Blockchain unchained

Bitcoin

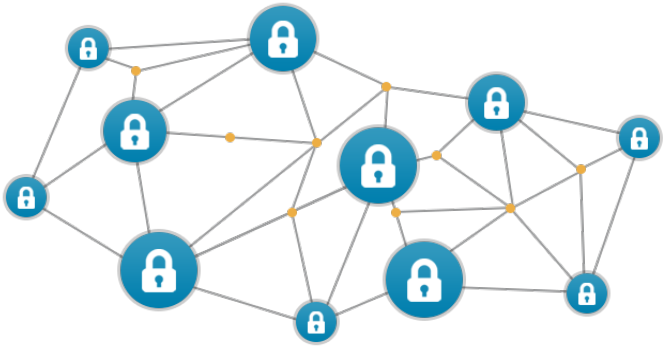
If you can't explain it simply, you don't understand it well.

by TYR

application
cryptocurrency

---

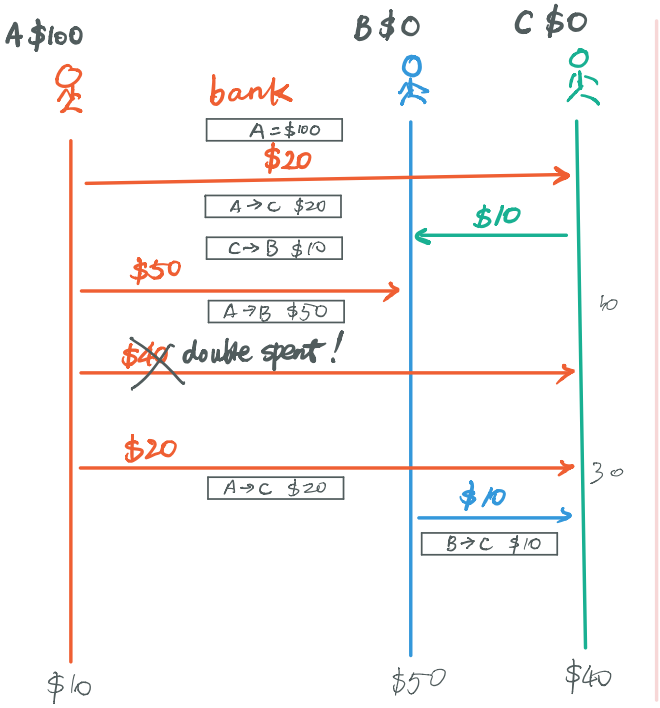
infrastructure
blockchain

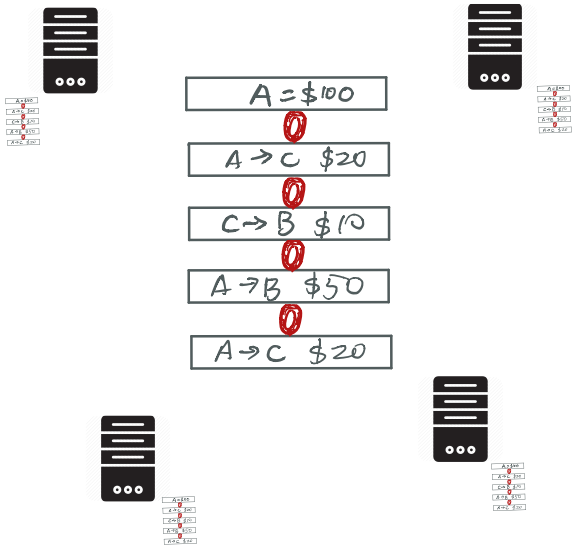


Money Transfer

① takes too long
② fee too high

Ledger





The history of the transaction is Currency.

Consensus ?

Synchronization ?

Capacity ?

privacy ?

