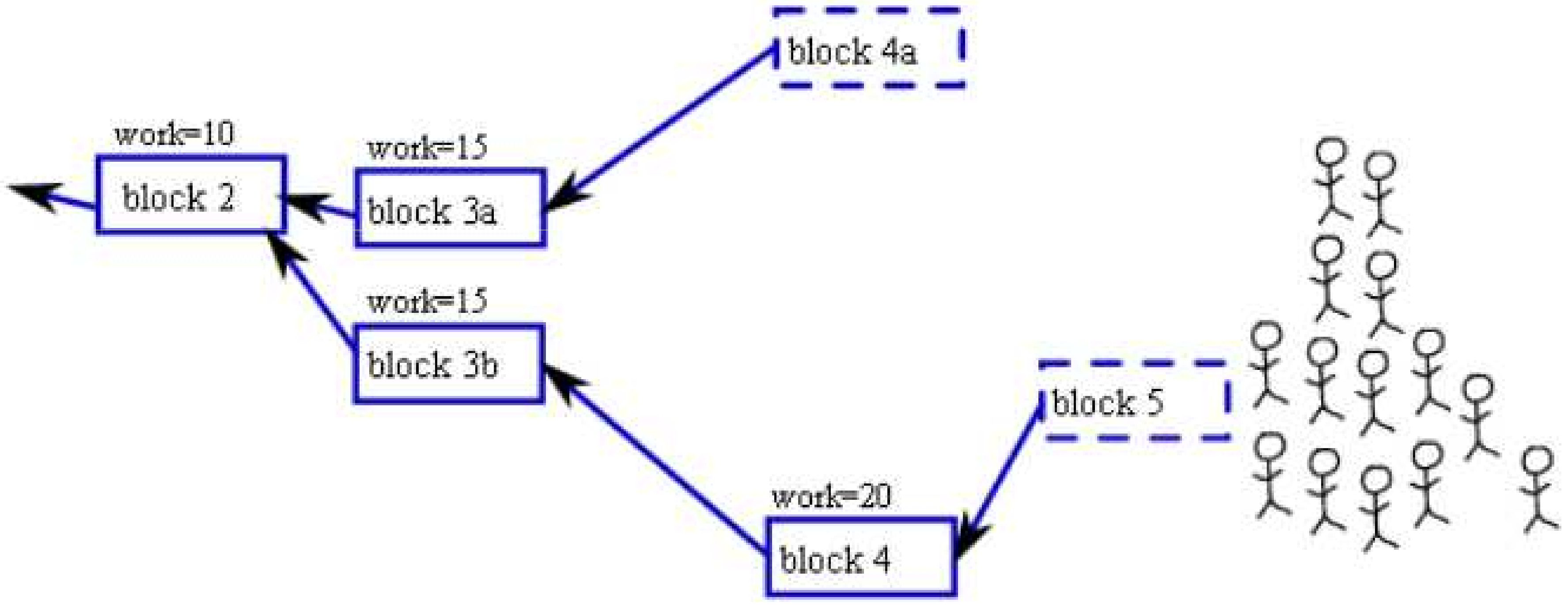
blocks	*wasted*
less than 140,000	0.00%
140,000-149,999	0.21%
150,000-159,999	0.27%
160,000-169,999	1.01%
170,000-179,999	1.77%
180,000-189,999	1.71%
190,000-199,999	1.15%
200,000-209,999	0.88%
210,000-219,999	1.05%
220,000-229,999	1.28%
230,000-239,999	0.78%
240,000-249,999	0.43%
250,000-259,999	0.67%
260,000-now	0.91%

Fork – Miners Mine On Both Branches

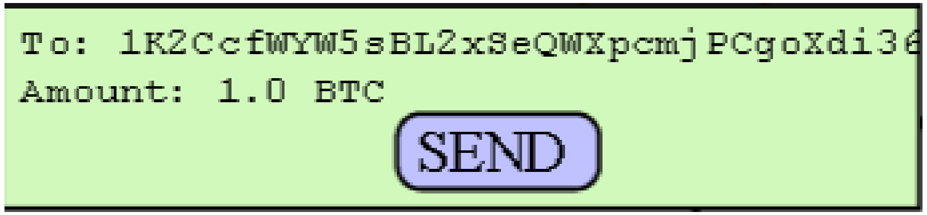


Longest Chain Rule – Clear Winner

“1 ASIC 1 vote”



Bitcoin Address



To: 1K2CcfWYW5sBL2xSeQWXpcmjPCgoXdi36

Amount: 1.0 BTC

Ledger-Based Currency

A “Bitcoin Address” = a sort of equivalent of a bank account.

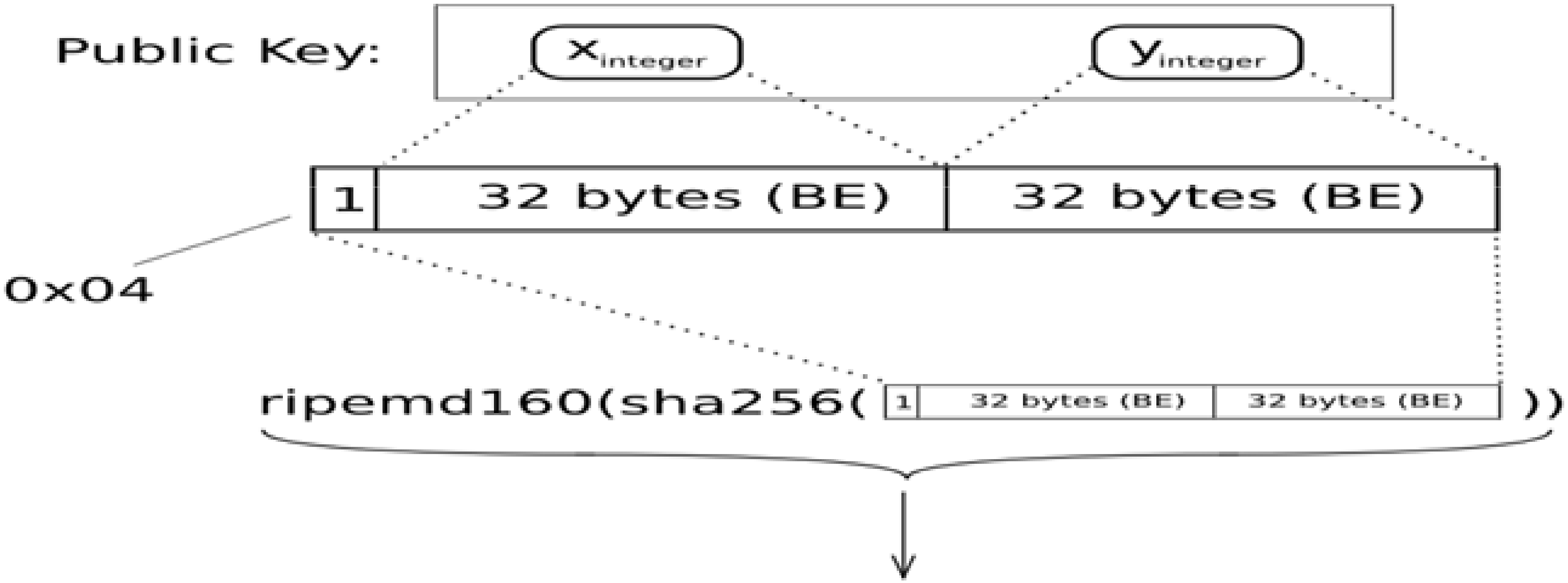
Three formats.

- First format like full Pkey 2*32 byte points, redundant!
  "scriptPubKey":"04a39b9e4fbd213ef24bb9be69de4a118dd0644082e47c01fd9159d38637b83fbcdc115a5d6e970586a012d1cfe3e3a8b1a3d04e763bdc5a071c0e827c0bd834a5 OP_CHECKSIG“
- Hash it on 160 bits, conceals the PK key! (NSA: attacks possible!).
  - e.g. 0568015a9facccfd09d70d409b6fc1a5546cecc6
- Recode with checksum on 1+20+4 bytes checksum, 160+32 bits,
  - Base58: 1VayNert3x1KzbpzMGt2qdqrAThiRovi8 27-34 chars

PK itself remains confidential until some part is spent.

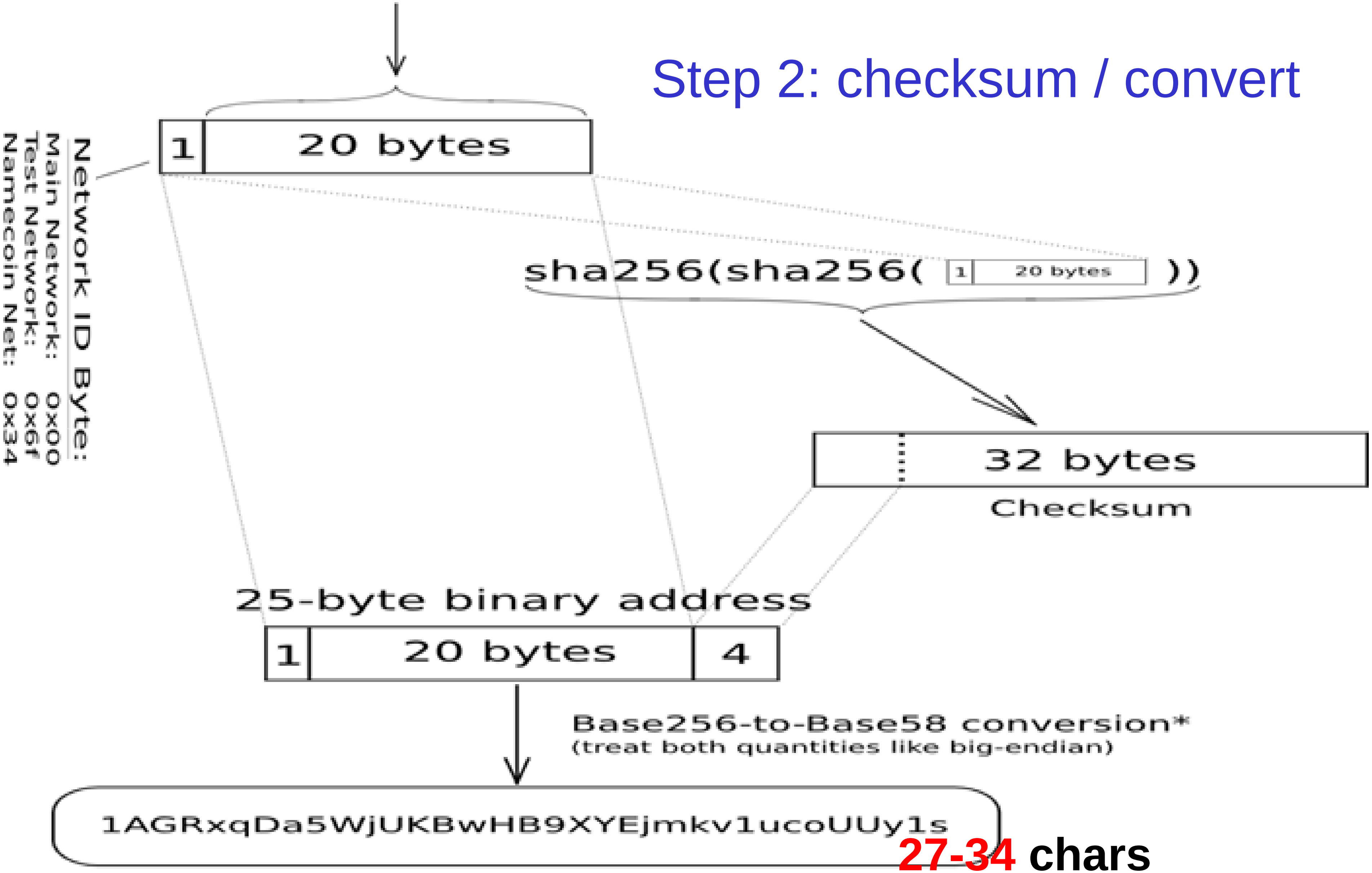
SK = private key is always kept private, allows transfer of funds.

Step 1: Hash



40 chars (nibbles)

Step 2: checksum / convert

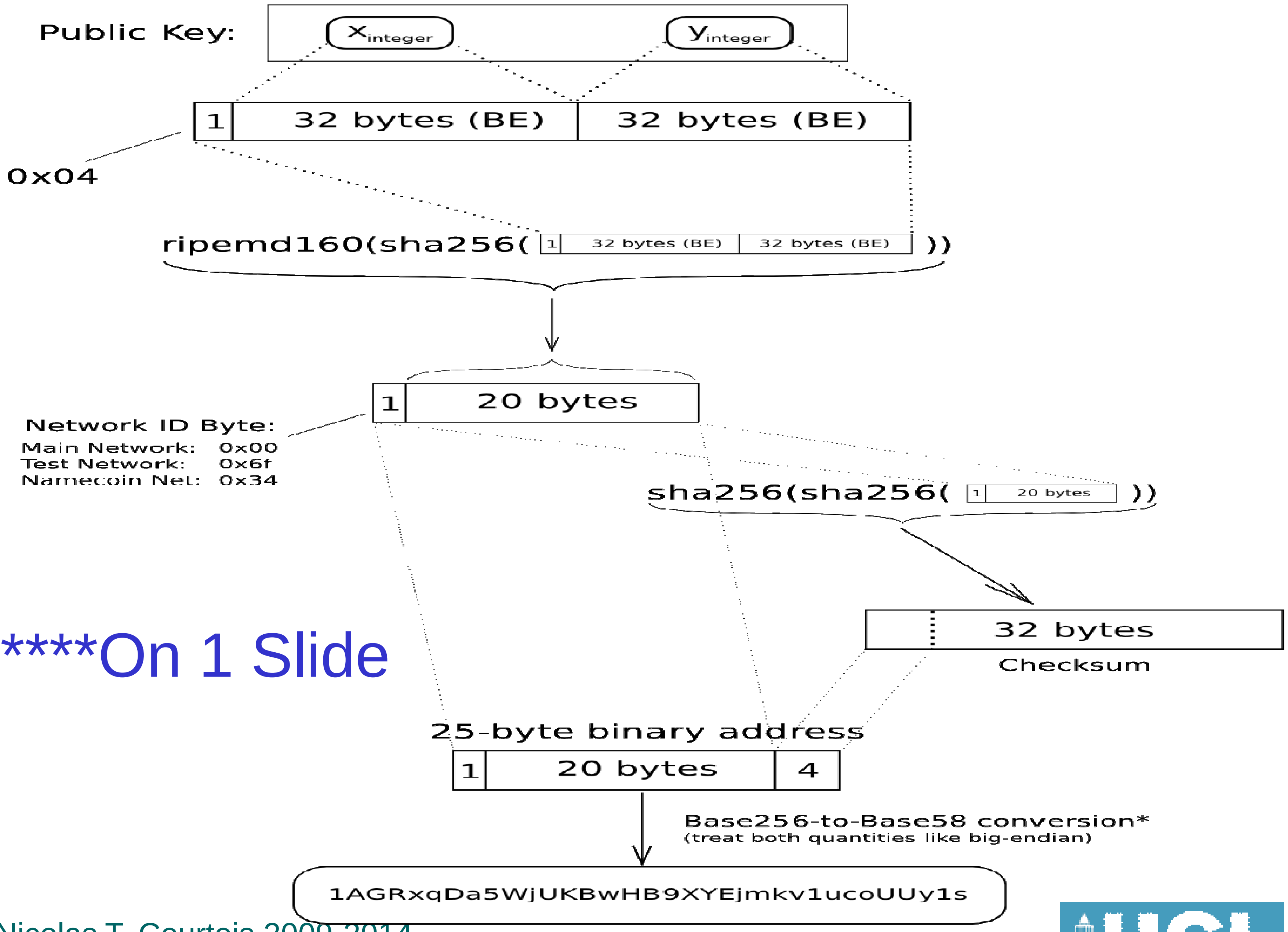

Network ID Byte:
Main Network: 0x00
Test Network: 0x6f
Namecoin Net: 0x34
1
20 bytes
sha256(sha256(1 20 bytes))
32 bytes
Checksum
25-byte binary address
1
20 bytes
4
Base256-to-Base58 conversion*
(treat both quantities like big-endian)
1AGRxqDa5WjUKBwHB9XYEjmkv1ucoUUy1s

