

Go blockchain



Peter Borovanský, KAI, I-18,
borovan(a)ii.fmph.uniba.sk

Zdroje a motivácia:

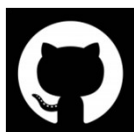
Séria vynikajúcich tutoriálnych článkov, ktorá začala:

Code your own blockchain in less than 200 lines of Go

Žiadna ambícia ich pozmeniť/vylepšiť, len prezentovať + malý refactoring kódu

Vybrané z nich:













- localhost <https://medium.com/@mycoralhealth/code-your-own-blockchain-in-less-than-200-lines-of-go-e296282bcffc>
- networking <https://medium.com/@mycoralhealth/part-2-networking-code-your-own-blockchain-in-less-than-200-lines-of-go-17fe1dad46e1>
- ťažba:
 - Proof of work <https://medium.com/@mycoralhealth/code-your-own-blockchain-mining-algorithm-in-go-82c6a71aba1f>
 - Proof of stake <https://medium.com/@mycoralhealth/code-your-own-proof-of-stake-blockchain-in-go-610cd99aa658>
- P2P <https://medium.com/@mycoralhealth/code-a-simple-p2p-blockchain-in-go-46662601f417>



Všetky originálne zdrojáky nájdete tu: <https://github.com/mycoralhealth/blockchain-tutorial>

Pokus o intro

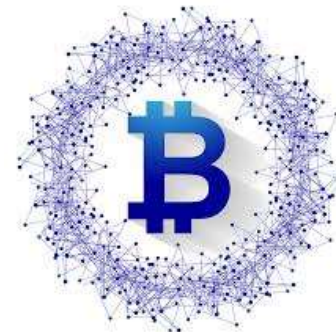
<https://coinmarketcap.com/>

Name	Price	24h	7d	Market Cap ⓘ	Volume ⓘ	Circulating Supply ⓘ	Last 7 Days
 Bitcoin BTC	\$11,445.56	▲ 1.09%	▲ 0.25%	\$211,980,169,589	\$17,564,844,150 1,534,642 BTC	18,520,731 BTC	
 Ethereum ETH	\$375.89	▲ 2.43%	▲ 0.02%	\$42,497,482,169	\$10,531,889,950 28,018,584 ETH	113,058,463 ETH	
 Tether USDT	\$1.00	▲ 0.01%	▼ 0.03%	\$15,828,841,061	\$29,243,380,880 29,221,437,027 USDT	15,816,963,304 USDT	
 XRP XRP	\$0.242508	▲ 0.78%	▼ 5.88%	\$10,973,035,693	\$1,174,217,281 4,841,965,074 XRP	45,248,061,374 XRP	
 Bitcoin Cash BCH	\$249.38	▲ 2.25%	▲ 3.1%	\$4,625,634,172	\$1,758,464,301 7,051,250 BCH	18,548,288 BCH	
 Binance Coin BNB	\$30.61	▲ 1.24%	▲ 7.11%	\$4,420,396,780	\$437,298,831 14,285,781 BNB	144,406,561 BNB	

Ďalší pokus o intro

- Laszlo Hanyecz, May 22, 2010, two pizzas for 10,000 BTC





Intro

Najlepšia cesta, ako pochopiť blockchain, je vyrobiť si vlastný.

- vytvoríme vlastný blockchain
- pochopíme, ako funguje hašovanie pri udržiavaní integrity blockchainu
- pochopíme, ako funguje pridávanie vrcholov blockchainu
- zobrazíme blockchain v browseri, resp. Postman
- distribuujeme blockchain na viacero uzlov
- pochopíme, čo je proof of work, napr. pre ťažbu, napr. ₿
- pochopíme, čo je proof of stake, pri iných menách

Pre spúšťanie zdrojákov k prednáške si doinštalujte balíky (v cmd okne):

```
go get github.com/joho/godotenv
```