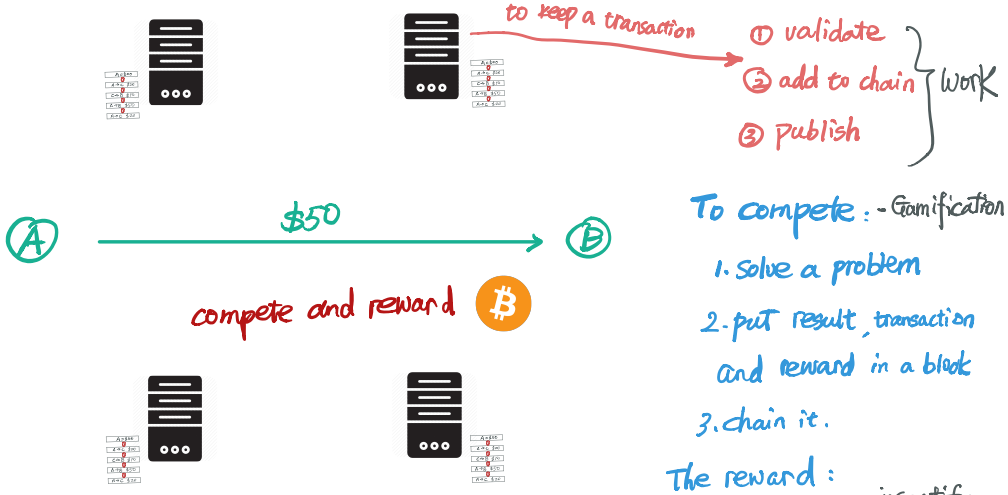
Security ?

Incentivity ?

Open Ledger

- distributed
  - decentralized
  - not easy to attack
- faster
- less cost (hopefully)

Why other people in the network wants to keep you transaction?



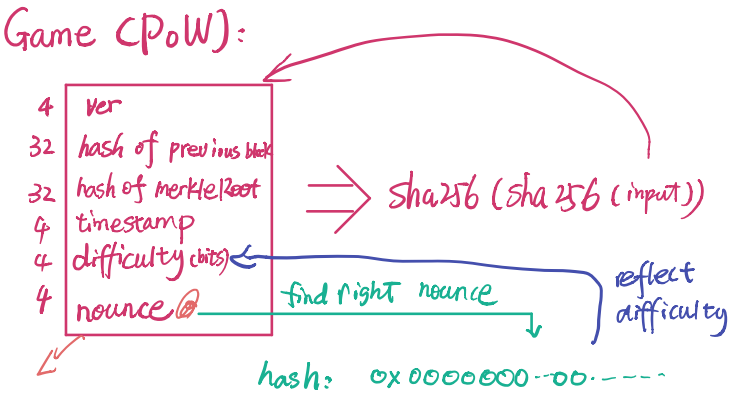
To compete: - Gamification

1. Solve a problem
2. put result, transaction and reward in a block
3. chain it.

The reward:

฿50 - incentify system

bytes	
4	magic : 0xd9b4bef9
4	length
4	ver 0x1
32	sha256 (previous block
32	merkleRoot
4	timestamp : creation time of the block
4	difficulty : 3,007,383,866,459.732
4	nounce: the problem to solve
	transaction count
	transactions



what we learned:

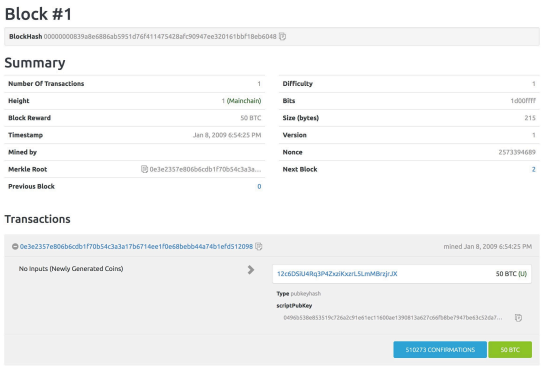
1. to find a "nounce" is hard (random)
2. by tuning difficulty, the system can limit the generation of a block to close to 10min.
3. merkle Tree + bloomfilter made wallet applicable
4. chaining + PoW greatly reduced the chance of "do evil thing".
5. "incentive" encourage nodes to be good.