27-34 chars

Base_58 O0I1

Elliptic-Curve Public Key to BTC Address conversion

Crypto Currencies and Digital Signatures



*****On 1 Slide

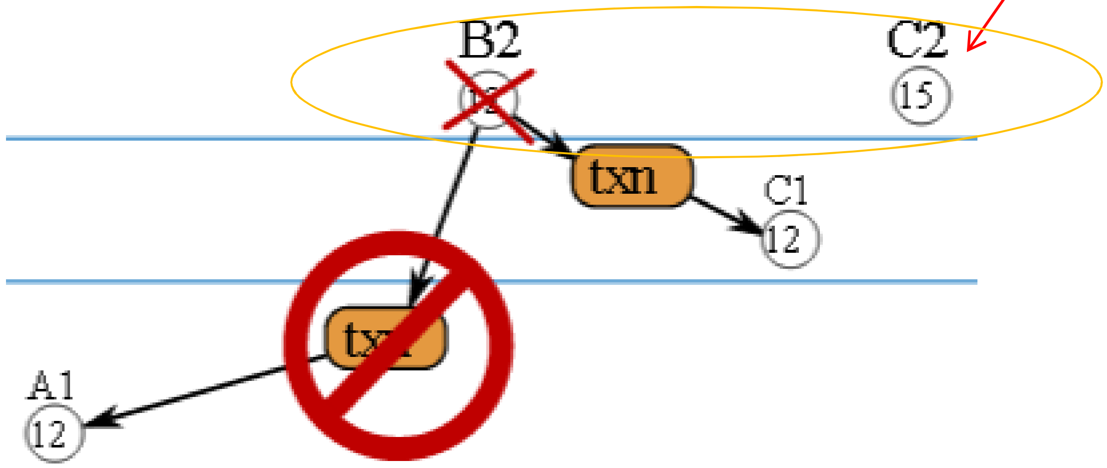
Bitcoin Ownership

Amounts of money are attributed to public keys.

Owner of a certain “Attribution to PK” can at any moment transfer it to some other PK (== another address).

Destructive, cannot spend twice:

not spent



Multi-Signature Addresses

Special Type of Addresses

Bitcoin can require simultaneously several private keys, in order to transfer the money.

The keys can be stored on different devices (highly secure).

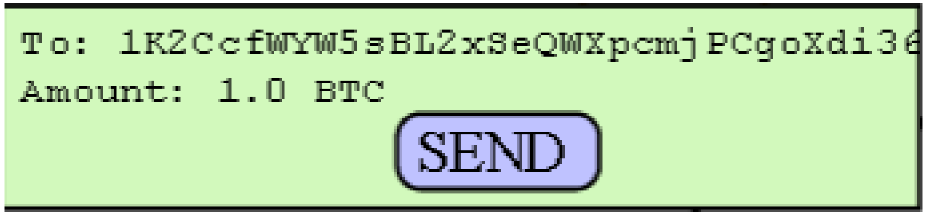
2 out of 3 are also already implemented in bitcoin.

(1 device could be absent, money can still be used).

Very cool, solves the problem of insecure devices…

Except if the attacker can break into many devices…

Bitcoin Circulation



To: 1K2CcfWYW5sBL2xSeQWXpcmjPCgoXdi36

Amount: 1.0 BTC

Bitcoin Myths (not true)

“Transactions are irreversible,”

- really???? The opposite can be argued:
  - The Longest Chain Rule means probabilistic certitude,
    - HOWEVER in theory EVERY TRANSACTION CAN BE INVALIDATED, (at a large expense),

    ⇒possible even 100 years later

    ⇒if there is a longer chain!

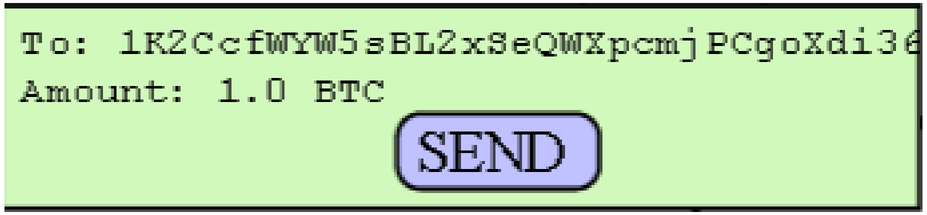
“No intermediary in transactions?”

- Not true (unless one of the parties is a miner)

Bitcoin Transactions:

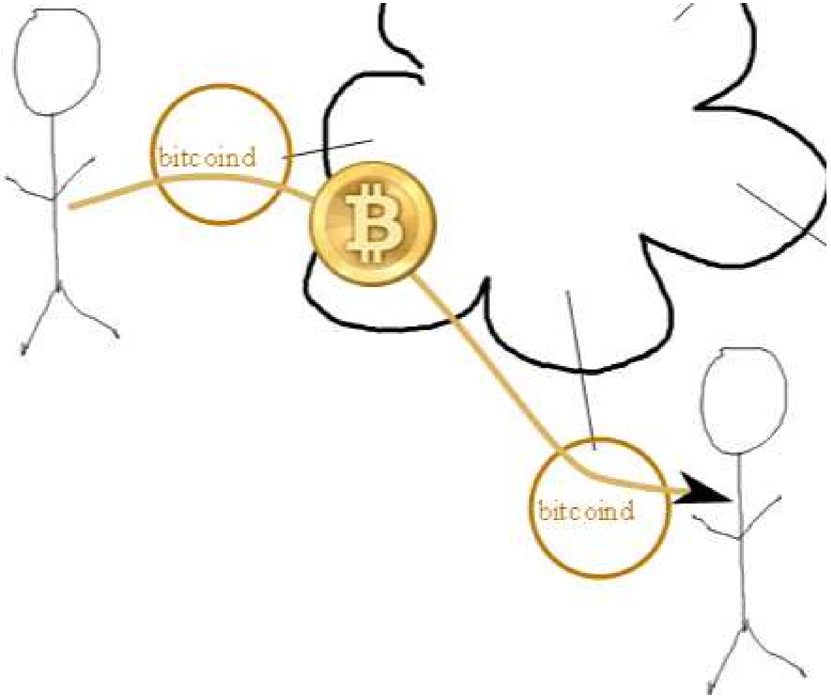
- between any two addresses [and any two network nodes],
  - at any time [no market closing hours].
  - validated within 10-60 minutes.
    - should wait longer for larger transactions, beware of “cheating miners”…
    - many websites accept instantly,
      - they trust your application not to double spend
      - and trust miners to reject the second spent based on later time, easy and plausible!

Transfer



To: 1K2CcfWYW5sBL2xSeQWXpcmjPCgoXdi36

Amount: 1.0 BTC

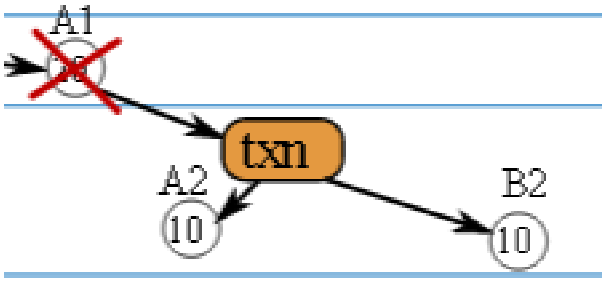


In / Out

Owner of a certain "Attribution to PK" can at any moment transfer it to some other PK addresses.

=> 0 inputs possible if minting transaction… new money.

=> Several outputs are a norm for bitcoin transactions.

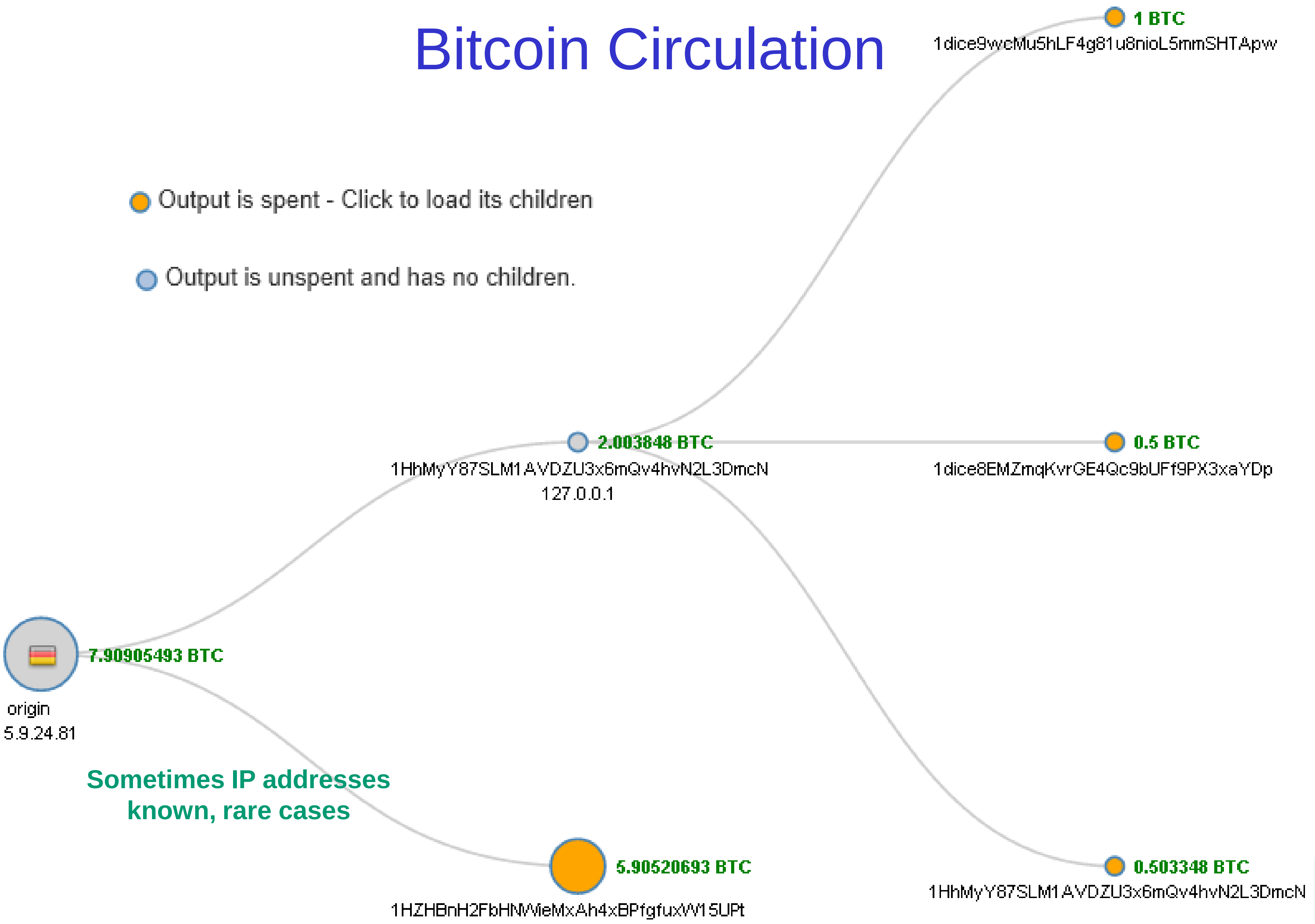


on this picture we ignore the fees

Bitcoin Transfer

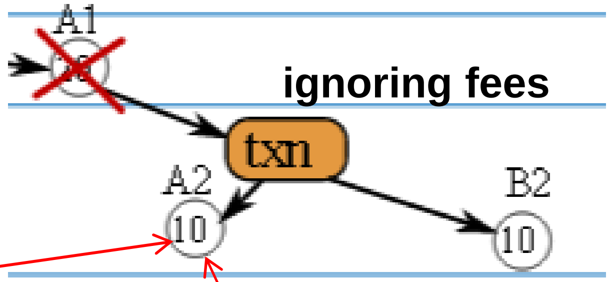
Owner of a certain “Attribution to PK” can at any moment transfer it to any other PK address.

Bitcoin Circulation


● Output is spent - Click to load its children
○ Output is unspent and has no children.
1 BTC
1dice9wcMu5hLF4g81u8nioL5mmSHTApw
2.003848 BTC
1HhMyY87SLM1AVDZU3x6mQv4hvN2L3DmcN
127.0.0.1
0.5 BTC
1dice8EMZmqKvrGE4Qc9bUFf9PX3xaYDp
7.90905493 BTC
origin
5.9.24.81
5.90520693 BTC
1HZHBnH2FbHNWieMxAh4xBPfgfuxW15UPt
0.503348 BTC
1HhMyY87SLM1AVDZU3x6mQv4hvN2L3DmcN

Sometimes IP addresses known, rare cases

Attributions



ignoring fees

DEFINITION

“Attribution to PK” =

act of an owner of
a previous attribution (always destroyed)
which transfers a certain amount to the new PK = A2

(using a digital signature)

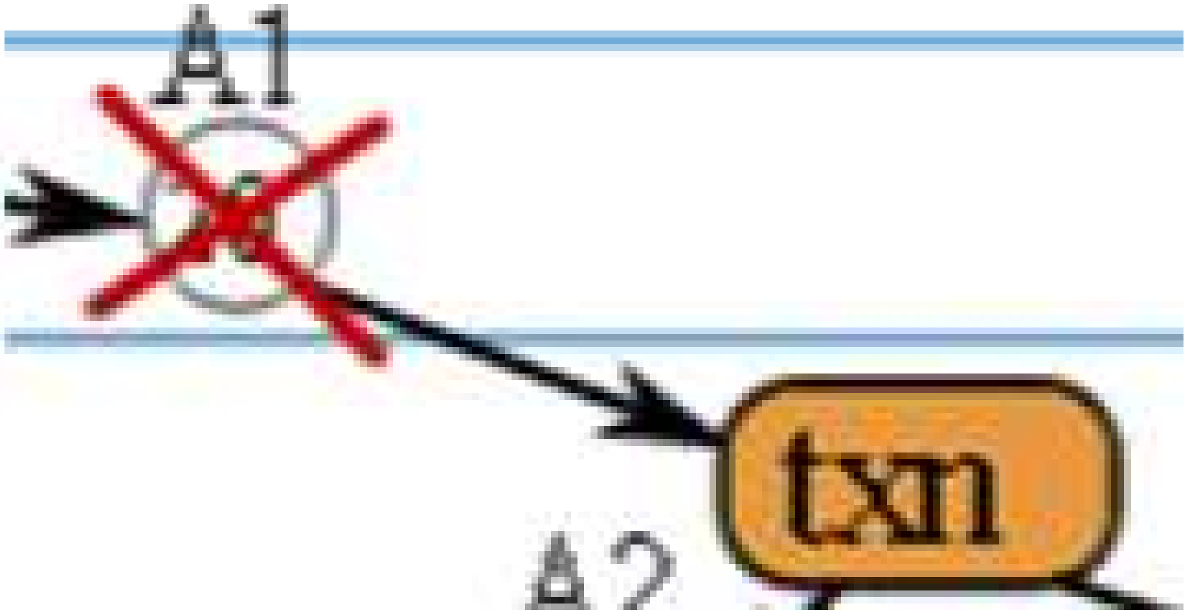
Caveat: Each attribution can be traced back to the initial mining event.

