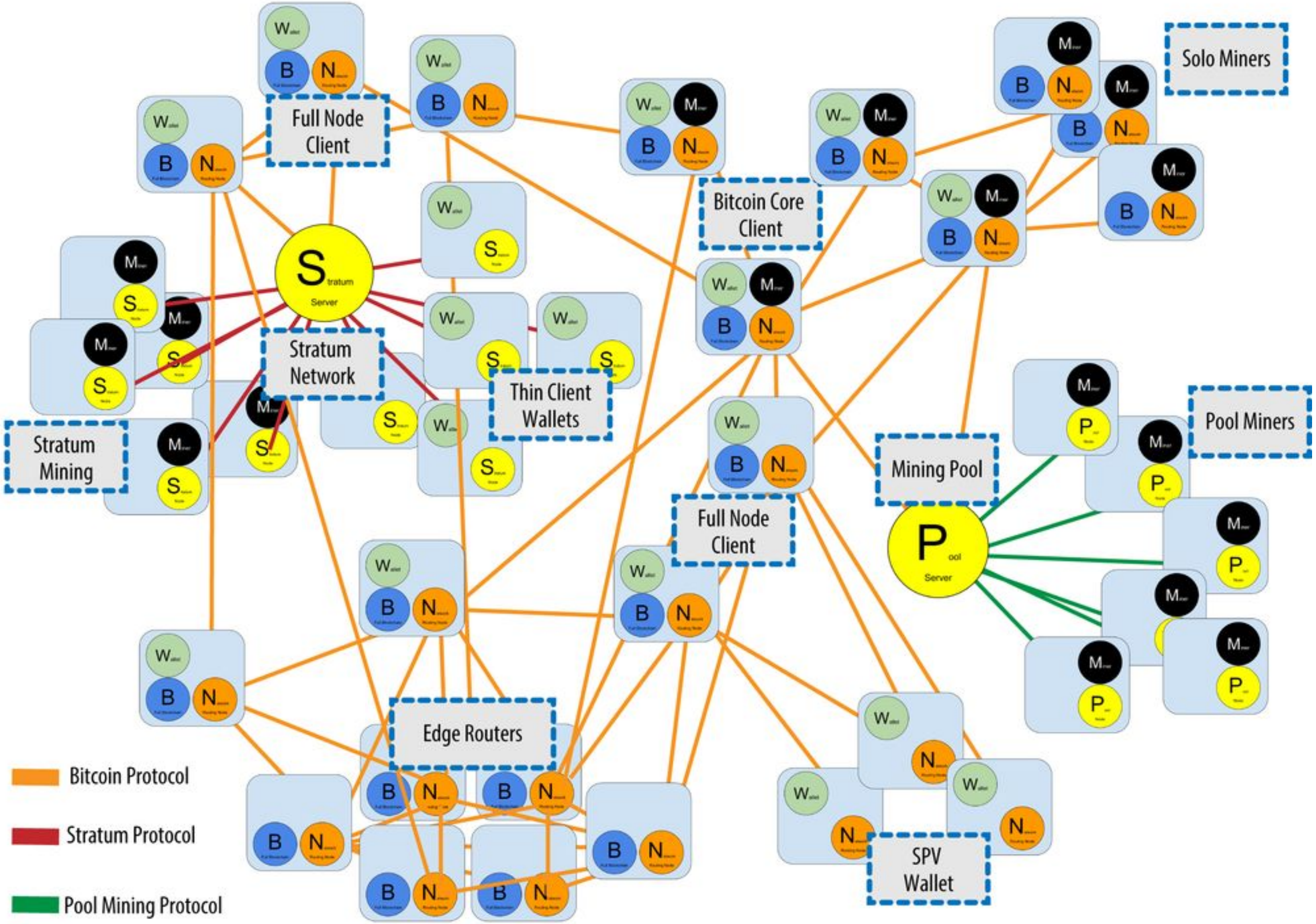
Cryptocurrencies and Go

Afanasev Stanislav
@superstas88
15.02.2017

Agenda

- Theory
- Practice

Slides



https://bitcoin.org/bitcoin.pdf- Bitcoin: A P2P Electronic Cash System

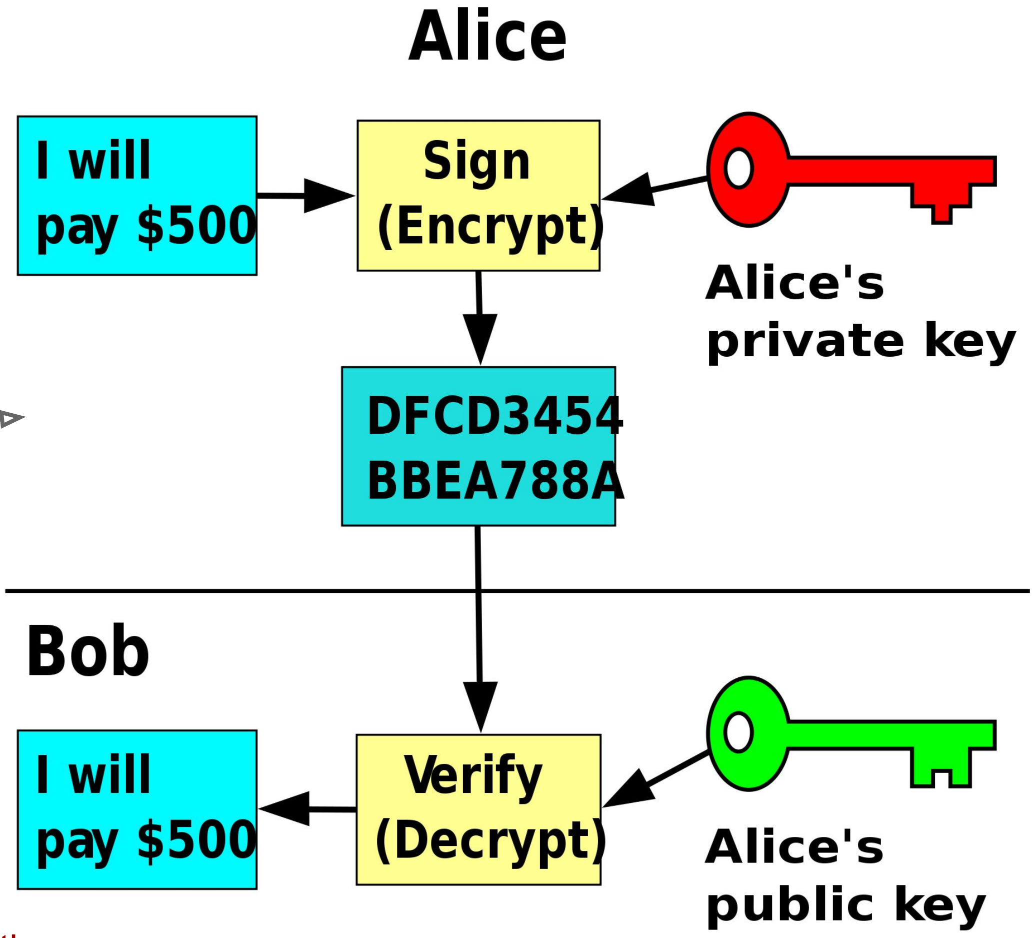
Part 1

Theory

Cryptographic keys

- ECDSA (SECP256k1)