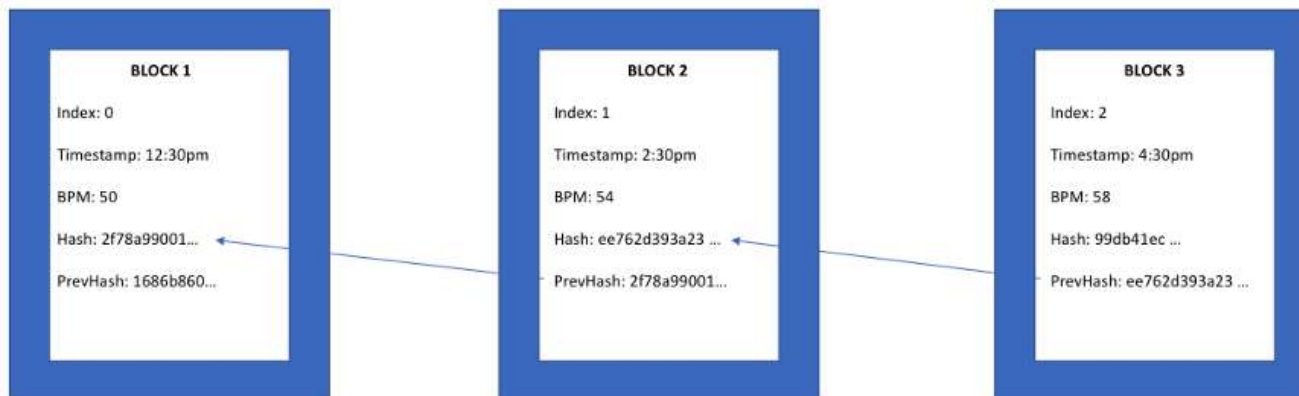
```
go get github.com/gorilla/mux
```

```
go get github.com/davecgh/go-spew/spew
```

Disclaimer: Nie je vylúčené, že na konci prednášky budete bohatší, ale nie o bitcoin...

Block a chain

```
type Block struct {  
    Index      int    // poradové číslo bloku v reťazci, začína 0  
    Timestamp  string // čas vytvorenia  
    Data       int     // data blockchainu (v originali BPM)  
    Hash       string  // SHA256 haš hodnôt (Index, Timestamp, )  
    PrevHash   string  // SHA256 haš (hash) predošlého bloku  
}  
  
var Blockchain []Block // nič viac ako pole previazaných blokov  
Data predstavuje dátovú informáciu v bloku, napr. stav účtu v ₿
```





Čo je SHA-256 haš ?

Určite poznáte rôzne kódovania/šifrovania:

- MD5, SHA-1
- SHA-256 (32x8bits), SHA-512 (64x8bits)

Ich výpočtová náročnosť (64 bits Windows 10 Intel i7 2.60GHz, 16GB RAM)

Hash	#1 (ms)	#2 (ms)	#3 (ms)	#4 (ms)	#5 (ms)	Average per 1M (ms)
MD5	649	623	621	624	620	627.4
SHA-1	608	588	630	600	594	604
SHA-256	746	724	741	720	758	737.8
SHA-512	1073	1055	1050	1052	1052	1056.4

Haš kalkulačka

- <https://www.xorbin.com/tools/sha256-hash-calculator>

Ako sa SHA-256 počíta

(z čoho) ?

```
type Block struct {  
    Index      int  
    Timestamp  string  
    Data       int  
    Hash       string  
    PrevHash   string  
}
```

```
// vypočíta SHA 256 haš pre blok
```

```
func calculateHash(block Block) string {  
    h := sha256.New()  
    h.Write([]byte(string(block.Index) + block.Timestamp +  
                    string(block.Data) + block.PrevHash))  
    hashed := h.Sum(nil) // zlepí všetky dáta bloku do string  
    return hex.EncodeToString(hashed)
```

```
}
```

```
// vygeneruje nový blok s hodnotou Data, indexom old.Index+1,  
// ale musíme poznať haš predošlého bloku
```

```
func generateBlock(oldBlock Block, Data int) (Block, error) {  
    newBlock := Block{ oldBlock.Index + 1,  
                        time.Now().String(),  
                        Data, "", oldBlock.Hash} // pointer na  
    newBlock.Hash = calculateHash(newBlock)      // starý blok  
    return newBlock, nil
```

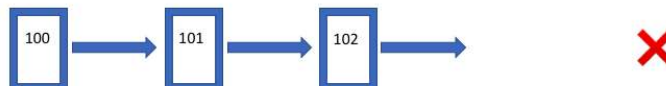
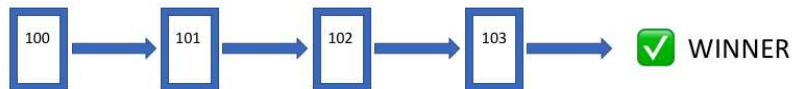
```
}
```

<https://medium.com/@mycoralhealth/code-your-own-blockchain-in-less-than-200-lines-of-go-e296282bcffc>