Fragmentation and Summation Rule

Each PK has a balance, say 20 BTC

current balance = sum(unspent attributions).

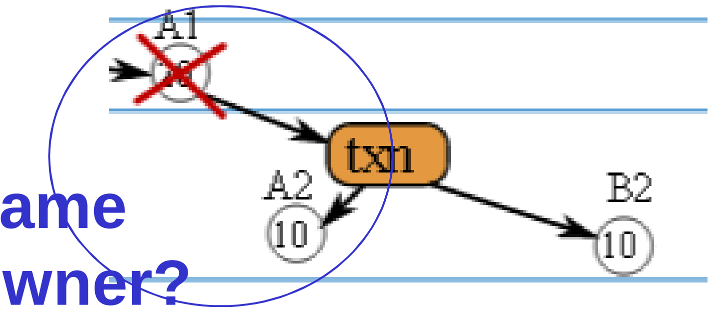
Attributions are ALWAYS destroyed when used,



From Single Attribution

Example

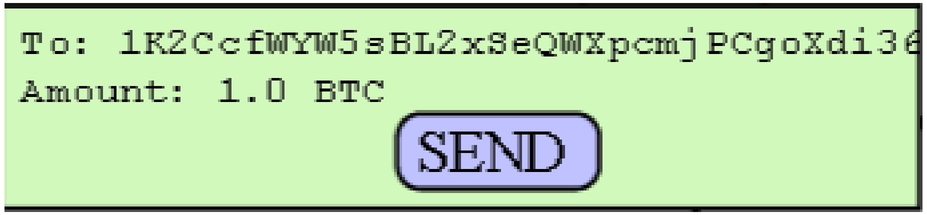
- Change: return some money to ourselves inside the same transaction
  - this implies most transactions have 2 or more outputs
  - most apps use the same address
  - could use another fresh address for better anonymity, but too lazy…



same owner?

no way to know for sure…

With Multiple Attributions



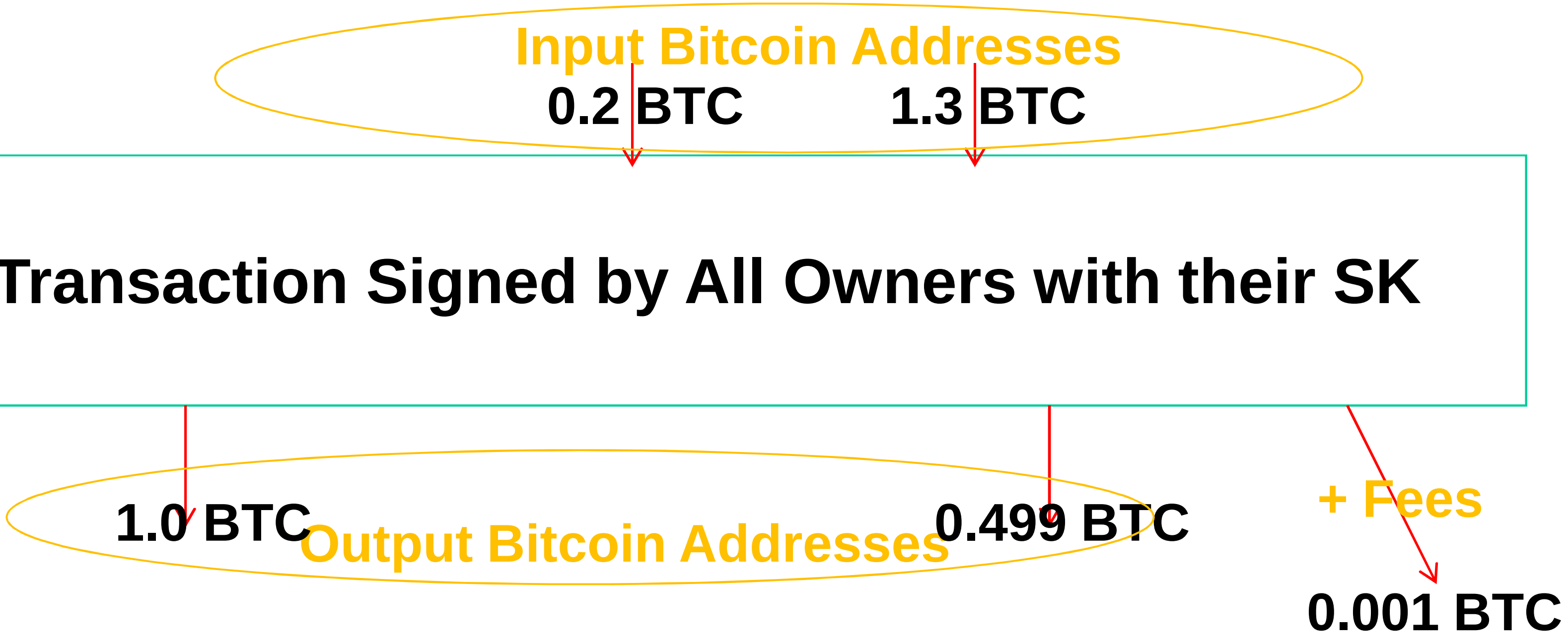
To: 1K2CcfWYW5sBL2xSeQWXpcmjPCgoXdi36

Amount: 1.0 BTC

typical case, even for a single user

Bitcoin Transfer

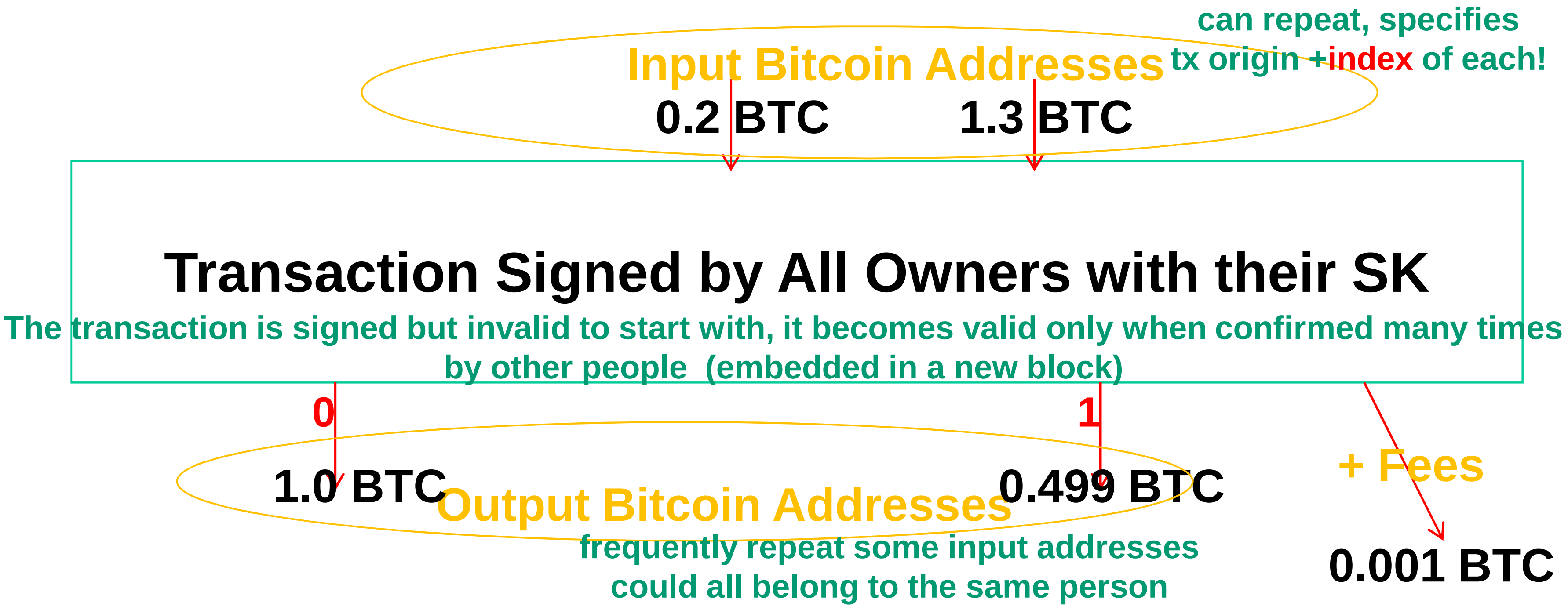
Transactions have multiple inputs and multiple outputs.



Bitcoin Transfer

Transactions have multiple inputs and multiple outputs.

- helps for anonymity.
- destroys all current attributions,
- requires everybody's signature



can repeat, specifies tx origin +index of each!

Input Bitcoin Addresses

0.2 BTC

1.3 BTC

Transaction Signed by All Owners with their SK

The transaction is signed but invalid to start with, it becomes valid only when confirmed many times by other people (embedded in a new block)

0

1

1.0 BTC

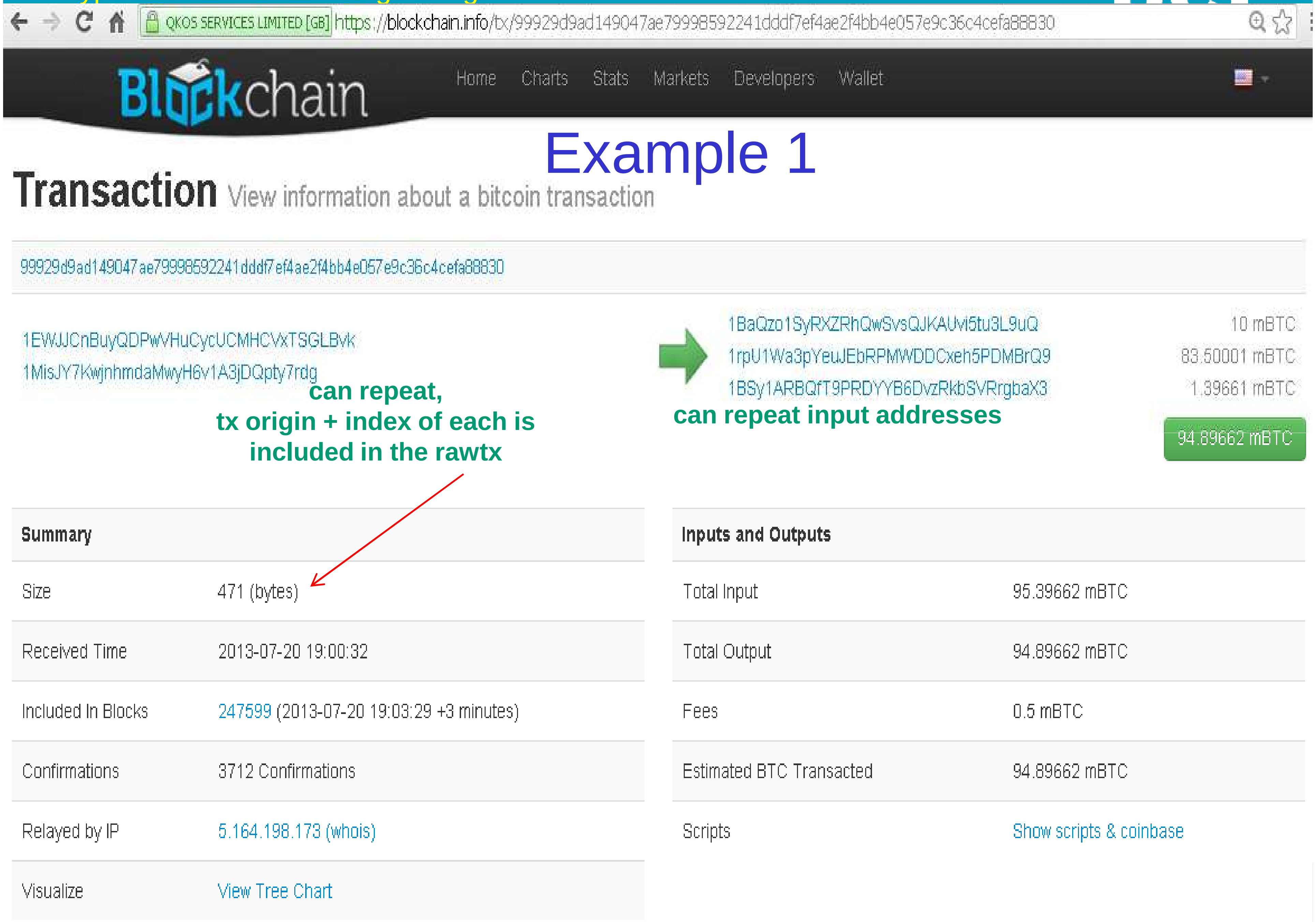
Output Bitcoin Addresses

0.499 BTC

+ Fees

0.001 BTC

frequently repeat some input addresses could all belong to the same person



QKOS SERVICES LIMITED [GB] https://blockchain.info/tx/99929d9ad149047ae79998592241dddf7ef4ae2f4bb4e057e9c36c4cefa88830

Example 1

Transaction View information about a bitcoin transaction

99929d9ad149047ae79998592241dddf7ef4ae2f4bb4e057e9c36c4cefa88830

1EWJJCnBuyQDPwVHuCycUCMHCVxTSGLBvk
1MisJY7KwjnhmdaMwyH6v1A3jDQpty7rdg

1BaQzo1SyRXZRhQwSvsQJKAUvi5tu3L9uQ	10 mBTC
1rpU1Wa3pYeuJEbRPMWDDCxeh5PDMBrQ9	83.50001 mBTC
1BSy1ARBQfT9PRDYYB6DvzRkbSVRgbaX3	1.39661 mBTC
	94.89662 mBTC