It's important to understand



Alice

I will pay $500

Sign (Encrypt)

Alice's private key

DFCD3454 BBEA788A

Bob

I will pay $500

Verify (Decrypt)

Alice's public key

[Why not RSA?]

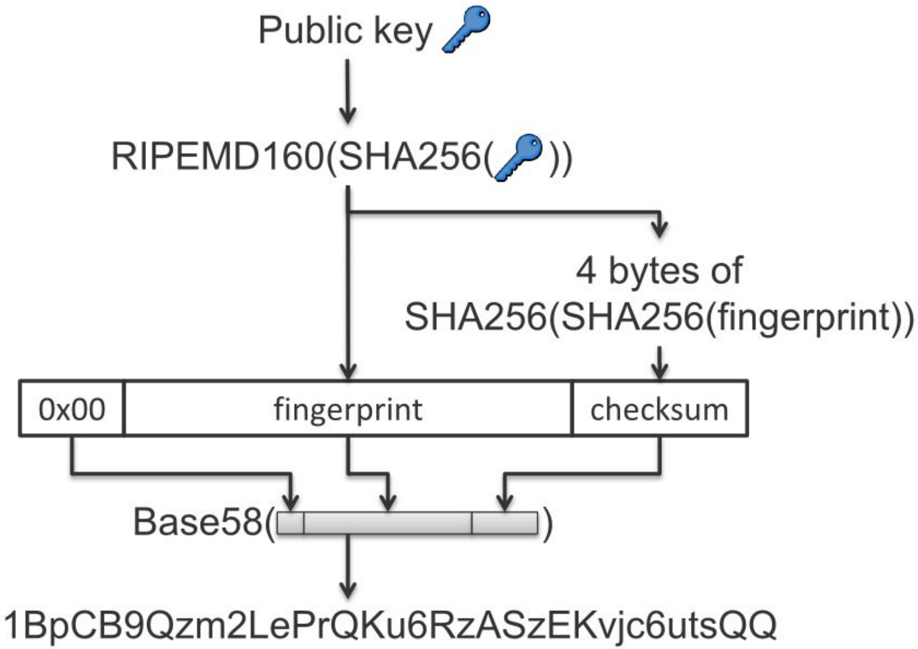
[Elliptic Curve Cryptography: a gentle introduction]

Addresses

- RIPEMD160
- SHA256
- Base58 (“0OIl” are not used)