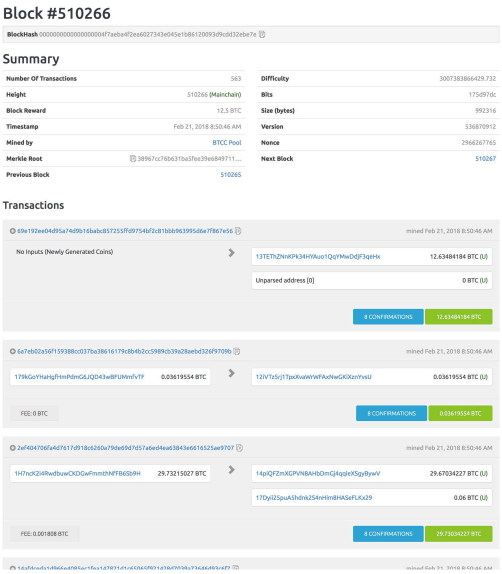
What is a block?

- a data structure contains transactions
- linked with previous work
- blocks in longest chain excel

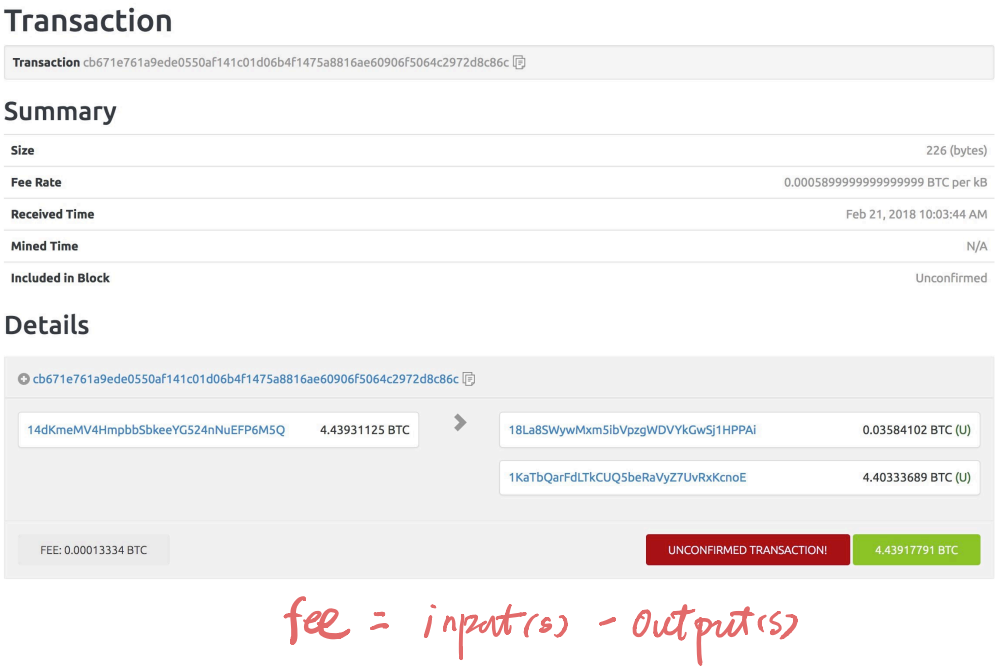