SHA-256

```
// skontroluje, či oldBlock je predchodcom newBlock
// skontroluje index, haš v newBlock oproti predošlému oldBlock
func isValidBlock(newBlock, oldBlock Block) bool {
    return (oldBlock.Index+1 == newBlock.Index) &&
        (oldBlock.Hash == newBlock.PrevHash) &&
        (calculateHash(newBlock) == newBlock.Hash)
}
```

Blockchain je vždy distribuovaný na viacerých uzloch eko-systému. **Robustnosť**.
Ak máme dva blockchain, oba vznikli pridávaním uzlov, ktorý pravdivý? **Dlhší**.



HTTP Server – JSON

(použitím Gorilla/mux package)

```
muxRouter := mux.NewRouter()
muxRouter.HandleFunc("/", handleGET).Methods("GET")
muxRouter.HandleFunc("/", handlePOST).Methods("POST")
log.Println("Listening on ", os.Getenv("ADDR")) // port: 8080
s := &http.Server{
    Addr:           ":", // počúva na localhost: 8080
    Handler:        muxRouter, // obsluhuje HTTP GET a POST reqs.
    ReadTimeout:    10 * time.Second,
    WriteTimeout:   10 * time.Second,
    MaxHeaderBytes: 1 << 20,
}
s.ListenAndServe(); // ... trochu, zjednodušené, pozri .go
....
// GET Method handler
func handleGET(w http.ResponseWriter, r *http.Request) {
    bytes, err := json.MarshalIndent(Blockchain, "", " ")
    // celý blockchain prehodí do JSONu a vypíše do responsu
    io.WriteString(w, string(bytes))
}
```



HTTP Server – PUT

(použitím Gorilla/mux package)

```
// POST Method handler
```

```
→ func handlePOST(w http.ResponseWriter, r *http.Request) {
```

```
    var m Message
```

```
    decoder := json.NewDecoder(r.Body)
```

```
    // dekódujeme body POST do štruktúry Message, ak sa dá  
    decoder.Decode(&m)
```

```
    if ... okay // ... trochu, zjednodušené, pozri .go
```

```
    defer r.Body.Close() // vytvoríme nový blok s novou hodnotou
```

```
        // od posledného blocku v chain, a Data z POST requestu
```

```
    newBlock:=generateBlock(Blockchain[len(Blockchain)-1], m.Data)
```

```
    if isValid(newBlock, Blockchain[len(Blockchain)-1]) {
```

```
        newBlockchain := append(Blockchain, newBlock)
```

```
        replaceChain(newBlockchain)
```

```
        spew.Dump(Blockchain) // fmt.Println(Blockchain)
```

```
    }
```

```
    respondWithJSON(w, r, http.StatusCreated, newBlock)
```