Base58:
1EiK2ZgptmS5HZ2hDnQvEXC93L1JSnbttY

Base64:
AJZpvN5wxZC7duwu17DB2WR3Lq3l04yU2Q==

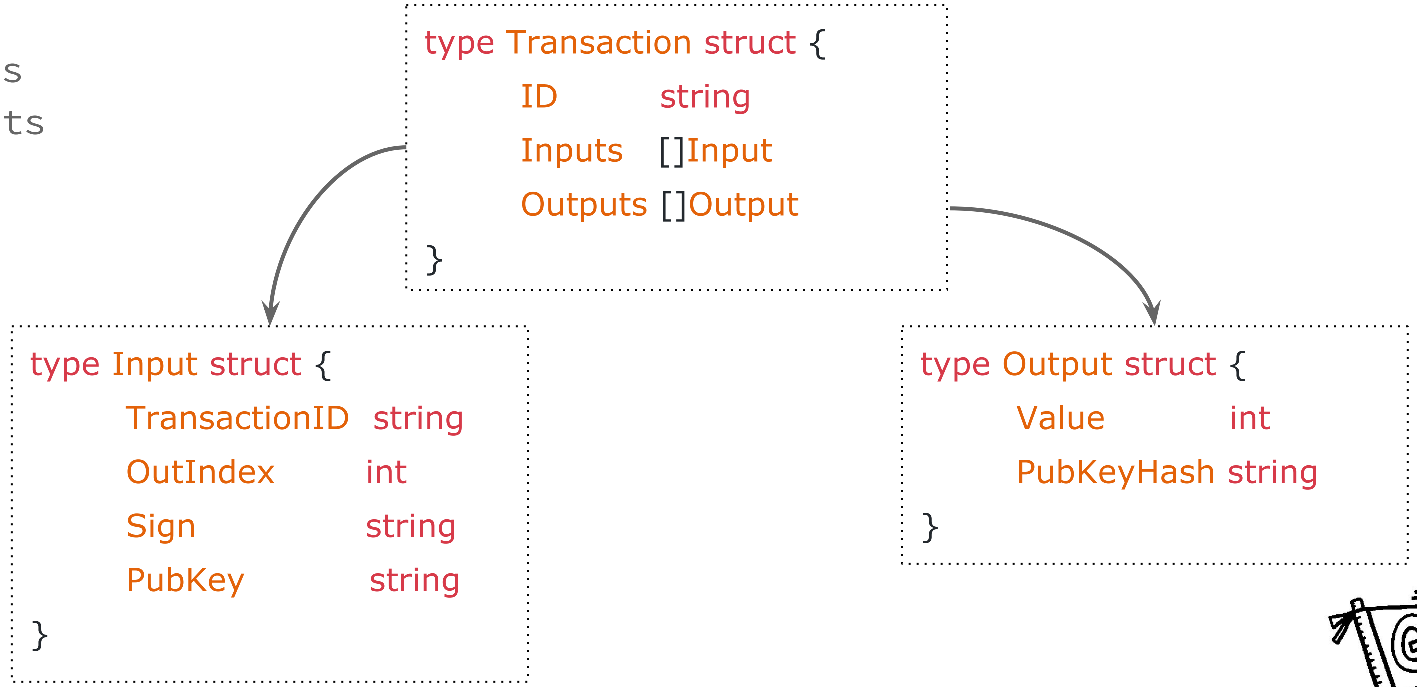


Just compare

[Address](#)
[Base58Check_encoding](#)
[Why does Bitcoin use two hash functions (SHA-256 and RIPEMD-160)?](#)

Transactions

- ID
- Inputs
- Outputs
- P2PKH

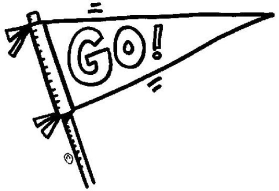


```
type Transaction struct {
        ID         string
        Inputs     []Input
        Outputs []Output
}
```

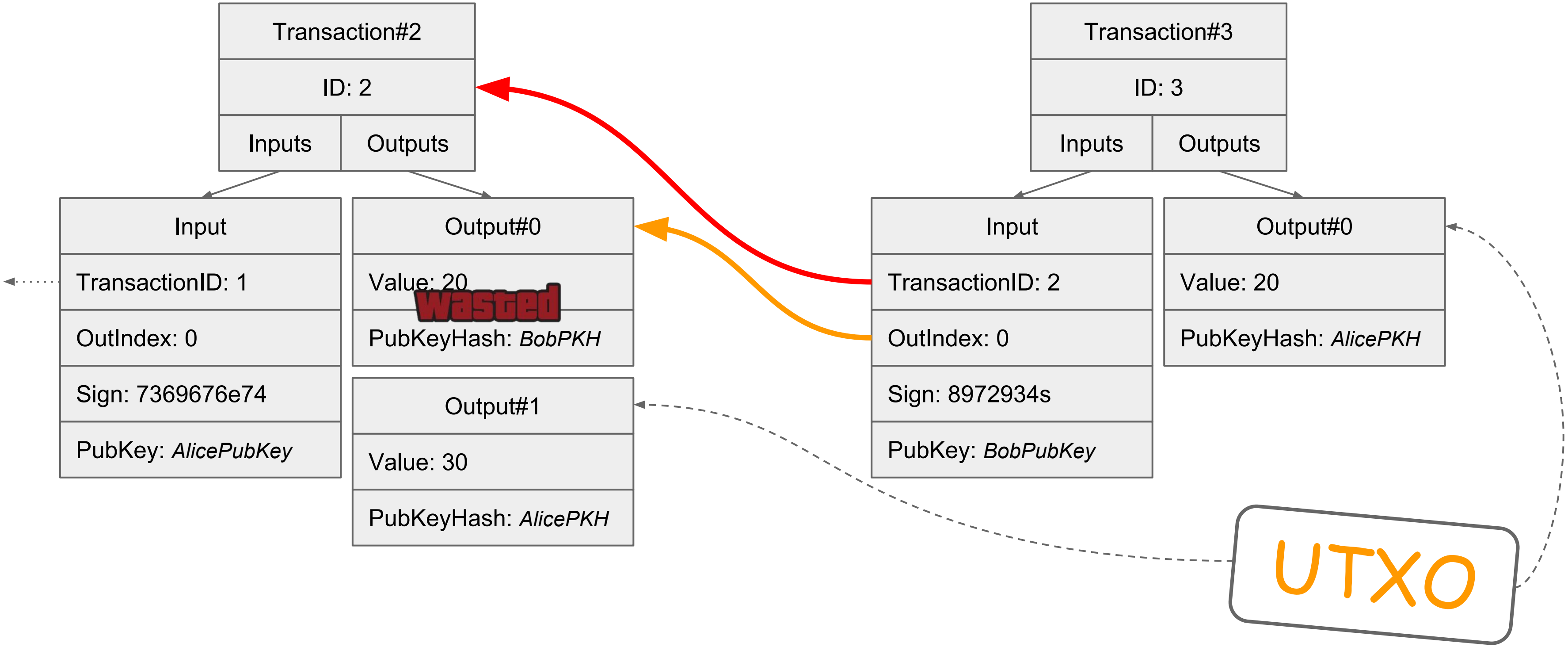
```
type Input struct {
        TransactionID  string
        OutIndex         int
        Sign                string
        PubKey             string
}
```

```
type Output struct {
        Value              int
        PubKeyHash string
}
```