Genesis Block

Recent Block

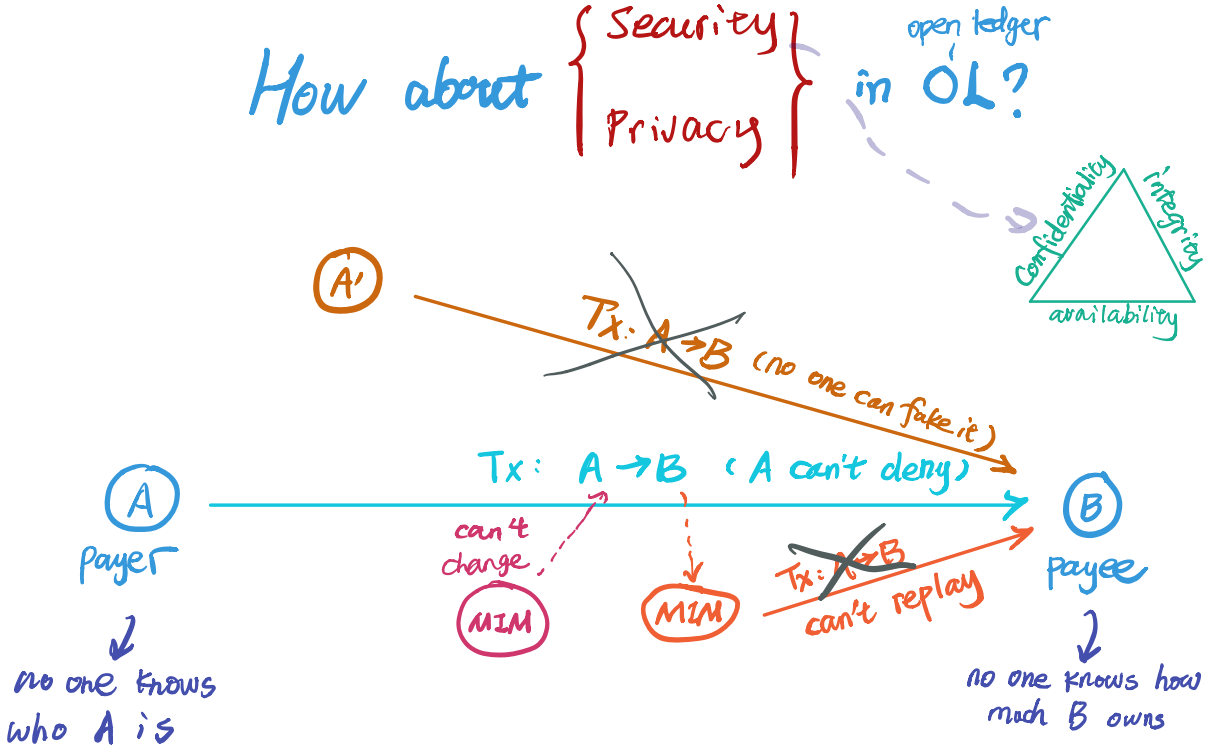




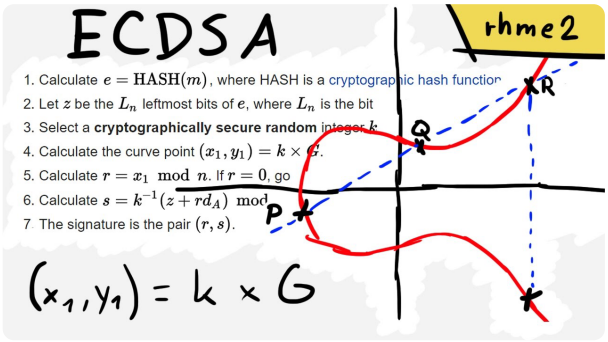
A Tx



fee = input(s) - output(s)



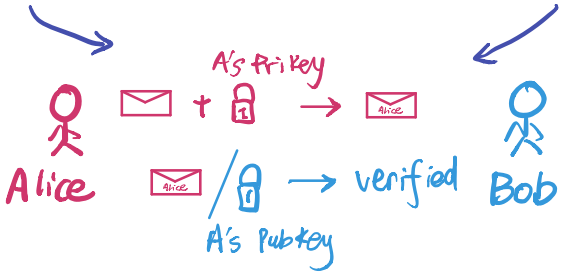
what do we have?