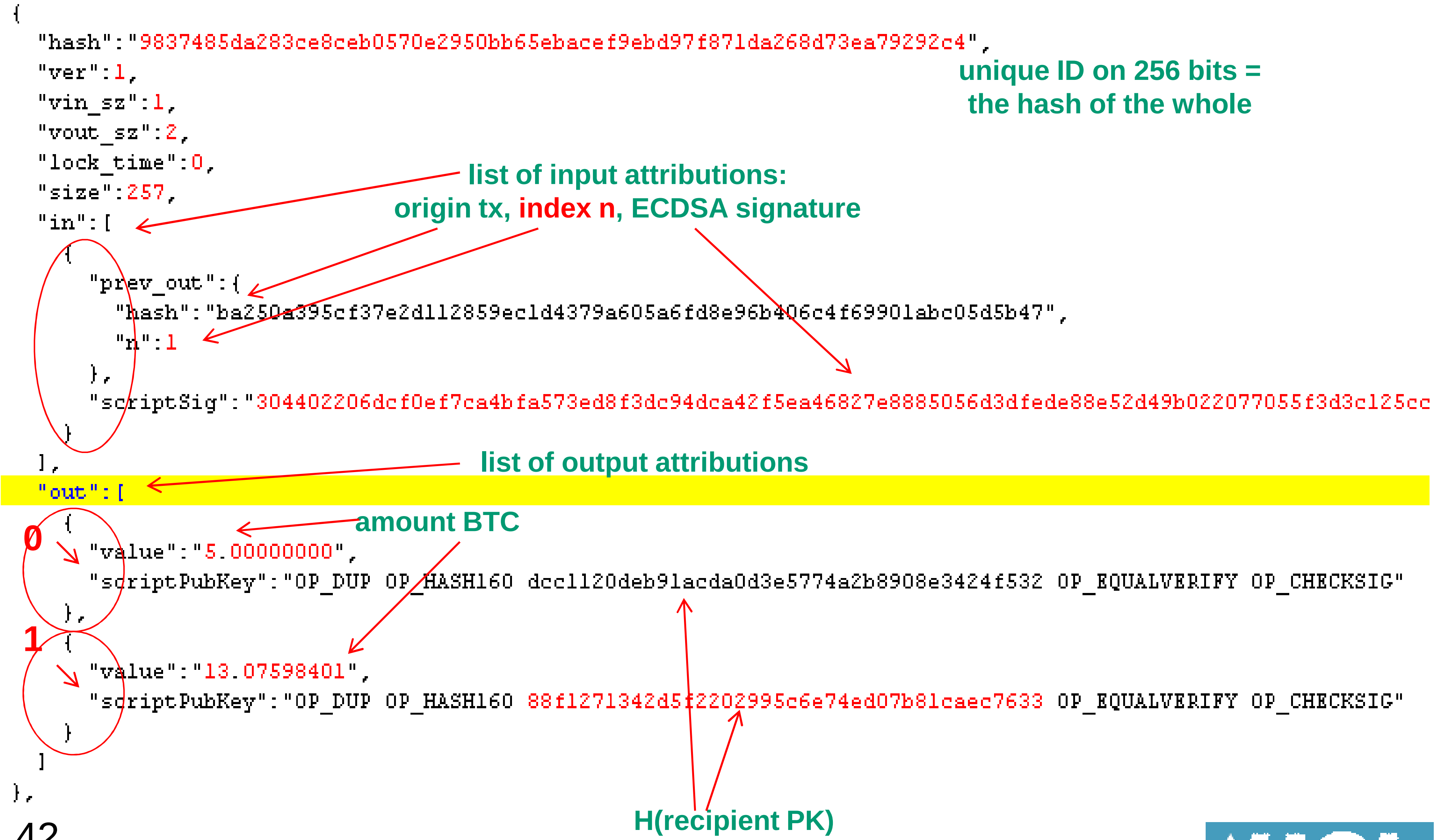
can repeat,
tx origin + index of each is
included in the rawtx

can repeat input addresses

Summary	
Size	471 (bytes)
Received Time	2013-07-20 19:00:32
Included In Blocks	247599 (2013-07-20 19:03:29 +3 minutes)
Confirmations	3712 Confirmations
Relayed by IP	5.164.198.173 (whois)
Visualize	View Tree Chart

Inputs and Outputs	
Total Input	95.39662 mBTC
Total Output	94.89662 mBTC
Fees	0.5 mBTC
Estimated BTC Transacted	94.89662 mBTC
Scripts	Show scripts & coinbase

Example 2 = Raw Transaction



```
{
  "hash":"9837485da283ce8ceb0570e2950bb65ebacef9ebd97f871da268d73ea79292c4",
  "ver":1,
  "vin_sz":1,
  "vout_sz":2,
  "lock_time":0,
  "size":257,
  "in":[
    {
      "prev_out":{
        "hash":"ba250a395cf37e2d112859ec1d4379a605a6fd8e96b406c4f69901abc05d5b47",
        "n":1
      },
      "scriptSig":"304402206dcf0ef7ca4bfa573ed8f3dc94dca42f5ea46827e8885056d3dfede88e52d49b022077055f3d3c125cc
    }
  ],
  "out":[
    {
      "value":"5.00000000",
      "scriptPubKey":"OP_DUP OP_HASH160 dcc1120deb91acda0d3e5774a2b8908e3424f532 OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value":"13.07598401",
      "scriptPubKey":"OP_DUP OP_HASH160 88f1271342d5f2202995c6e74ed07b81caec7633 OP_EQUALVERIFY OP_CHECKSIG"
    }
  ]
},
```

unique ID on 256 bits =
the hash of the whole
list of input attributions:
origin tx, index n, ECDSA signature
list of output attributions
amount BTC
0
1
H(recipient PK)

Remarks:

About 30 million transactions ever made.

To know the balance of one account, we must “in theory” store ALL the transactions which send money for this address and then check ALL transactions made since then to see some of these are not already spent.

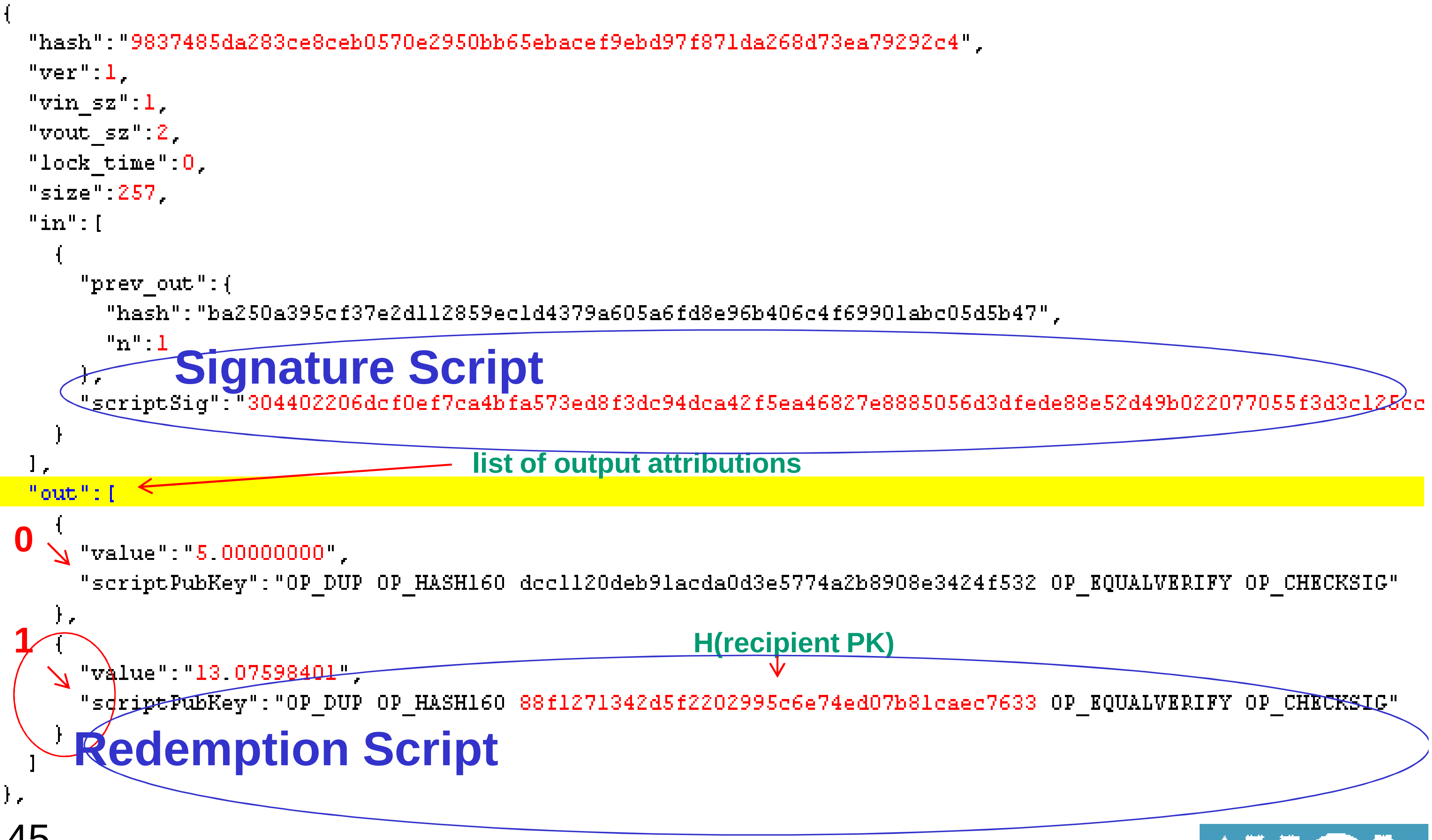
Full bitcoin network nodes stored all transactions ever made and checks their correctness (all the digital signatures).

About 15 Gbytes data, 24 hours full download.

In practice one could skip check for things confirmed by many miners… dangerous though. There is no absolute proof that miners have already checked them (maybe they forgot, a bug).

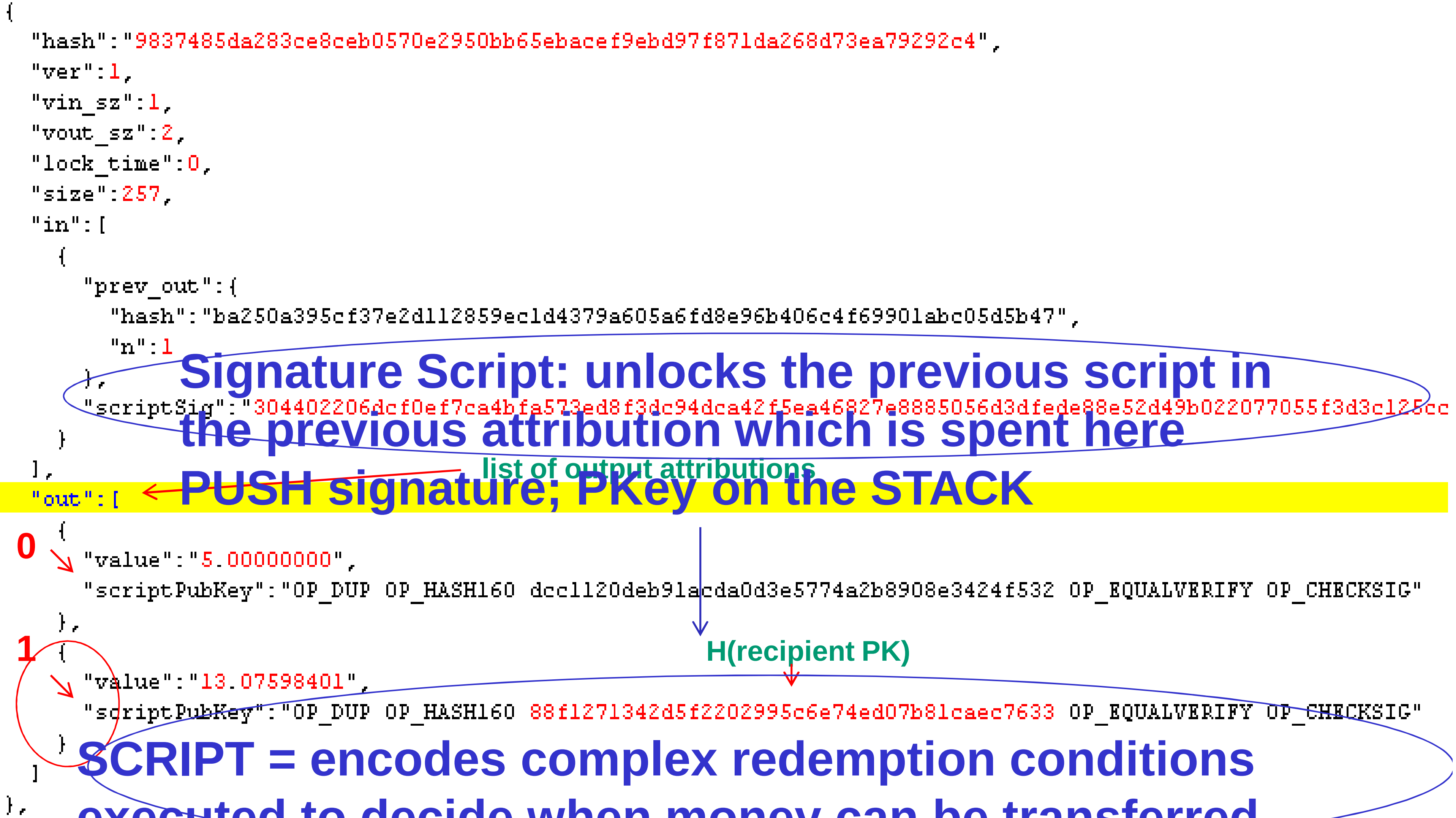
Transaction Scripts

***Scripts



```
{
  "hash":"9837485da283ce8ceb0570e2950bb65ebacef9ebd97f871da268d73ea79292c4",
  "ver":1,
  "vin_sz":1,
  "vout_sz":2,
  "lock_time":0,
  "size":257,
  "in":[
    {
      "prev_out":{
        "hash":"ba250a395cf37e2d112859ec1d4379a605a6fd8e96b406c4f69901abc05d5b47",
        "n":1
      },
      "scriptSig":"304402206dcf0ef7ca4bfa573ed8f3dc94dca42f5ea46827e8885056d3dfede88e52d49b022077055f3d3c125cc
    }
  ],
  "out":[
    {
      "value":"5.00000000",
      "scriptPubKey":"OP_DUP OP_HASH160 dcc1120deb91acda0d3e5774a2b8908e3424f532 OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value":"13.07598401",
      "scriptPubKey":"OP_DUP OP_HASH160 88f1271342d5f2202995c6e74ed07b81caec7633 OP_EQUALVERIFY OP_CHECKSIG"
    }
  ]
},
```

Executing Scripts



```
{
  "hash":"9837485da283ce8ceb0570e2950bb65ebacef9ebd97f871da268d73ea79292c4",
  "ver":1,
  "vin_sz":1,
  "vout_sz":2,
  "lock_time":0,
  "size":257,
  "in":[
    {
      "prev_out":{
        "hash":"ba250a395cf37e2d112859ec1d4379a605a6fd8e96b406c4f69901abc05d5b47",
        "n":1
      },
      "scriptSig":"304402206dcf0ef7ca4bfa573ed8f3dc94dca42f5ea46827e8885056d3dfede88e52d49b022077055f3d3c125cc
    }
  ],
  "out":[
    {
      "value":"5.00000000",
      "scriptPubKey":"OP_DUP OP_HASH160 dcc1120deb91acda0d3e5774a2b8908e3424f532 OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value":"13.07598401",
      "scriptPubKey":"OP_DUP OP_HASH160 88f1271342d5f2202995c6e74ed07b81caec7633 OP_EQUALVERIFY OP_CHECKSIG"
    }
  ]
},
```