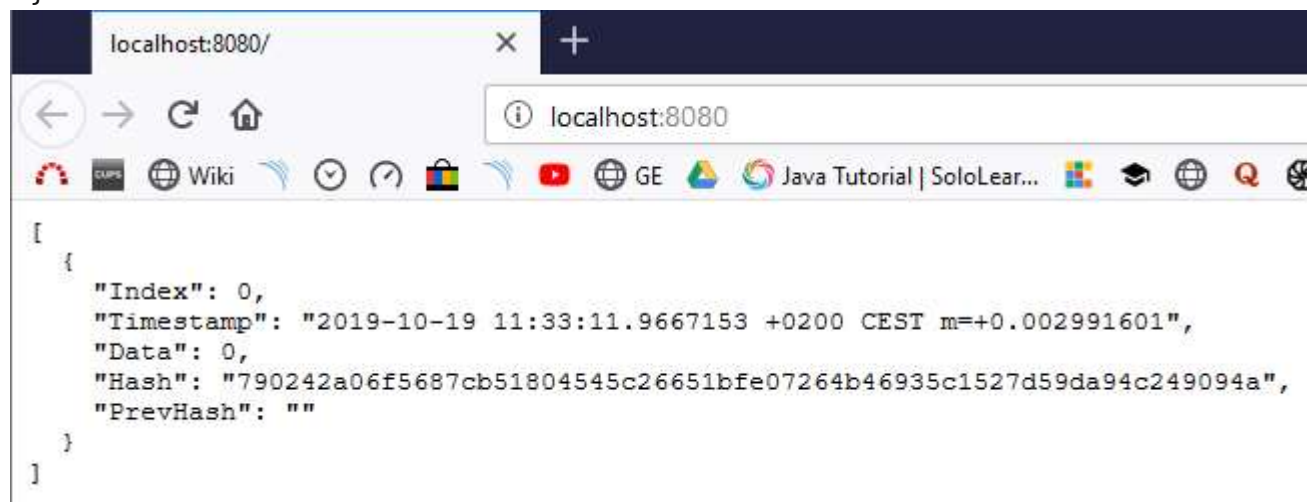
```
}
```

Vytvoríme genesis block

```
genesisBlock := Block{0, time.Now().String(), 0, "", ""} // počiatočný
genesisBlock.Hash = calculateHash(genesisBlock)
spew.Dump(genesisBlock) // miesto fmt.Println(genesisBlock)
Blockchain = append(Blockchain, genesisBlock) // genesis seedovaný
```

2019/10/19 11:33:11 Listening on 8080

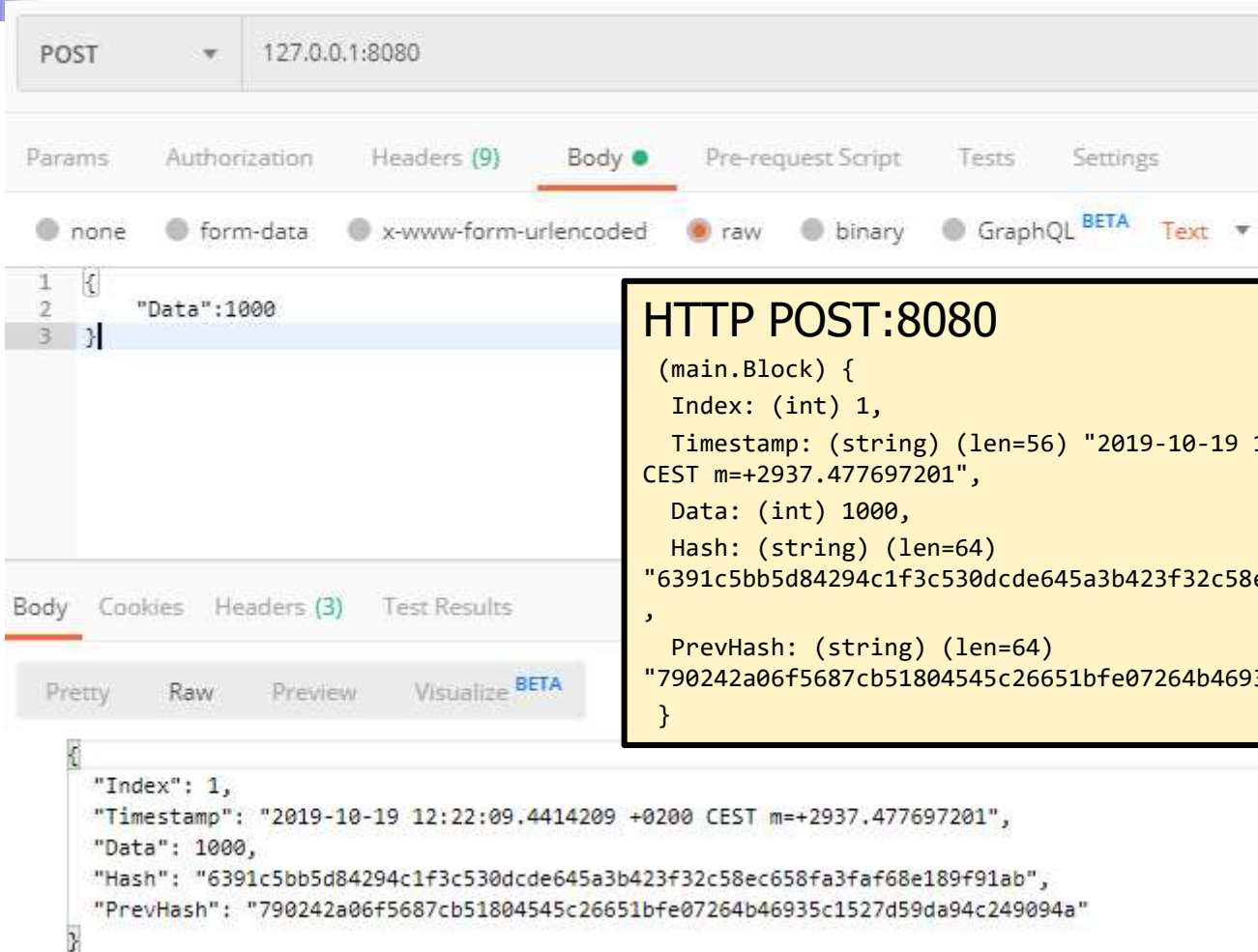
```
(main.Block) {
  Index: (int) 0,
  Timestamp: (string) (len=53) "2019-10-19 11:33:11.9667153 +0200 CEST m=+0.002991601",
  Data: (int) 0,
  Hash: (string) (len=64) "790242a06f5687cb51804545c26651bfe07264b46935c1527d59da94c249094a",
  PrevHash: (string) ""
}
```