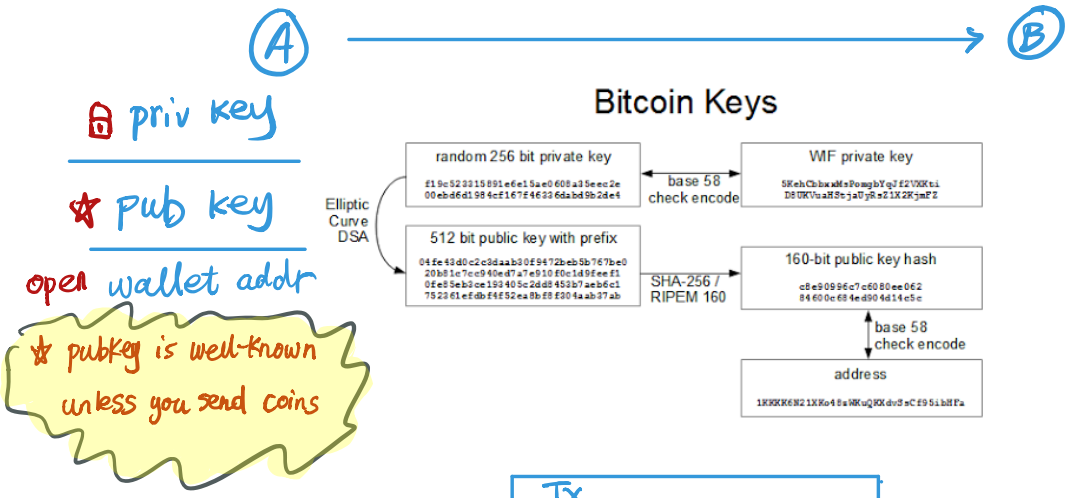
RSA

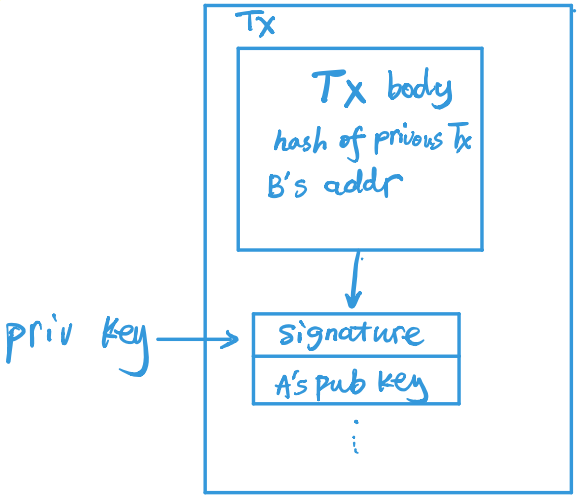
- Select two large prime numbers p, q
- Compute
  n = p × q
  v = (p-1) × (q-1)
- Select small odd integer k relatively prime to v
  gcd(k, v) = 1
- Compute d such that
  (d × k)%v = (k × d)%v = 1
- Public key is (k, n)
- Private key is (d, n)

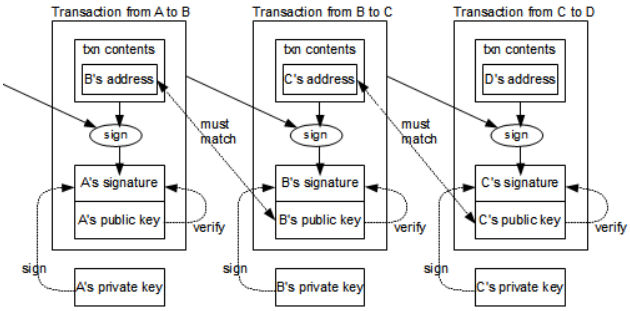
- example
  - p = 11
  - q = 29
  - n = 319
  - v = 280
  - k = 3
  - d = 187
- public key
  (3, 319)
- private key
  (187, 319)



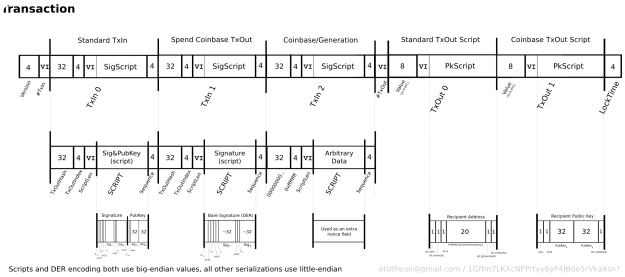
What does bitcoin do?







More on Tx signing



more on Tx structure

It's a VM

What about security of ecosystem?

- bad nodes
- bad nodes > 51% of computing power
- chain is too big to fit a commodity node (availability)

SPV ← Simplified Payment Verification

- Threat from Quantum computing

Shor's algo

Grover's algo

basics of RSA ←

45424339 1453

1299721 × 349493

given [a, b, ----]

$O(2^{K/2}) \rightarrow O(K^3)$

$O(N) \rightarrow O(\sqrt{N})$

could be used in ECDSA

Any problem

Crack 256 bit priv key : classic 340 T T T (T = Trillion)

a few hundred Million	18 million T

hash of pub key is important!