https://en.bitcoin.it/wiki/Transaction
https://en.bitcoin.it/wiki/Script



Transactions. UTXO



Transaction#2

ID: 2

Inputs | Outputs

Input

TransactionID: 1

OutIndex: 0

Sign: 7369676e74

PubKey: *AlicePubKey*

Output#0

Value: 20

PubKeyHash: *BobPKH*

Output#1

Value: 30

PubKeyHash: *AlicePKH*

Transaction#3

ID: 3

Inputs | Outputs

Input

TransactionID: 2

OutIndex: 0

Sign: 8972934s

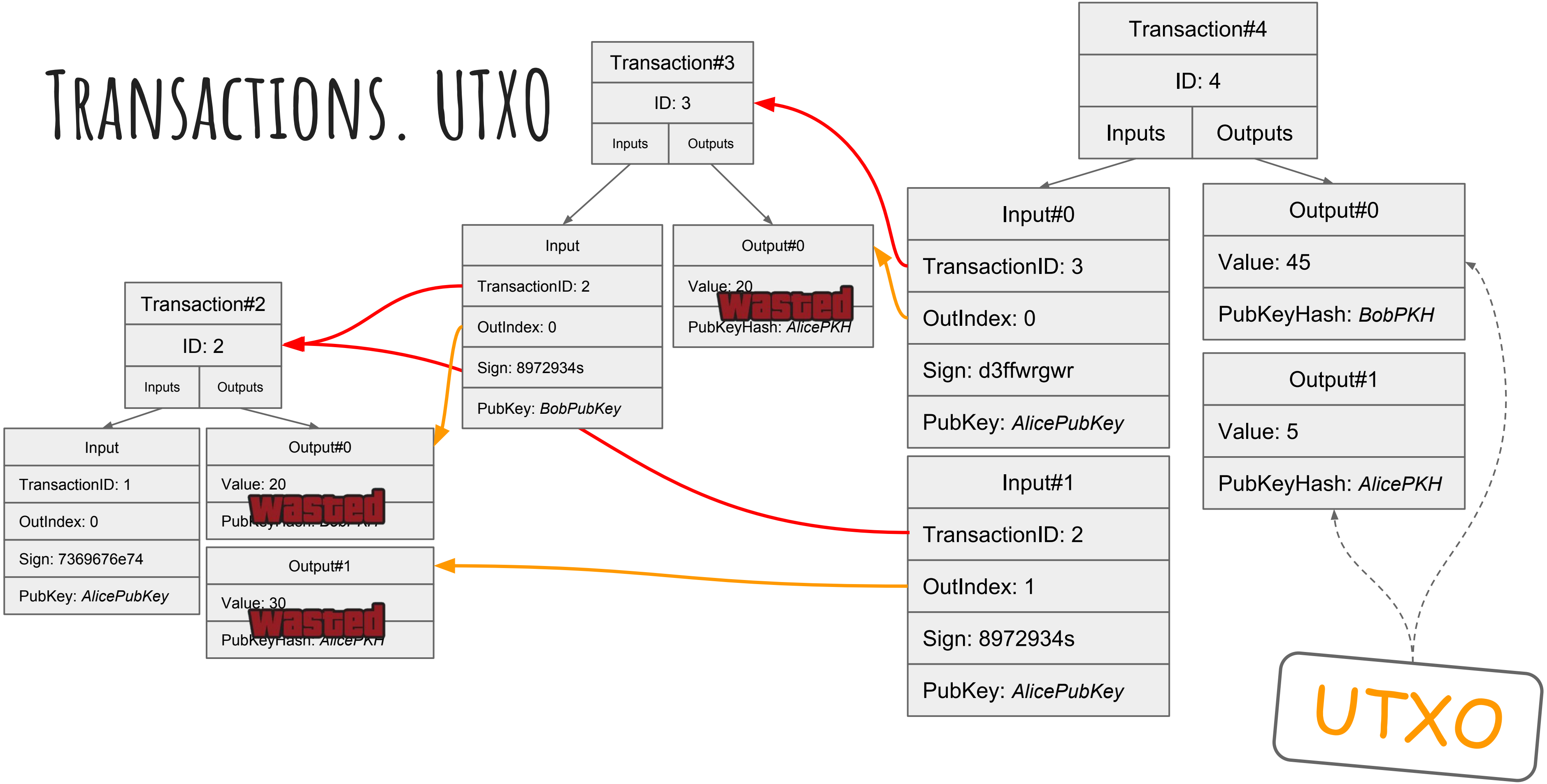
PubKey: *BobPubKey*

Output#0

Value: 20

PubKeyHash: *AlicePKH*

Transactions. UTXO



Transaction#3
ID: 3
Inputs | Outputs

Input
TransactionID: 2
OutIndex: 0
Sign: 8972934s
PubKey: *BobPubKey*

Output#0
Value: 20
PubKeyHash: *AlicePKH*

Transaction#4
ID: 4
Inputs | Outputs

Input#0
TransactionID: 3
OutIndex: 0
Sign: d3ffwrgwr
PubKey: *AlicePubKey*

Input#1
TransactionID: 2
OutIndex: 1
Sign: 8972934s
PubKey: *AlicePubKey*

Output#0
Value: 45
PubKeyHash: *BobPKH*

Output#1
Value: 5
PubKeyHash: *AlicePKH*

Transaction#2
ID: 2
Inputs | Outputs

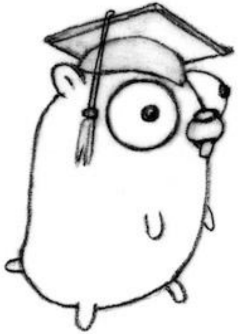
Input
TransactionID: 1
OutIndex: 0
Sign: 7369676e74
PubKey: *AlicePubKey*