Signature Script: unlocks the previous script in the previous attribution which is spent here

list of output attributions

PUSH signature; PKey on the STACK

0

1

H(recipient PK)

SCRIPT = encodes complex redemption conditions executed to decide when money can be transferred

*Multiple signers:

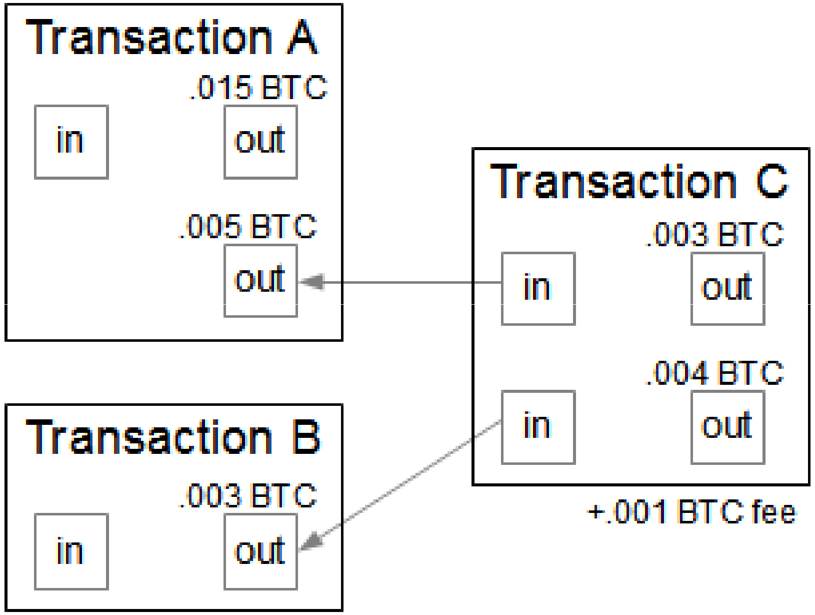
Issues:

- Who signs first?
  - In any order.
- What if one signs and other refuse?
  - Transaction is non-existent.
  - Cannot be used to sign something different.
- Do they KNOW what are they signing?
  - Yes, well, not sure

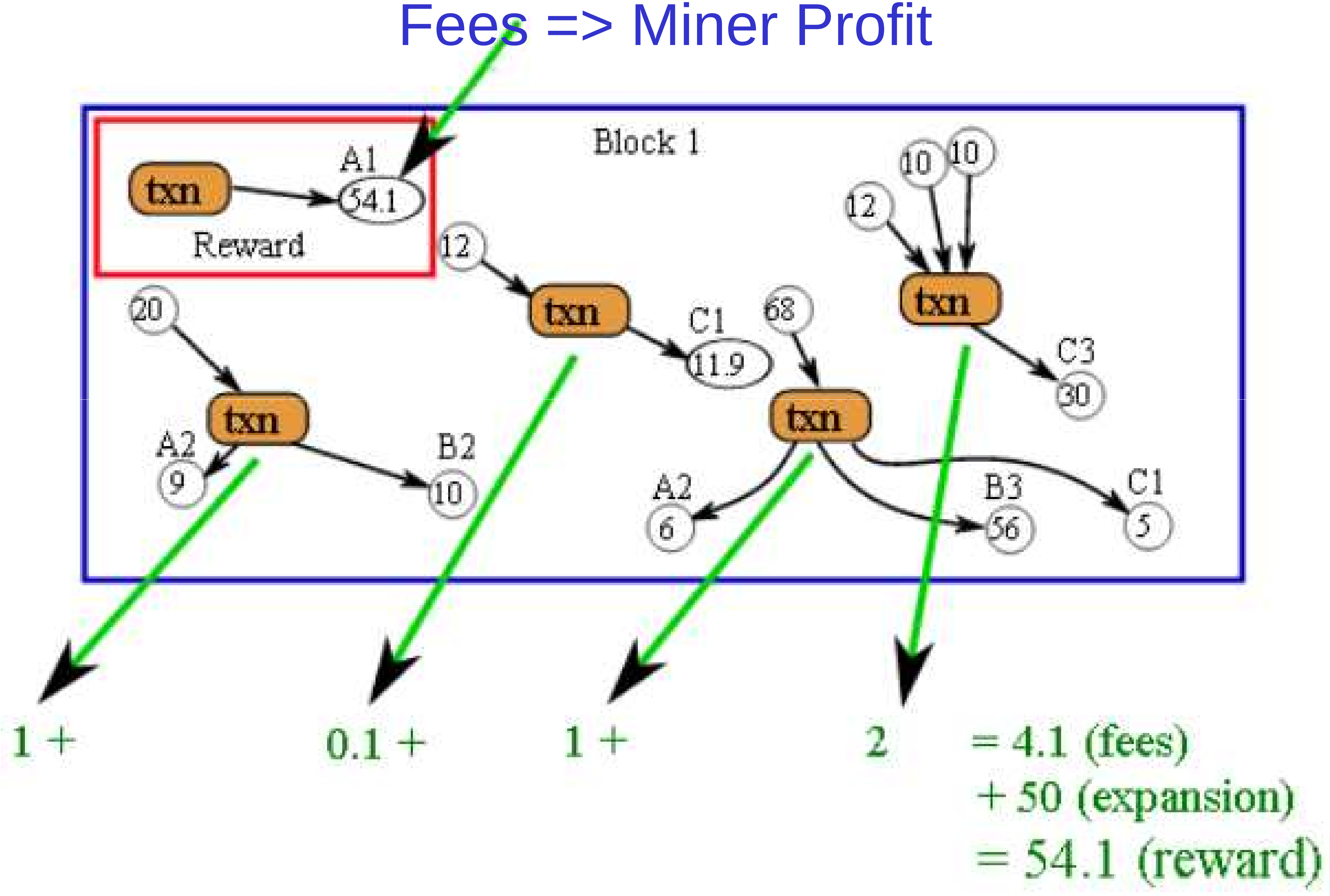
- What if some other inputs in this transaction are involved in illegal activity?

Transaction Chaining

2 attributions:



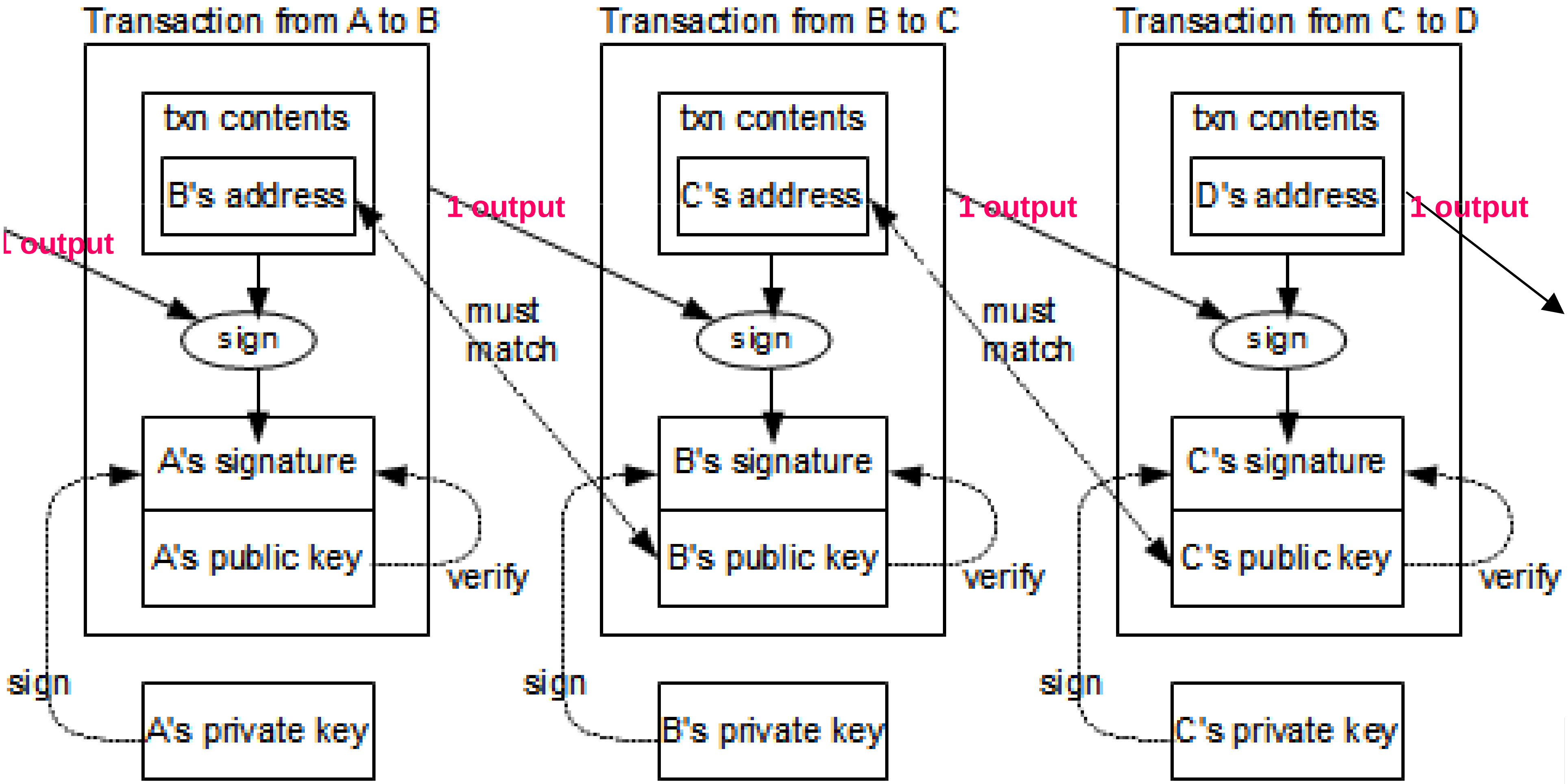
Fees => Miner Profit



1 + 0.1 + 1 + 2 = 4.1 (fees)
+ 50 (expansion)
= 54.1 (reward)

*Chaining and Checks

one branch of a tree:



Spot On Signatures

Bottom Line

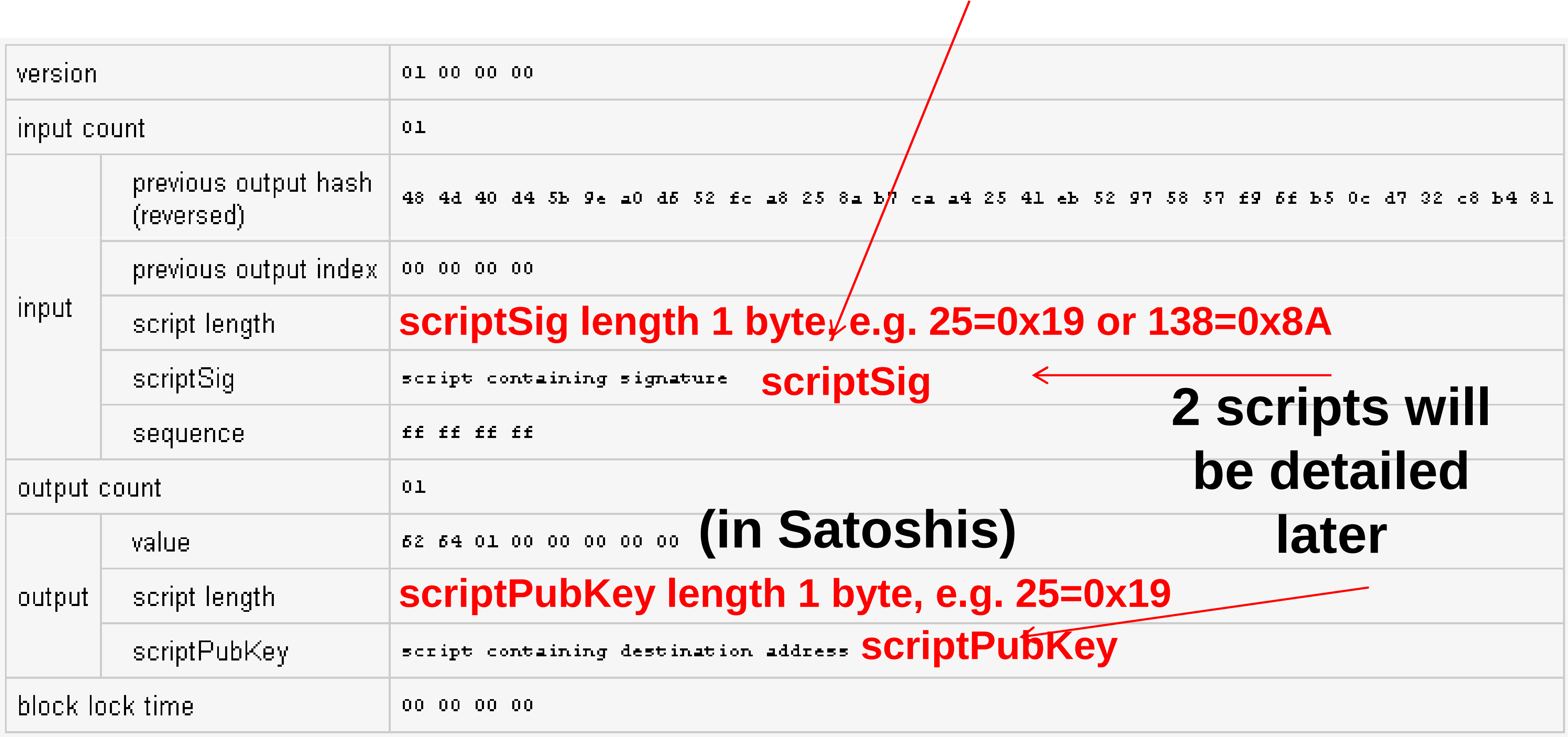
What gets signed in bitcoin and what is used in the P2P network are very different things.