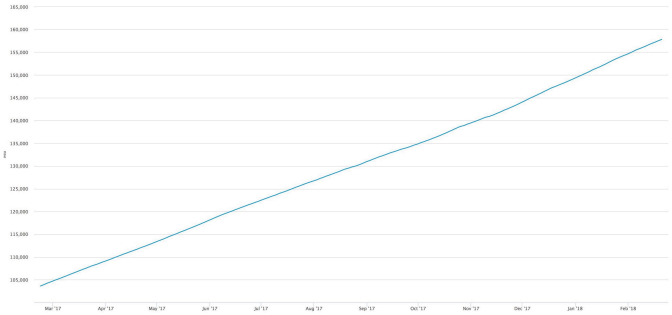
How about capacity?

- 1 M block
  added 2010
- 10 Min a block

(6×24×365×9 = 474336) latest : 510438

- Wallet just need

  headers and related Merkle Tree path



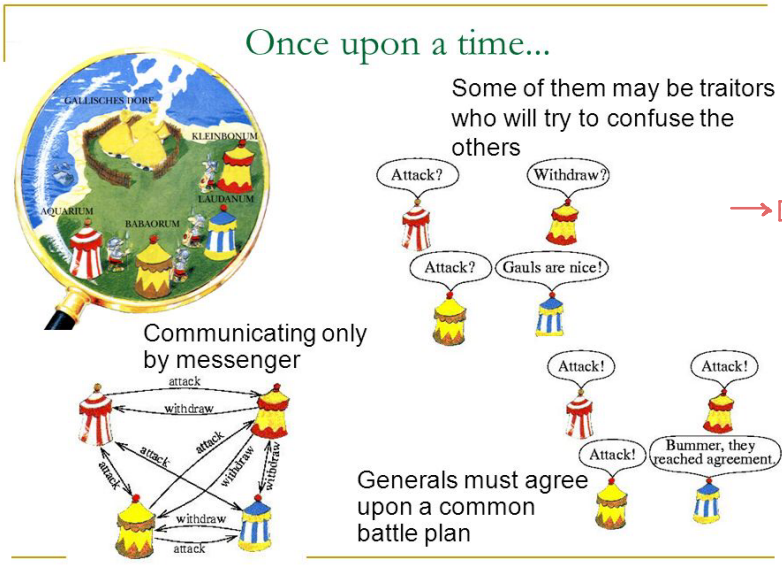
why not bigger?

1MB × 8bit × 7peers/30sec = 1.86Mb/s

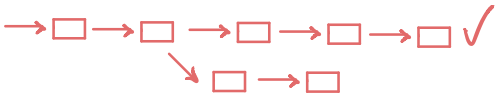
8MB × 8bit × 7peers/30sec = 15Mb/s

upload speed

How { Sync / Consensus } work?



PoW



The pictures are taken from: R. Goscinny and A. Uderzo, Asterix and Latraviata.

How miner work ?

(TBD)

Genius Design

- Chained ledger
- Proof-of-Work
- incentification
- Merkle-Tree supported SPV
- Hashed public key (hide the flaw of ECDSA)
- 21,000,000 total coins
  = 2.1 quadrillion satoshis $\approx 2^{50.9} < 2^{53}$
- base58 - why why why!
  - human readable (0, O, I, l striped)

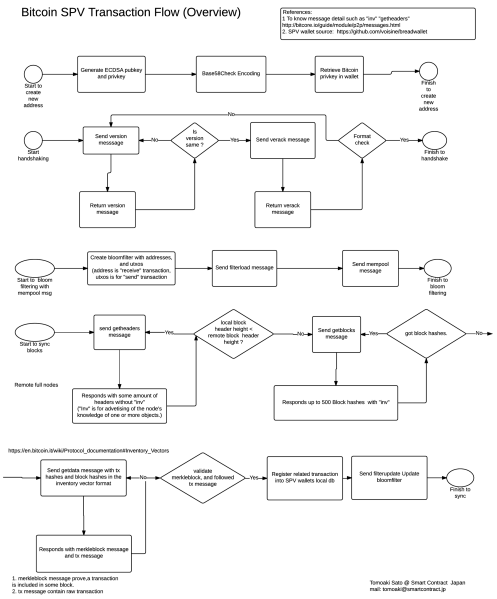
What can we learn for sw design?