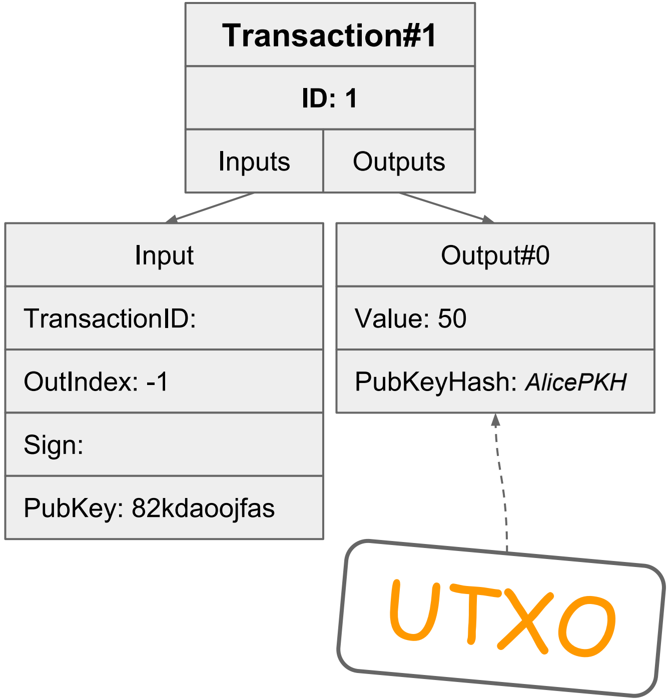
Output#0
Value: 20
PubKeyHash: *BobPKH*

Output#1
Value: 30
PubKeyHash: *AlicePKH*

Coinbase transaction

- Mining reward
- TxID = sha256(sha256(TX))





https://en.bitcoin.it/wiki/Controlled_supply

Blocks

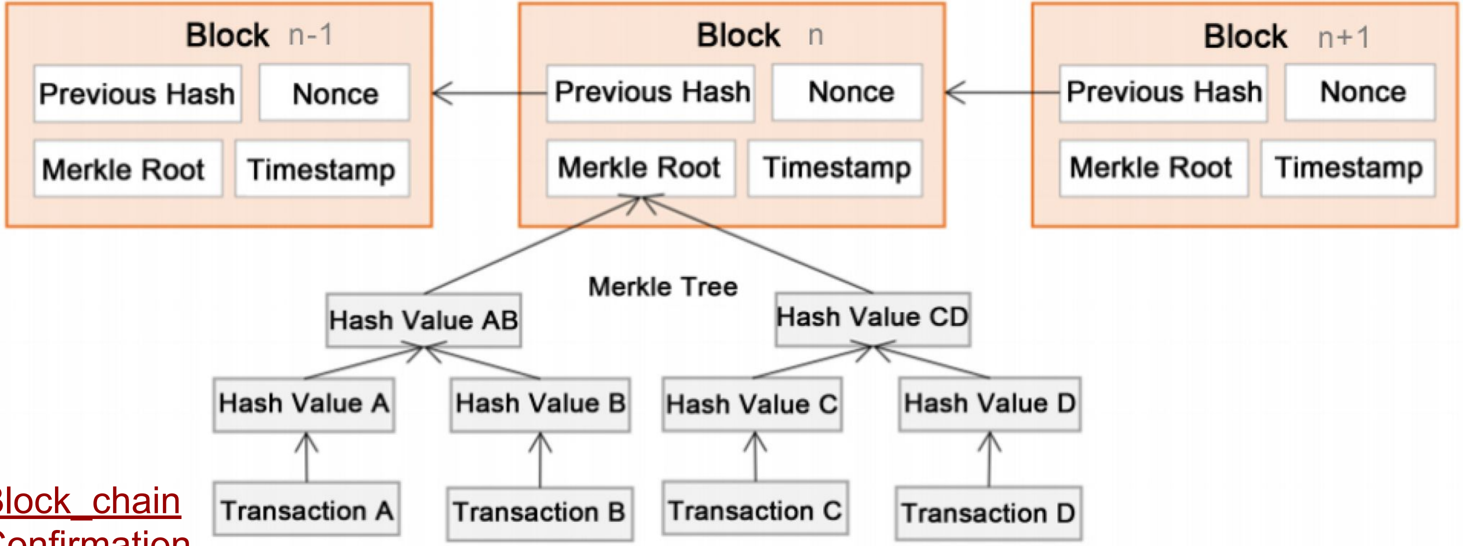
- Coinbase transaction
- Block.Hash = Sha256(Sha256(BlockHeader))
- Merkle tree
- Block size / Bits

```
type BlockHeader struct {
        PreviousBlockHash string
        MerkleRootHash    string
        Timestamp         int64
        Nonce             int
}

type Block struct {
        BlockHeader
        Hash          string
        Transactions  []Transaction
}
```

https://en.bitcoin.it/wiki/Protocol_documentation#Merkle_Trees
https://en.bitcoin.it/wiki/Protocol_documentation#Block_Headers

Blockchain

- Genesis block
- TX confirmations
- P2P



https://en.bitcoin.it/wiki/Block_chain
https://en.bitcoin.it/wiki/Confirmation
https://en.bitcoin.it/wiki/Genesis_block

Proof-of-Work (PoW)

- Time block
- Target
- Difficulty = MaxTarget (8 leading zeros) / Target

Mining

Round 1: **Sha256(Sha256(BlockHeader with Nonce=0)) < Target**
Round 2: **Sha256(Sha256(BlockHeader with Nonce=1)) < Target**
...
Round N: **Sha256(Sha256(BlockHeader with Nonce=N)) < Target**