This is inevitable:

- a digital signature cannot sign itself:
  - bitcoin uses hashes to make transactions impossible to modify, these hashes include the signatures.
  - but in order to produce the signature, the hash must be a different value, precisely because the signature is not yet computed.
  - what is done in practice is like “we hash+sign the raw transactions sigs removed, 1 script inserted”… details follow
    - this is problematic, see MtGox incident later on

Signed Tx / Final Tx

byte by byte (similar but not identical to raw blocks seen before)
(this is done twice, with different scriptSig)

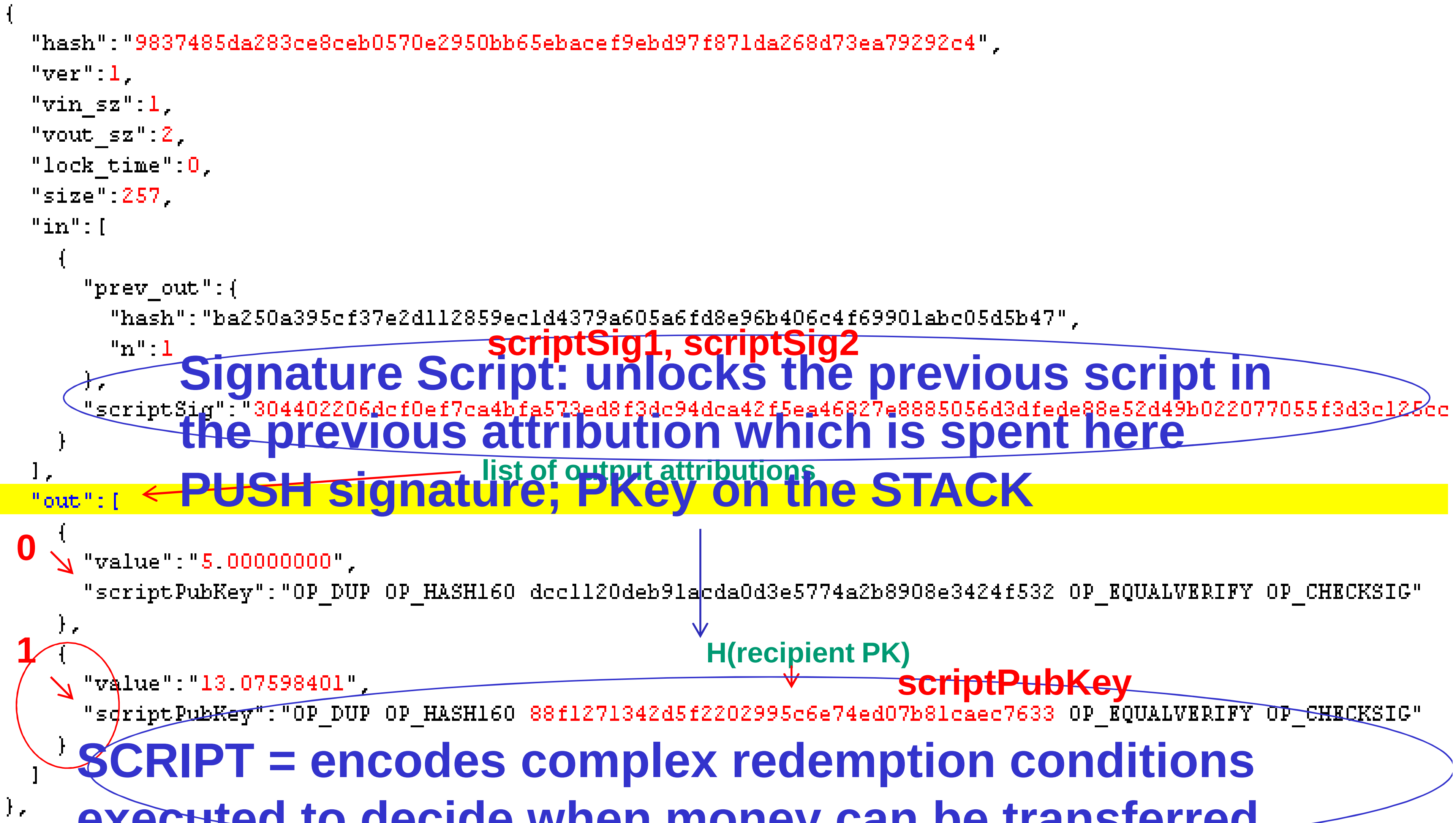


version		01 00 00 00
input count		01
input	previous output hash (reversed)	48 4d 40 d4 5b 9e a0 d6 52 fc a8 25 8a b7 ca a4 25 41 eb 52 97 58 57 f9 6f b5 0c d7 32 c8 b4 81
	previous output index	00 00 00 00
	script length	**scriptSig length 1 byte, e.g. 25=0x19 or 138=0x8A**
	scriptSig	script containing signature **scriptSig**
	sequence	ff ff ff ff
output count		01
output	value	62 64 01 00 00 00 00 00 **(in Satoshis)**
	script length	**scriptPubKey length 1 byte, e.g. 25=0x19**
	scriptPubKey	script containing destination address **scriptPubKey**
block lock time		00 00 00 00

2 scripts will be detailed later

len(1i/1o)= 223=4+1+32+4+1+ 1+71+ 1+65+ 4+1+8+ 1+25+4

**nearly same scripts appear here:



```
{
  "hash":"9837485da283ce8ceb0570e2950bb65ebacef9ebd97f871da268d73ea79292c4",
  "ver":1,
  "vin_sz":1,
  "vout_sz":2,
  "lock_time":0,
  "size":257,
  "in":[
    {
      "prev_out":{
        "hash":"ba250a395cf37e2d112859ec1d4379a605a6fd8e96b406c4f69901abc05d5b47",
        "n":1
      },
      "scriptSig":"304402206dcf0ef7ca4bfa573ed8f3dc94dca42f5ea46827e8885056d3dfede88e52d49b022077055f3d3c125cc
    }
  ],
  "out":[
    {
      "value":"5.00000000",
      "scriptPubKey":"OP_DUP OP_HASH160 dcc1120deb91acda0d3e5774a2b8908e3424f532 OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value":"13.07598401",
      "scriptPubKey":"OP_DUP OP_HASH160 88f1271342d5f2202995c6e74ed07b81caec7633 OP_EQUALVERIFY OP_CHECKSIG"
    }
  ]
},
```

Signature Script: unlocks the previous script in the previous attribution which is spent here

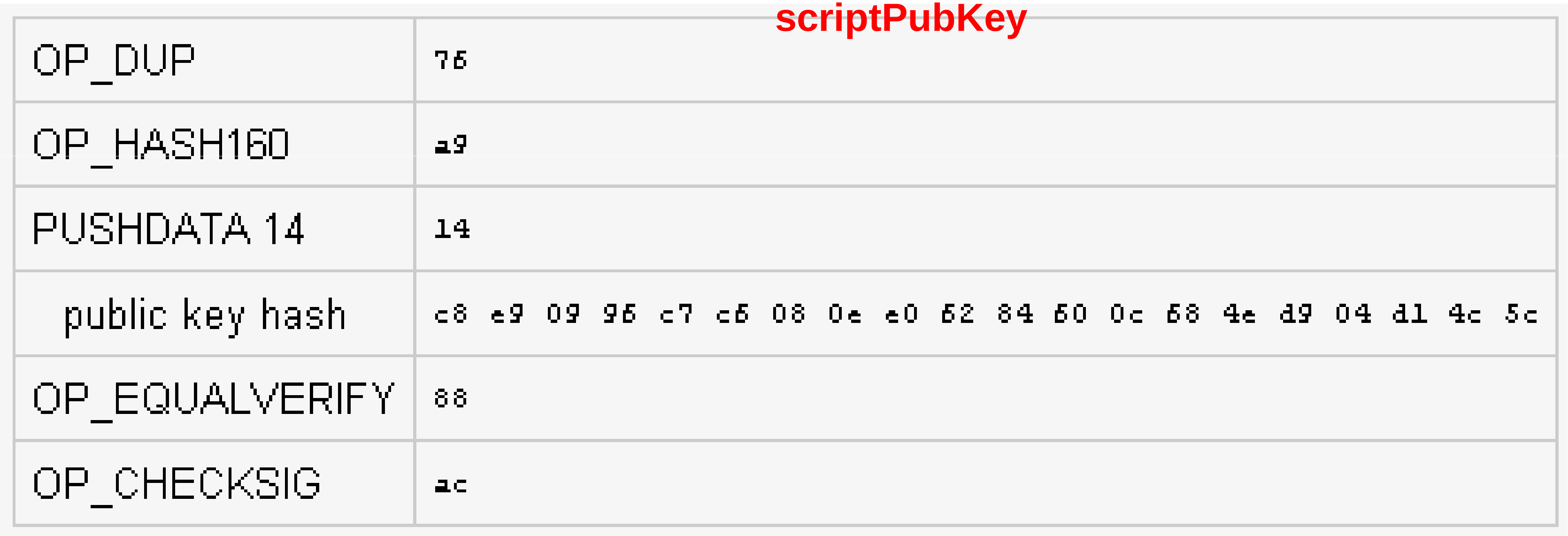
PUSH signature; PKey on the STACK

SCRIPT = encodes complex redemption conditions executed to decide when money can be transferred

Example of scriptPubKey

= a script which verifies if one can spend bitcoins
(who and under which conditions)

scriptPubKey



OP_DUP	76
OP_HASH160	a9
PUSHDATA 14	14
public key hash	c8 e9 09 96 c7 c6 08 0e e0 62 84 60 0c 68 4e d9 04 d1 4c 5c
OP_EQUALVERIFY	88
OP_CHECKSIG	ac

len= 25=3+20+2

First scriptSig

It is scriptPubKey BUT copied from the previous transaction (peculiarity)

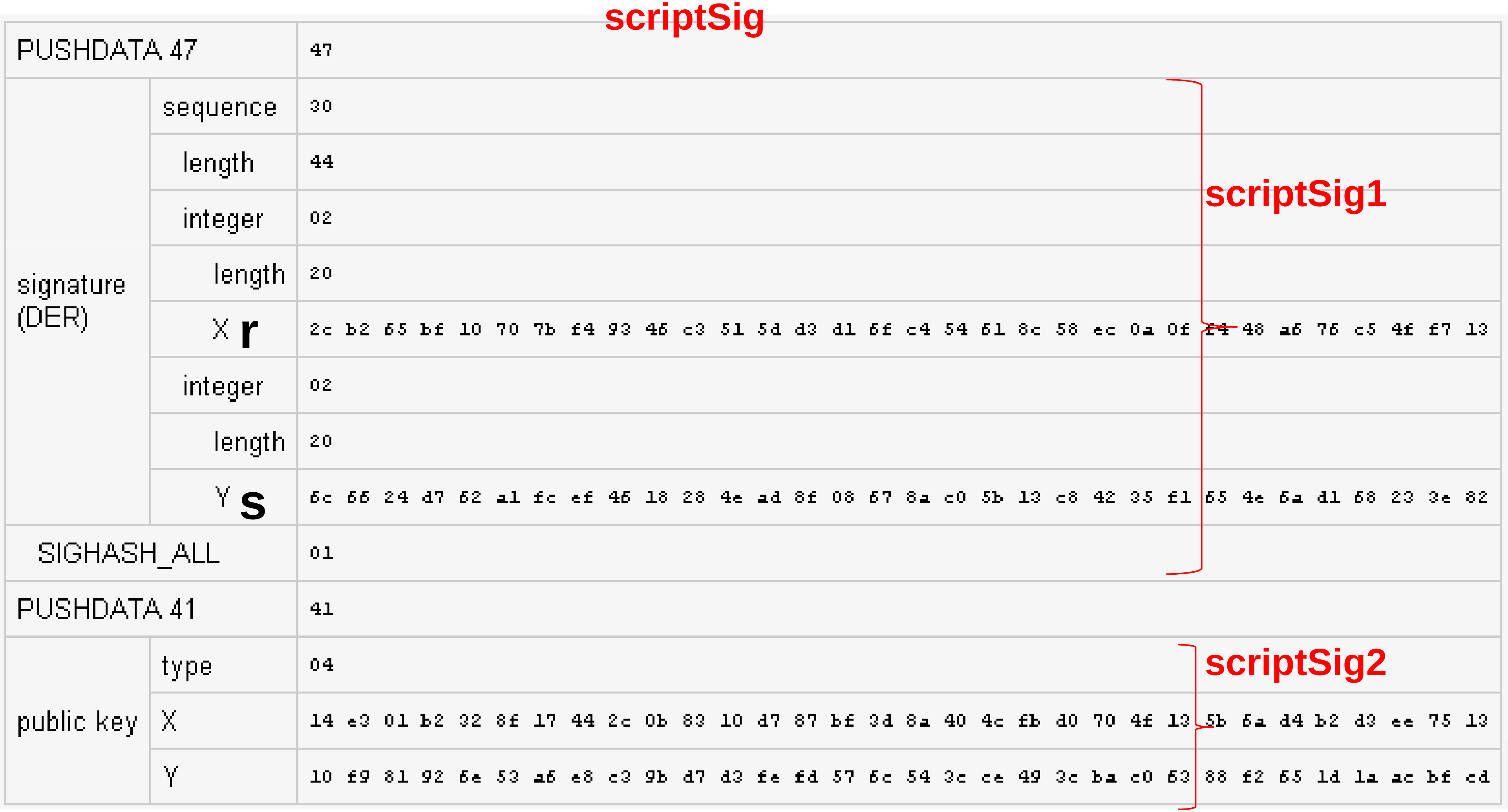
len= 25=3+20+2 typically

Second scriptSig

sign+PKey

len= 1+71+ 1+65 = 138 BUT NOT ALWAYS!

scriptSig



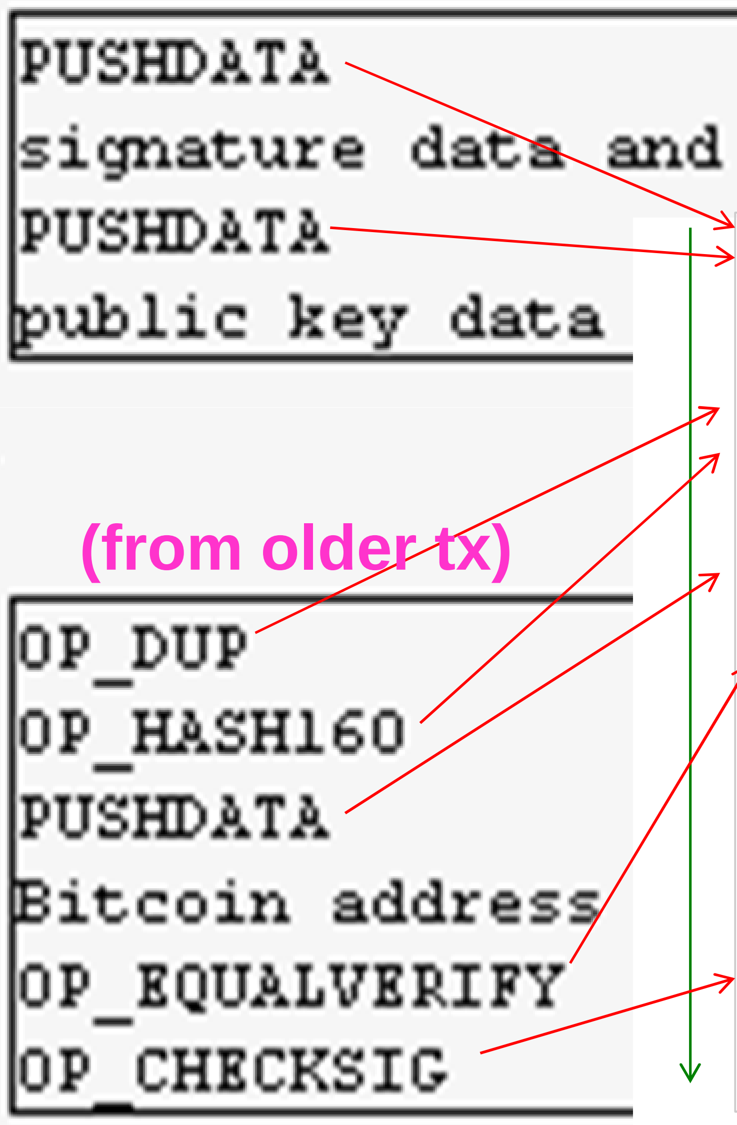
PUSHDATA 47		47
	sequence	30
	length	44
	integer	02
signature (DER)	length	20
	X **r**	2c b2 65 bf 10 70 7b f4 93 46 c3 51 5d d3 d1 6f c4 54 61 8c 58 ec 0a 0f f4 48 a6 76 c5 4f f7 13
	integer	02
	length	20
	Y **s**	6c 66 24 d7 62 a1 fc ef 46 18 28 4e ad 8f 08 67 8a c0 5b 13 c8 42 35 f1 65 4e 6a d1 68 23 3e 82
SIGHASH_ALL		01
PUSHDATA 41		41
public key	type	04
	X	14 e3 01 b2 32 8f 17 44 2c 0b 83 10 d7 87 bf 3d 8a 40 4c fb d0 70 4f 13 5b 6a d4 b2 d3 ee 75 13
	Y	10 f9 81 92 6e 53 a6 e8 c3 9b d7 d3 fe fd 57 6c 54 3c ce 49 3c ba c0 63 88 f2 65 1d 1a ac bf cd

scriptSig1

scriptSig2

*Detailed Execution = Signature Verification

The simplest case:



```
PUSHDATA
signature data and SIGHASH_ALL
PUSHDATA
public key data
```

(from older tx)

```
OP_DUP
OP_HASH160
PUSHDATA
Bitcoin address
OP_EQUALVERIFY
OP_CHECKSIG
```

PUSHDATA pushes the signature + Pkey to the stack.