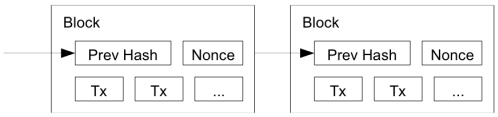
- ask questions
- ask great questions
- ask great questions that future oriented
- Think it through and connect the dots

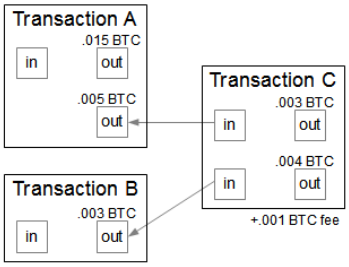
Thank!

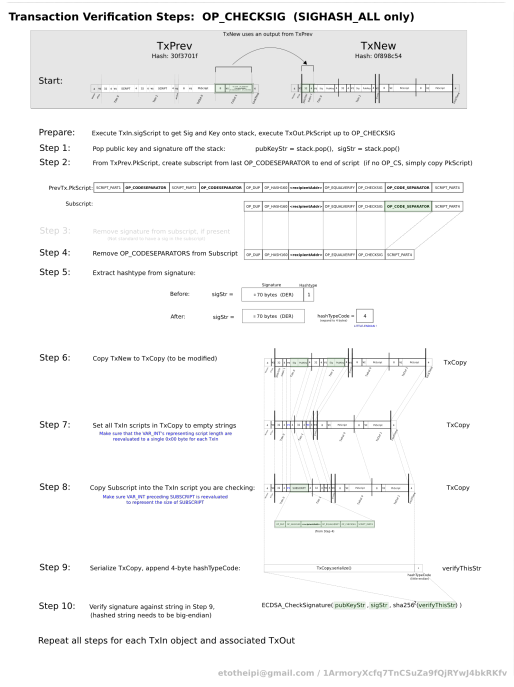
程序人生公众号

unused


Bitcoin SPV Transaction Flow (Overview)
References:
1. To know message detail such as "inv" "getheaders"
http://chizomi.is/guide/medusa/js2p/messages.html
2. SPV wallet source : https://github.com/nvisinet/breadwallet
Start to create new address
Generate ECDSA pubkey and privkey
Base58Check Encoding
Retrieve Bitcoin privkey in wallet
Finish to create new address
Start handshaking
Send version message
Is version same ?
Send verack message
Format check
Finish to handshake
Return version message
Return verack message
Start to sync blocks
send getheaders message
local block header height < remote block header height ?
Send getblocks message
got block hashes
Remote full nodes
Responds with some amount of headers without "inv" ("inv" is for advetising of the node's knowledge of one or more objects.)
Responds up to 500 Block hashes with "inv"
https://en.bitcoin.it/wiki/Protocol_documentation#Inventory_Vectors
Send getdata message with tx hashes and block hashes in the inventory vector format
validate merkleblock, and followed tx message
Register related transaction into SPV wallets local db
Send filterupdate Update bloomfilter
Finish to sync
Responds with merkleblock message and tx message
1. merkleblock message prove a transaction is included in some block.
2. tx message contain raw transaction
Tomoaki Sato @ Smart Contract Japan
mail: tomoaki@smartcontract.jp


Block
Prev Hash
Nonce
Tx
Tx
...
Block
Prev Hash
Nonce
Tx
Tx
...


Transaction A
.015 BTC
in
out
.005 BTC
out
Transaction C
.003 BTC
in
out
.004 BTC
in
out
+.001 BTC fee
Transaction B
.003 BTC
in
out


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:
Subscript:
Step 3: Remove signature from subscript, if present
Step 4: Remove OP_CODESEPARATORS from Subscript
Step 5: Extract hashtype from signature:
Before: sigStr =
After: sigStr =
Step 6: Copy TxNew to TxCopy (to be modified)
TxCopy
Step 7: Set all TxIn scripts in TxCopy to empty strings
TxCopy
Step 8: Copy Subscript into the TxIn script you are checking:
TxCopy
Step 9: Serialize TxCopy, append 4-byte hashTypeCode:
verifyThisStr
Step 10: Verify signature against string in Step 9, (hashed string needs to be big-endian)
ECDSA_CheckSignature(pubKeyStr , sigStr , sha256(verifyThisStr))
Repeat all steps for each TxIn object and associated TxOut
etotheipi@gmail.com / 1ArmoryXcfq7TnCSuZa9fQjRYwJ4bkRKfv