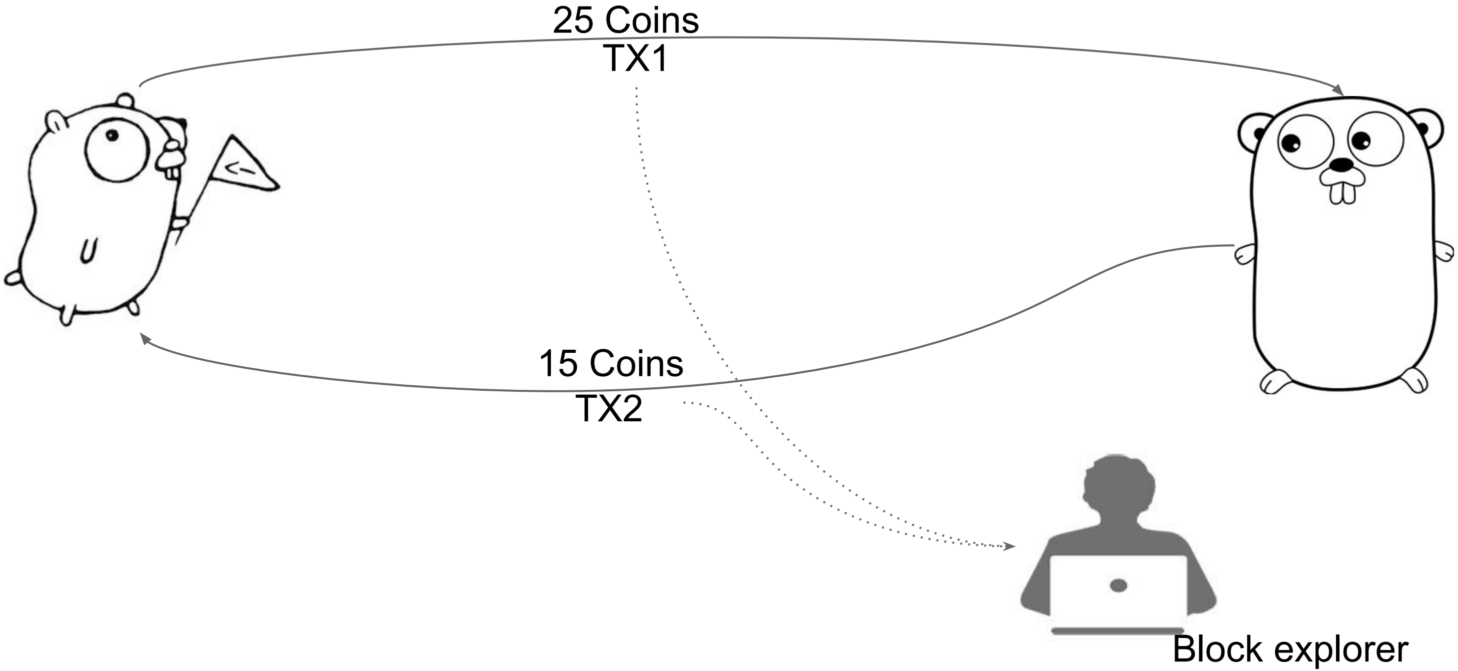
```
Hash   = 0x000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
Target = 0x00000000ffffffffffffffffffffffffffffffffffffffffffffffffffffffff
                   Hash < Target = We've found a hash!
```

https://en.bitcoin.it/wiki/Target
https://en.bitcoin.it/wiki/Consensus

Part 2

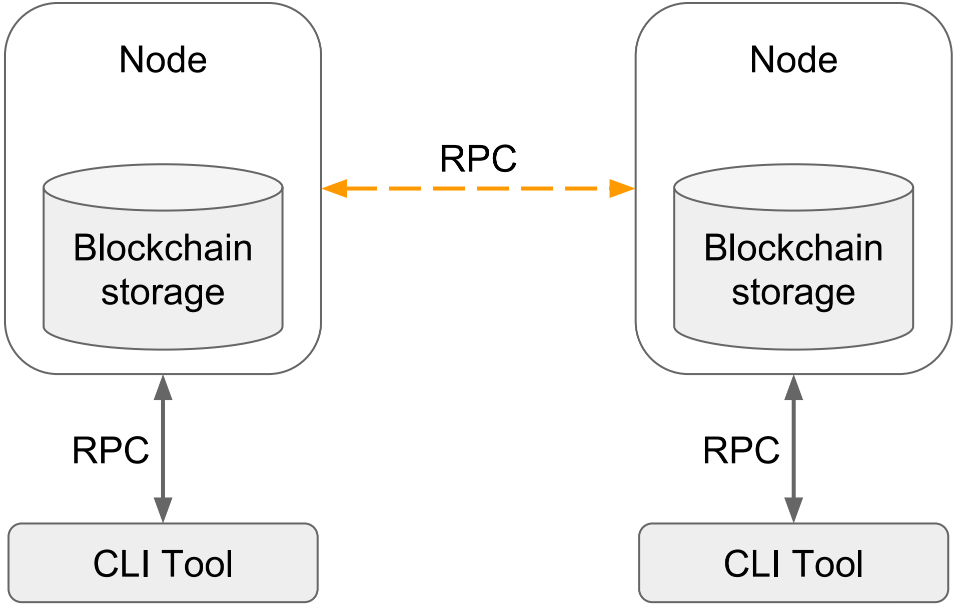
Practice

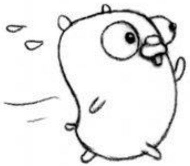
Let's send some coins...



Proof-of-concept. Todolist

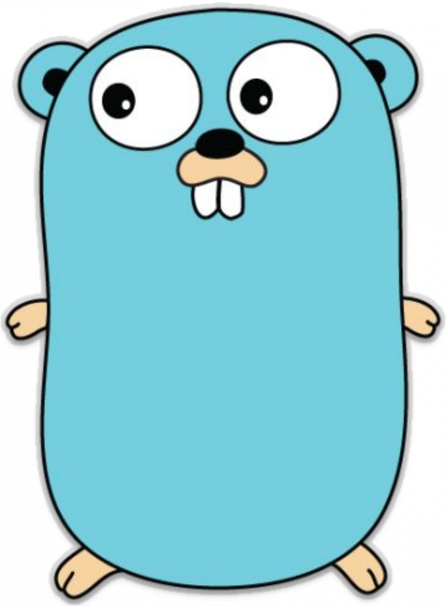
- Blockchain storage
- Business logic
  - Types
  - Wallet
  - Transactions
  - Blocks/Blockchain
  - PoW
  - Mempool
- Network layer
- Daemon / CLI modes
- Network discovery
- Block explorer