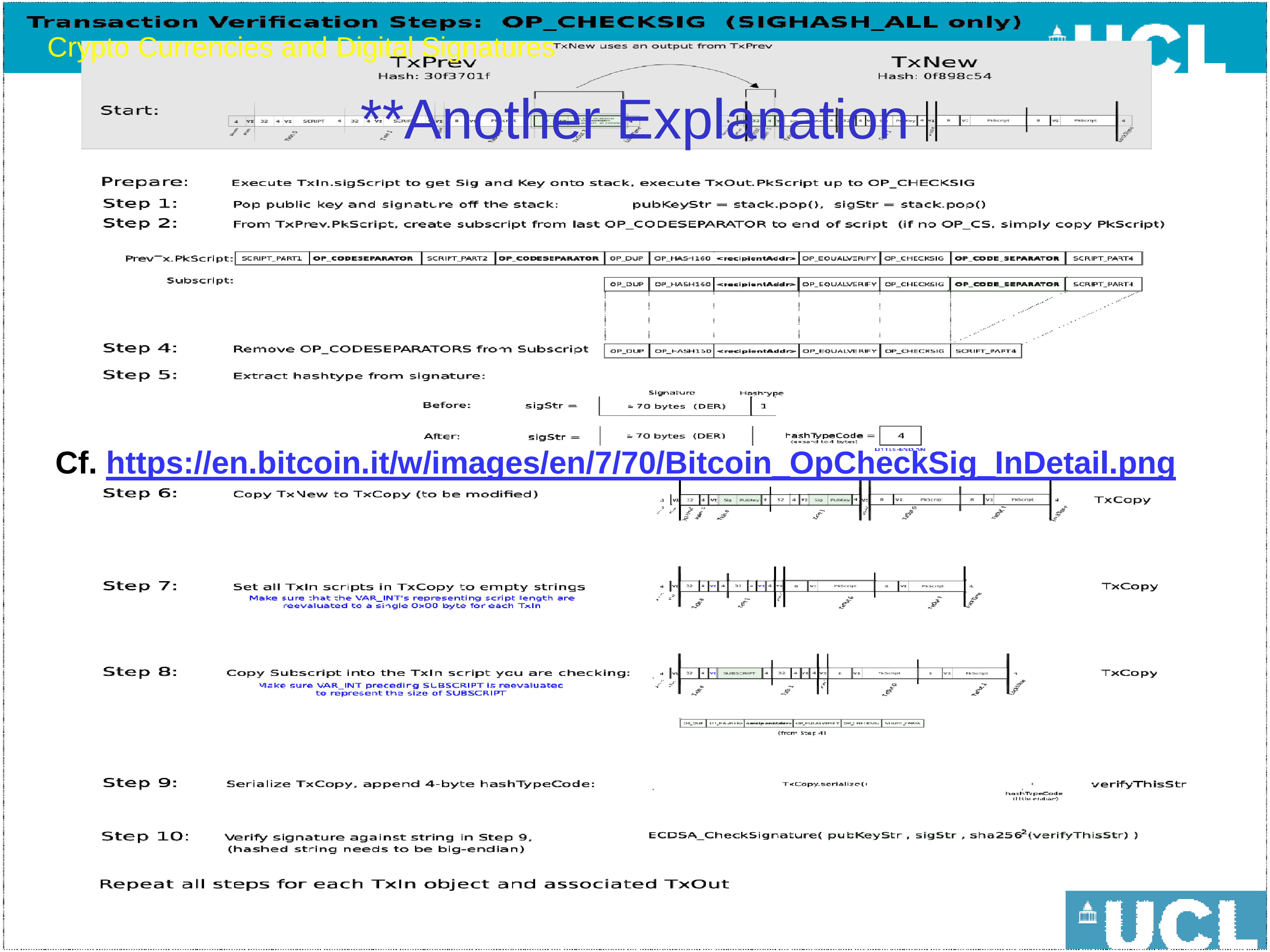
OP_DUP duplicates the public key on the stack.
OP_HASH160 computes the 160-bit hash of the public key.
PUSHDATA pushes the result = Bitcoin address.

Then OP_EQUALVERIFY verifies the top two addresses are equal - the public key hash from the new transaction must matches the address in the old script.
This proves that the public key is valid.

OP_CHECKSIG checks that the signature of the transaction matches the public key and signature on the stack. This proves that the signature is valid.

Crypto Currencies and Digital Signatures

**Another Explanation



Transaction Verification Steps: OP_CHECKSIG (SIGHASH_ALL only)

TxNew uses an output from TxPrev

TxPrev
Hash: 30f3701f

TxNew
Hash: 0f898c54

Start:

Prepare: Execute TxIn.sigScript to get Sig and Key onto stack, execute TxOut.PkScript up to OP_CHECKSIG

Step 1: Pop public key and signature off the stack: pubKeyStr = stack.pop(), sigStr = stack.pop()

Step 2: From TxPrev.PkScript, create subscript from last OP_CODESEPARATOR to end of script (if no OP_CS, simply copy PkScript)

PrevTx.PkScript: SCRIPT_PART1 | OP_CODESEPARATOR | SCRIPT_PART2 | OP_CODESEPARATOR | OP_DUP | OP_HASH160 | <recipientAddr> | OP_EQUALVERIFY | OP_CHECKSIG | OP_CODE_SEPARATOR | SCRIPT_PART4

Subscript: OP_DUP | OP_HASH160 | <recipientAddr> | OP_EQUALVERIFY | OP_CHECKSIG | OP_CODE_SEPARATOR | SCRIPT_PART4

Step 4: Remove OP_CODESEPARATORS from Subscript: OP_DUP | OP_HASH160 | <recipientAddr> | OP_EQUALVERIFY | OP_CHECKSIG | SCRIPT_PART4

Step 5: Extract hashtype from signature:

Signature | Hash-type

Before: sigStr = ≈ 70 bytes (DER) | 1

After: sigStr = ≈ 70 bytes (DER) | hashTypeCode = 4 (expand to 4 bytes: LITTLE-ENDIAN)

Cf. https://en.bitcoin.it/w/images/en/7/70/Bitcoin_OpCheckSig_InDetail.png

Step 6: Copy TxNew to TxCopy (to be modified) TxCopy

Step 7: Set all TxIn scripts in TxCopy to empty strings TxCopy
Make sure that the VAR_INT's representing script length are reevaluated to a single 0x00 byte for each TxIn

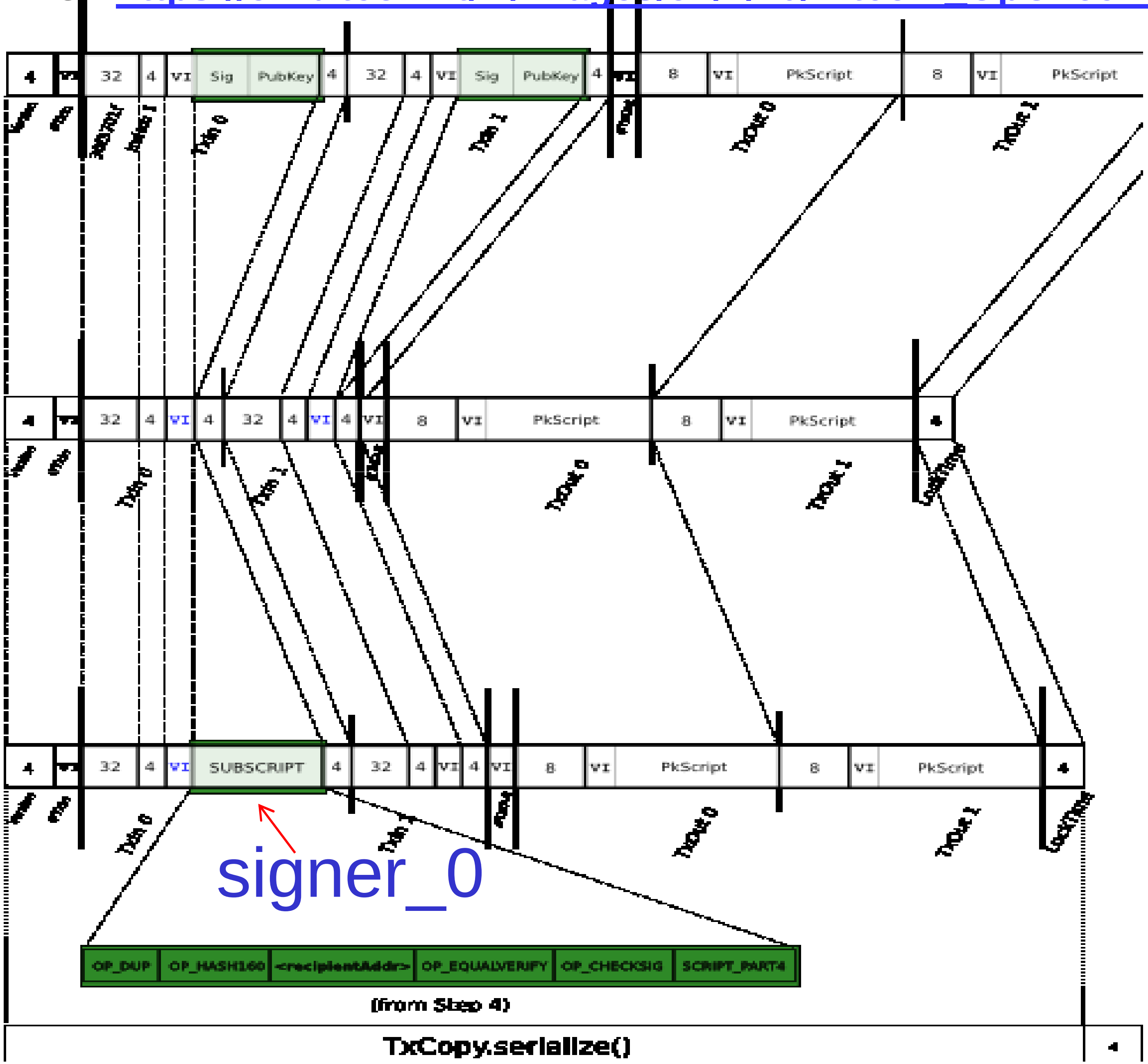
Step 8: Copy Subscript into the TxIn script you are checking: TxCopy
Make sure VAR_INT preceding SUBSCRIPT is reevaluated to represent the size of SUBSCRIPT
(from Step 4)

Step 9: Serialize TxCopy, append 4-byte hashTypeCode: TxCopy.serialize() hashTypeCode (little-endian) verifyThisStr

Step 10: Verify signature against string in Step 9, (hashed string needs to be big-endian) ECDSA_CheckSignature(pubKeyStr , sigStr , sha256^2(verifyThisStr))

Repeat all steps for each TxIn object and associated TxOut

Cf. https://en.bitcoin.it/w/images/en/7/70/Bitcoin_OpCheckSig_InDetail.png



signer_0

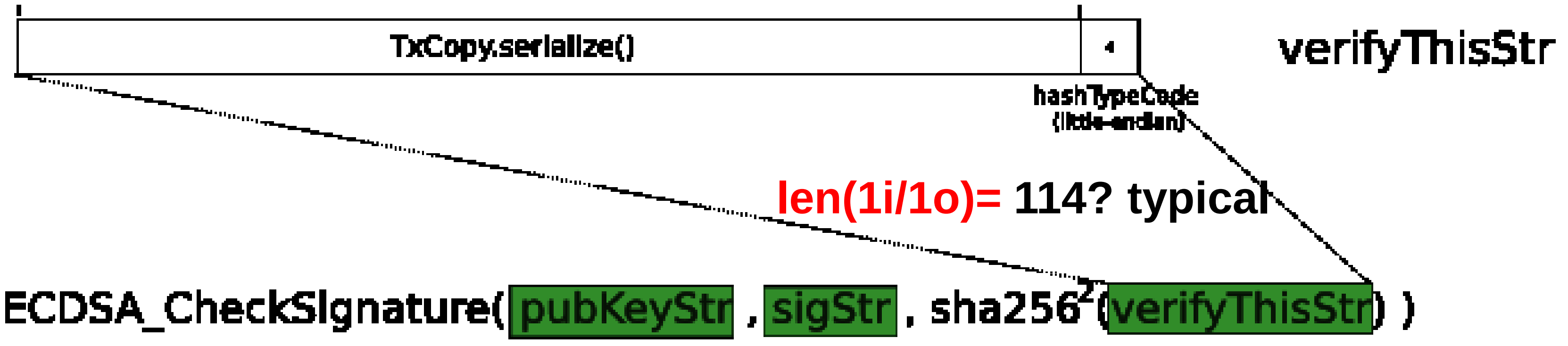
raw tx
with several
signatures

see SignatureHash()
inside script.cpp

to be signed

now hash+verif

<< nHashType
= SIGHASH_ALL
= 01 00 00 00


TxCopy.serialize()
4
hashTypeCode
(little-endian)
verifyThisStr
ECDSA_CheckSignature(pubKeyStr , sigStr , sha256² (verifyThisStr))

len(1i/1o)= 114? typical

see GetHash() inside
hash.h

MtGox Goes Bananas

Or Bitcoin Becomes A Hostage Held in Japan

MtGox Mysterious Fail 2014

Tokyo - JAPAN - February 07, 2014, MtGox press release:

- All bitcoin withdrawal requests will be on pause!
- However they still allow to exchange to USD or other currencies and withdraw…
  - Which will artificially drive down the market price of bitcoin. Is this legal????
    - Example: on 14022014 Bitcoin Price Index is 612 USD, except at MtGox it is at 328 USD!!!!
    - Later it went down to 130 USD and MtGox have closed on 25 Feb 20014.
  - Private question:
    - Is this related to some criminal investigations?
    - in this way criminals CANNOT withdraw money with identifying themselves (because it goes to a legit bank account).