Storage

- Block, Transactions
- UTXOSet
- LevelDB
- BoltDB

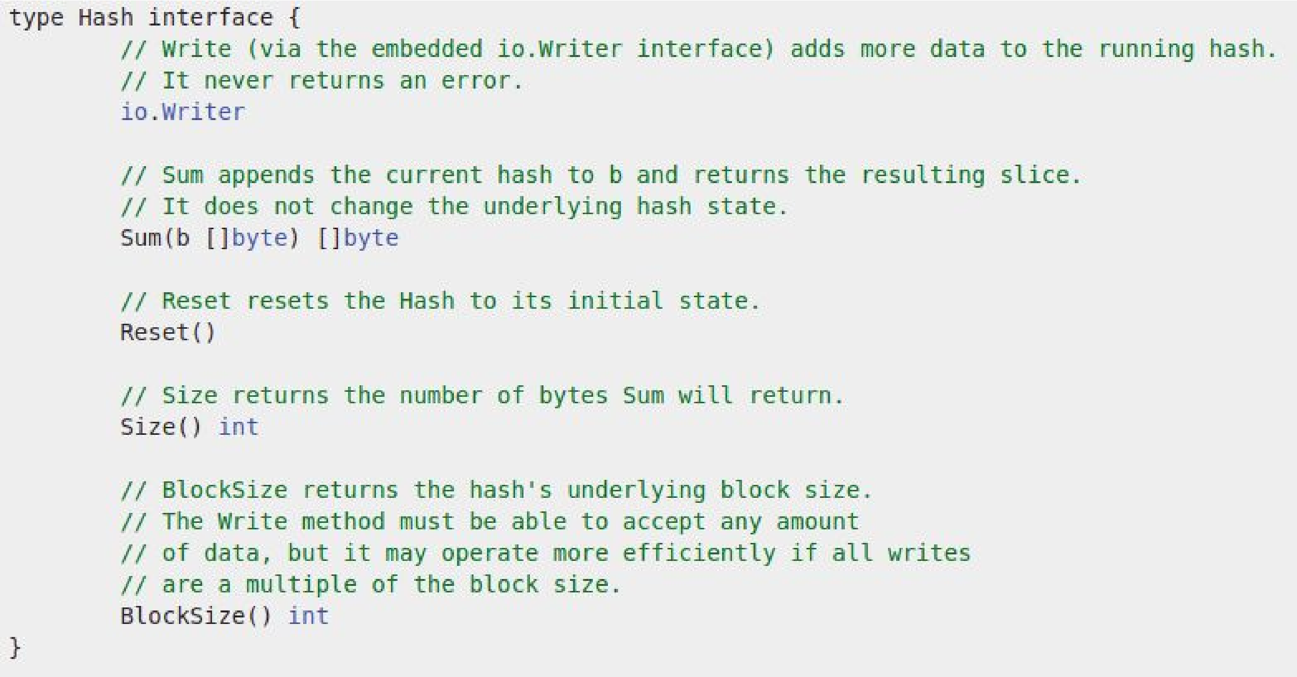


Why is Bitcoin Core using LevelDB instead of Redis or SQLite?

What are the keys used in the blockchain levelDB?

https://github.com/avelino/awesome-go#database

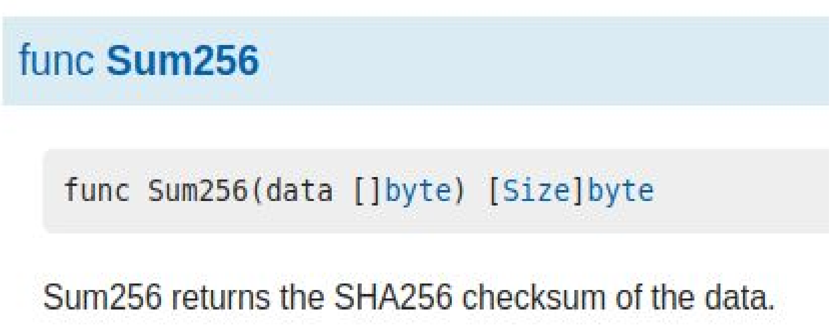
PKG/HASH



```
type Hash interface {
        // Write (via the embedded io.Writer interface) adds more data to the running hash.
        // It never returns an error.
        io.Writer

        // Sum appends the current hash to b and returns the resulting slice.
        // It does not change the underlying hash state.
        Sum(b []byte) []byte

        // Reset resets the Hash to its initial state.
        Reset()

        // Size returns the number of bytes Sum will return.
        Size() int

        // BlockSize returns the hash's underlying block size.
        // The Write method must be able to accept any amount
        // of data, but it may operate more efficiently if all writes
        // are a multiple of the block size.
        BlockSize() int
}
```

https://github.com/golang/go/wiki/Hashing

PKG/HASH



func **Sum256**

```
func Sum256(data []byte) [Size]byte
```

Sum256 returns the SHA256 checksum of the data.

```
func main() {
        blockHeader := []byte("blockHeader1")
        sha256.Sum256(sha256.Sum256(blockHeader)[:])
        // invalid operation sha256.Sum256(blockHeader)[:]
(slice of unaddressable value)
}
```

```
func DoubleHash(h hash.Hash, data []byte) []byte {
        h.Reset()
        h.Write(data)
        ch := h.Sum(nil)
        h.Reset()
        h.Write(ch)
        return h.Sum(nil)
}
```

```
func WrongDoubleHash(h hash.Hash, data []byte) []byte {
        h.Reset()
        h.Write(data)
        h.Write(h.Sum(nil))
        return h.Sum(nil)
}
```

https://github.com/golang/go/wiki/Hashing

MATH/BIG

```go
func main() {
	blockHash, _  := hex.DecodeString("000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f")
	targetHash, _ := hex.DecodeString("00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF")

	blockHashInt := big.NewInt(0).SetBytes(blockHash)
	targetHashInt := big.NewInt(0).SetBytes(targetHash)

	fmt.Printf("BlockHashInt: %s\n", blockHashInt.String())
	fmt.Printf("TargetHashInt: %s\n", targetHashInt.String())
	fmt.Printf("BlockHashInt < TargetHashInt: %v\n", blockHashInt.Cmp(targetHashInt) == -1)
}
// BlockHashInt: 10628944869218562084050143519444549580389464591454674019345556079
// TargetHashInt: 26959946667150639794667015087019630673637144422540572481103610249215
// BlockHashInt < TargetHashInt: true
```

https://golang.org/pkg/math/big/

Network layer

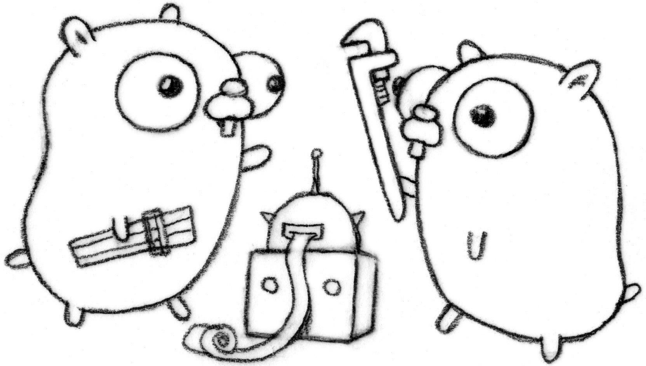
- gRPC
- go-libp2p

```
service Messager {
  rpc Message (Request) returns (Response) {}
  rpc Send (SendRequest) returns (SendResponse) {}
  rpc GetBalance (GetBalanceRequest) returns (GetBalanceResponse) {}
  rpc GetBlock (GetBlockRequest) returns (GetBlockResponse) {}
  rpc GetTX (GetTXRequest) returns (GetTXResponse) {}
  rpc GetAddress (GetAddressRequest) returns (GetAddressResponse) {}
}
```

https://github.com/grpc/grpc-go
https://mycodesmells.com/post/pooling-grpc-connections
https://github.com/libp2p/go-libp2p

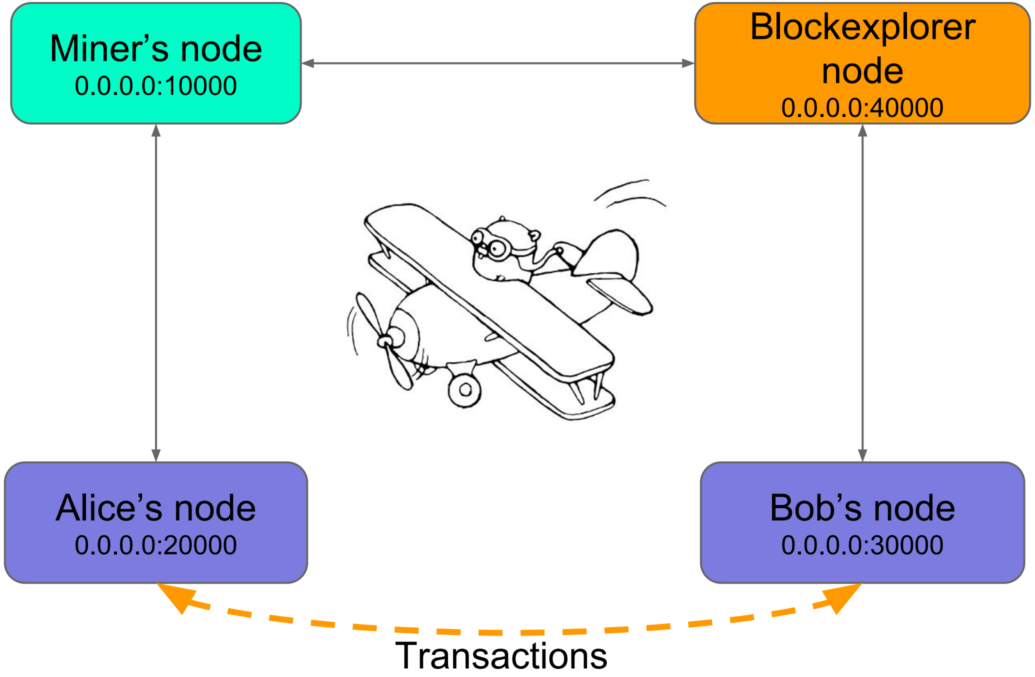
Block explorer

```
func main() {
        e := network.NewHTTPBlockExplorer(storage, memPool)
        http.HandleFunc("/tx/", e.ViewTXHandler)
        http.HandleFunc("/block/", e.ViewBlockHandler)
        http.ListenAndServe(l, nil)
}
```



https://github.com/urfave/cli

The plan of demo



Miner's node
0.0.0.0:10000

Blockexplorer node
0.0.0.0:40000

Alice's node
0.0.0.0:20000

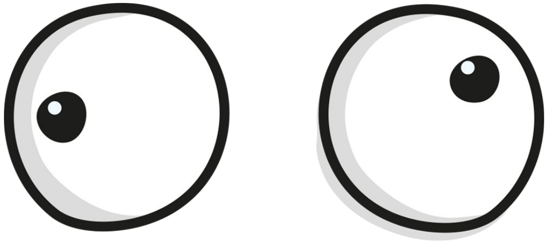
Bob's node
0.0.0.0:30000

Transactions

DEMO

Conclusion. Links

- https://github.com/btcsuite/btcd
- https://github.com/tendermint/tendermint
- https://github.com/cosmos/cosmos-sdk
- https://github.com/hyperledger/fabric-sdk-go
- https://github.com/amir20/sha-miner
- https://github.com/Jeiwan/blockchain_go



The end. Thank you!

@superstas88