MtGox Goes Bananas

Tokyo - JAPAN - February 10, 2014, MtGox press release:

- They claim theres is a bug in Bitcoin… “signature malleability”
- They propose to CREATE AND STANDARDIZE A NEW HASH for transaction tracking purposes.
  - this new transaction hash will allow signing parties to keep track of any transaction they have signed and can easily be computed, even for past transactions.
- “We have discussed this solution with the Bitcoin core developers and will allow Bitcoin withdrawals again once it has been approved and standardized”.
  - **OUTRAGEOUS** because there is NO REASON to do that
  - this is like holding bitcoins hostage
  - signature bug was known since 2011 and is benign?, see later slides
  - you cannot expect bitcoin spec to be changed on request (expected to be a decentralized currency)
  - hidden reasons?

MtGox Does Sth Really Strange

Next day, February 11, 2014,

– Bitstamp, another exchange also suspended withdrawals, claiming DOS attack.

– they promised to resume withdrawals in 2 days…

Consequences of this = a bank run: people exchange to normal currencies more than usual, the only way to withdraw money, so bitcoin value drops…

Such DOS very easily allows criminals to make profits:

– DOS => buy bitcoins => stop DOS => sell bitcoins.

Is It Actually True?

Later statement:

- "The issues that Mt. Gox has been experiencing are due to an unfortunate interaction between Mt. Gox's implementation of their highly customized wallet software, their customer support procedures, and their unpreparedness for transaction malleability, a technical detail that allows changes to the way transactions are identified," the Bitcoin Foundation said.

Missing 350 M$

- Late Feb 2013

- MtGox is down.

…it discovered that an estimated 744,000 bitcoins
about $350m had been stolen
due to a loophole in its security.

Bankruptcy

- 28 feb 2013

- MtGox filed for bankruptcy protection at a district court in Tokyo.
  - at the court hearing, the company said it had outstanding debts of about 6.5bn yen (£38m).
  - MtGox's lawyers are believed to have decided to apply to the court for protection after US regulators filed a subpoena against the company.

Malleability

How To Steal Money From Exchanges?

Short explanation:

When a user asks to withdraw some bitcoins from his account, the exchange generates a bitcoin transaction, and *keeps record of its hash value H1*.

They simply check whether a transaction with such hash value *H1* has been included in the blockchain? to confirm the user had effectively received his funds, some 10 or more minutes after its broadcast in P2P network.

The attacker manages to create an altered version or the original transaction with the same content and different hash H2, and to get it into the blockchain BEFORE H1 (race condition). The exchange will be fooled and may fail to see that the transaction went through. The money is withdrawn but the exchange thinks the transaction was NOT approved for some reason.

The user can then withdraw it again, and again (several times!).

Possible because (apparently) MtGox were too lazy to check such cases, they have allowed the customer's to request a repetition of an uncommitted transaction, and this happened in an automated way without human intervention and without any checks.

Is It Plausible?

Q1:

Don't they have accountants?

Didn't they ever compute balances of their accounts?

Criminal mismanagement?

One commentator Rick Falkvinge - the founder of the first Pirate Party- wrote:

"It would be absolutely impossible to not notice the loss of one billion dollars from company assets, unless you were actively cooking the books to hide a gaping hole where a billion should have been.

As it turns out, it seems such fraudulent cooking is exactly what has taken place".

http://falkvinge.net/2014/02/25/gox-goes-belly-up-after-losing-a-billion-dollars-without-noticing-blames-fault-in-corporate-bookkeeping-protocol/

Inside Job?

Falkvinge calls it "inside job".

http://falkvinge.net/2014/02/28/the-gox-crater-crowd-detectives-reveal-billion-dollar-heist-as-inside-job/

- **Missing money was missing for a long time?**
  - **Some since 2011? Maybe.**
  - **They were apparently hiding the truth for a long time.**
- **For example, it appears that already in May 2013:**
  "customer coins were fraudulently sold to cover Gox liabilities"
- 20 Oct 2013: first report of a withdrawal from MtGox that didn't go through.
- their reserves started to be insufficient at many occasions later ,
  always claimed on technical problems, panic was avoided.

04 Feb 2013: first report of 38 million dollars missing at Gox.

Fake Pretext of Malleability?

- *February 7, 2014* – Gox shuts down bitcoin withdrawals entirely, blaming problems with the bitcoin protocol as such, so-called “transaction malleability attacks”. This is immediately identitied as technical bullshit by a number of heavy names of bitcoin. Gox promises an update on February 10.

- *February 9, 2014* – First recorded event of an actual transaction malleability attack against the blockchain, two days *after* Gox had been blaming such attacks for all problems up until now.
- This contrasts with the fact that failed withdrawals occurred as early as 20 Oct 2013.

Falkvinge also claims: bad code management.

- Signature malleability was apparently fixed in bitcoin 0.8.X except… at MtGox.

http://falkvinge.net/2014/02/11/the-embarrassing-fact-mtgox-left-out-of-their-press-release/

Karpeles at Fault?

Many people believe that the CEO of Mt Gox Karpeles is a fraudster and this has been going on for a very long time. Examples:

- Jesse Powell said that
  “There’s nothing to indicate that Empty Gox was ever solvent”.

- Bryce Weiner claims even that:
  given the blockchain data,
  the bulk of what Karpeles stole
  was prior to the move of the 424k BTC in 2011

https://twitter.com/BryceWeiner/status/438932232544059392

Fake Pretext?

More:

http://falkvinge.net/2014/02/11/the-embarrassing-fact-mtgox-left-out-of-their-press-release/

Mt Gox blamed hackers for changing transactions on the fly but…

MtGox were creating **invalid** transaction records for some small but significant portion of their bitcoin withdrawals.

Apparently hackers? or miners? (for some or part of these problems) were just FIXING what MtGox did wrong… Wicked.…

Malleability

Source: https://en.bitcoin.it/wiki/Transaction_Malleability

Facts: Transaction Malleability

“While transactions are signed, the signature does not currently cover all the data in a transaction that is hashed to create the transaction hash. Thus while uncommon it is possible for a node on the network to change a transaction you send in such a way that the hash is invalidated. Note that this just changes the hash, the output of the transaction remains the same and the bitcoins will go to their intended recipient. “

Malleability

Source: https://en.bitcoin.it/wiki/Transaction_Malleability

Contd.

[…] However this does mean that, for instance, it is not safe to accept a chain of unconfirmed transactions under any circumstance because the later transactions will depend on the hashes of the previous transactions, and those hashes can be changed until they are confirmed in a block (and potentially even after a confirmation if the block chain is reorganized).

[…] In addition clients must always actively scan for transactions to them; assuming a txout exists because the client created it previously is unsafe."

The last piece is obvious, old news

Malleability 1

Source: https://en.bitcoin.it/wiki/Transaction_Malleability

OpenSSL is at fault?

“Signature Malleability

1. The first form of malleability is in the signatures themselves.
   - Each signature has exactly one DER-encoded ASN.1 octet representation, but OpenSSL does not enforce this, and as long as a signature isn't horribly malformed, it will be accepted.

2. In addition for every ECDSA signature (r,s), the signature (r, -s (mod N)) is a valid signature of the same message.

Efforts are underway to first make Bitcoin nodes not relay non-standard signatures, and eventually disallow them from being included in new blocks entirely.

[…]

Malleability 2

This what we discover by inspection: Valid ScriptSig: both signatures are valid!

```
30460221 009e0339f72c793a89e664a8a932df073962a3f84eda0bd9e02084a6a9567f75aa
  0221 00bd9cbaca2e5ec195751efdfac164b76250b1e21302e51ca86dd7ebd7020cdc0601
30440220 9e0339f72c793a89e664a8a932df073962a3f84eda0bd9e02084a6a9567f75aa
0220 bd9cbaca2e5ec195751efdfac164b76250b1e21302e51ca86dd7ebd7020cdc0601
```

Length can also be 1F, 20, 21 and even 22: these examples are REALLY FOUND in the blockchain:

```
3046 0222 00008bb0e67238e0fe2747270389753973df50f9eeb7b5b182ac575f13c967bb2794
      0220 05961771ca85877836a2bb24e43ea36b796dfdfd4d017d95a070afebdc537e72 01
3043 021f fa0e5ccffc990924c1dc6d0acda81c26c1ee96e1af724d3d9fb8434de6c95e
      0220 98ee7071fa90d5c453a5b1d8790de4995a0995ad8394f1e79206d01f29be852d01
```

I think bitcoin developers could be easily accused of ALLOWING THIS MALLEABILITY and forcing EVERYBODY to accept it, like a virus(!), because these things are so frequent in the blockchain….

REMARK: a multiple of p could also be added to any of these numbers, however this is NOT accepted by bool CPubKey::Verify(uint256 &, std::vector<u8>& vchSig) in Satoshi code

r,-s does NOT work either, checked with 0.8.5.

Malleability 3

[…]

“scriptSig Malleability

The signature algorithm used in Bitcoin does not sign any of the scriptSig to create the signature. While signing the whole scriptSig would be impossible - the signature would be signing itself - this does mean that additional data can be added such that it will be pushed on the stack prior to the required signatures and public keys. Similarly OP_DROP can be added to leave the stack exactly as before prior to scriptPubKey execution.

Preventing scriptSig malleability is being considered as well. Currently transactions with anything other than data push operations in their scriptSig are considered non-standard and are not relayed, and eventually this rule may extend to enforcing that the stack have exactly one item after execution. However doing that may interfere with later extensions to Bitcoin.”

Bitcoin Developers:

- This is weird. A transaction isn’t quite identified by its transaction ID, but only by its contents. So somebody can replay your transaction simply by giving it a different number, and then hoping to win the race.
- I think it’s reasonable to consider this a protocol flaw, and it’s something I didn’t know about, but how can mt. gox proclaim ignorance when people in the know have been talking about it for 3 years? Shouldn’t mt. gox be in the know, too?
- This is also another case where I’m confused how a bitcoin operator can even have the problem they’re having. I’m not the world’s greatest accountant, but I’m also not running a money service. When a bad customer performs the above trick and claims the transaction failed, why doesn’t mt. gox check the account balance (blockchain) and see that the money has in fact been transferred? mt. gox should know how much money is in their wallet, right? They can compare the amount they have now, the amount they had an hour ago, and the total of transfers they believe occurred.

Source: https://lobste.rs/s/m4p6nl

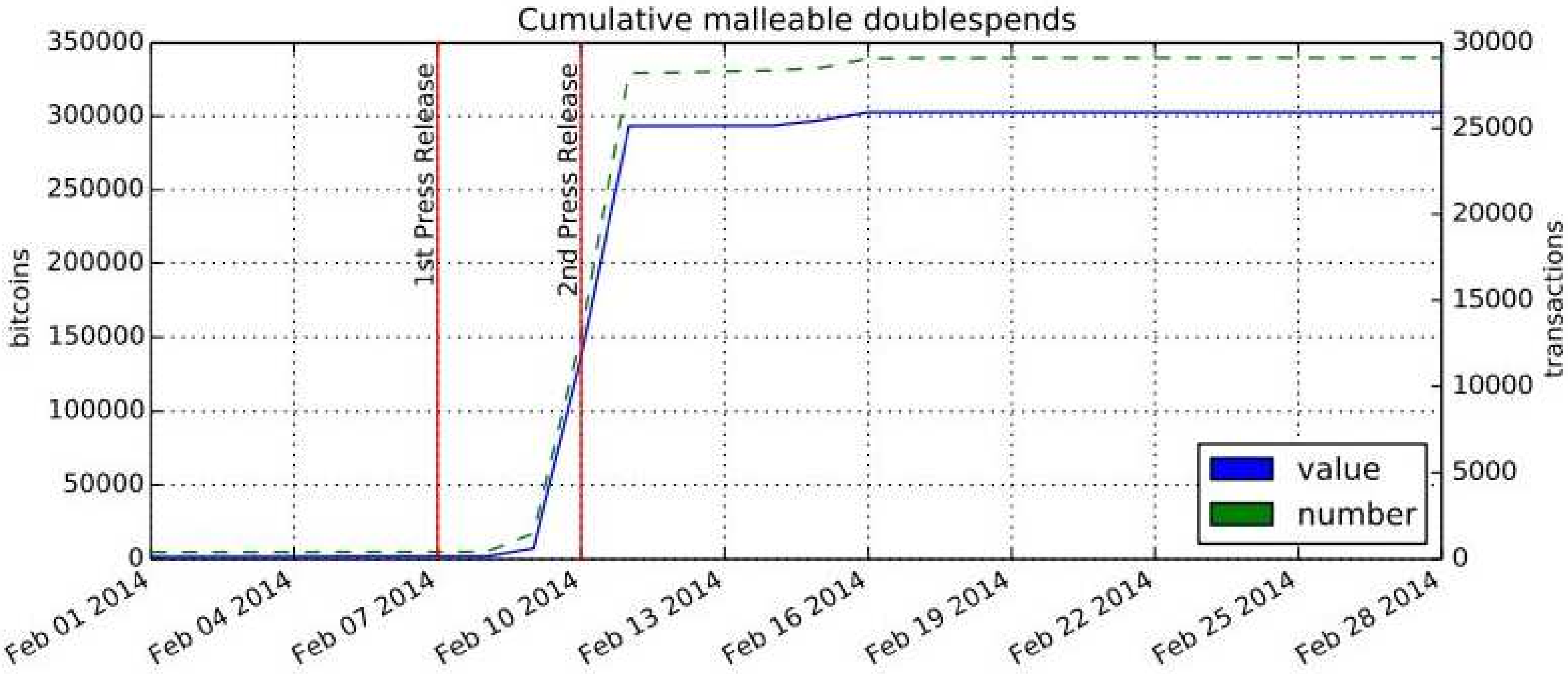
Bitcoin Developers:

- This is weird. A transaction isn't quite identified by its transaction ID, but only by its contents. So somebody can replay your transaction simply by giving it a different number, and then hoping to win the race.
- I think it's reasonable to consider this a protocol flaw, and it's something I didn't know about, but how can mt. gox proclaim ignorance when people in the know have been talking about it for 3 years? Shouldn't mt. gox be in the know, too?
- This is also another case where I'm confused how a bitcoin operator can even have the problem they're having. I'm not the world's greatest accountant, but I'm also not running a money service. When a bad customer performs the above trick and claims the transaction failed, why doesn't mt. gox check the account balance (blockchain) and see that the money has in fact been transferred? mt. gox should know how much money is in their wallet, right? They can compare the amount they have now, the amount they had an hour ago, and the total of transfers they believe occurred.

Source: https://lobste.rs/s/m4p6nl

Thefts

slides to be updated soon…


Cumulative malleable doublespends
bitcoins
transactions
1st Press Release
2nd Press Release
value
number
Feb 01 2014
Feb 04 2014
Feb 07 2014
Feb 10 2014
Feb 13 2014
Feb 16 2014
Feb 19 2014
Feb 22 2014
Feb 25 2014
Feb 28 2014