GET

verzia_localhost

Pridanie ďalšieho bloku



POST 127.0.0.1:8080

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded **raw** binary GraphQL BETA Text

```
1 {  
2   "Data": 1000  
3 }
```

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize BETA

```
[  
  {  
    "Index": 1,  
    "Timestamp": "2019-10-19 12:22:09.4414209 +0200 CEST m=+2937.477697201",  
    "Data": 1000,  
    "Hash": "6391c5bb5d84294c1f3c530dcde645a3b423f32c58ec658fa3faf68e189f91ab",  
    "PrevHash": "790242a06f5687cb51804545c26651bfe07264b46935c1527d59da94c249094a"  
  }  
]
```

HTTP POST:8080

```
(main.Block) {  
  Index: (int) 1,  
  Timestamp: (string) (len=56) "2019-10-19 12:22:09.4414209 +0200  
CEST m=+2937.477697201",  
  Data: (int) 1000,  
  Hash: (string) (len=64)  
"6391c5bb5d84294c1f3c530dcde645a3b423f32c58ec658fa3faf68e189f91ab"  
,  
  PrevHash: (string) (len=64)  
"790242a06f5687cb51804545c26651bfe07264b46935c1527d59da94c249094a"  
}
```

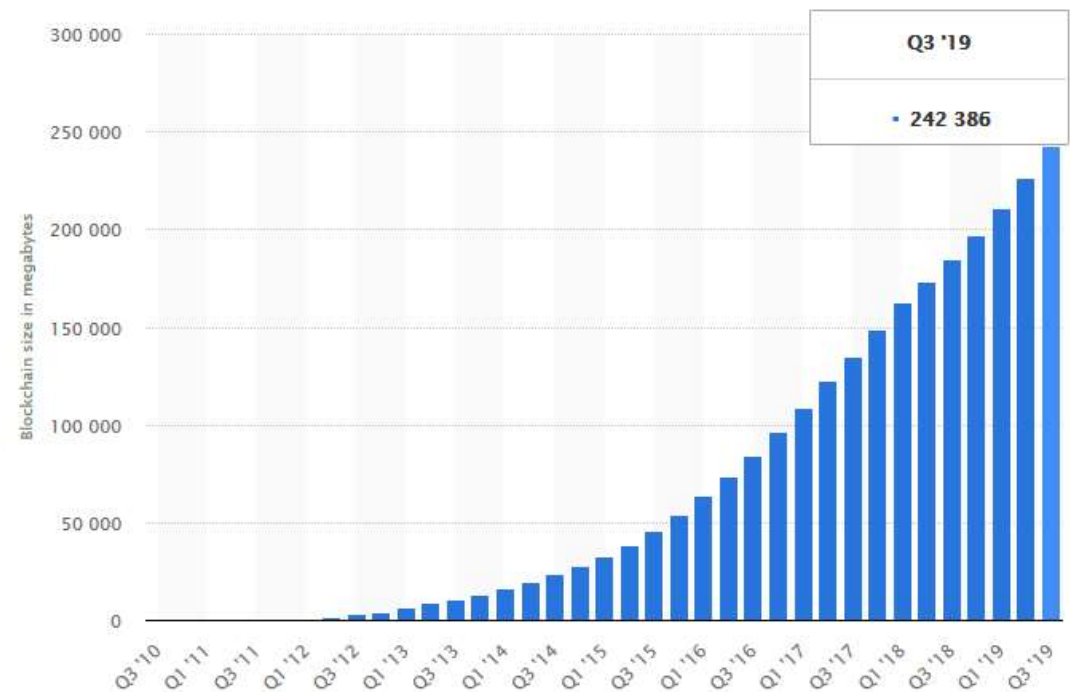
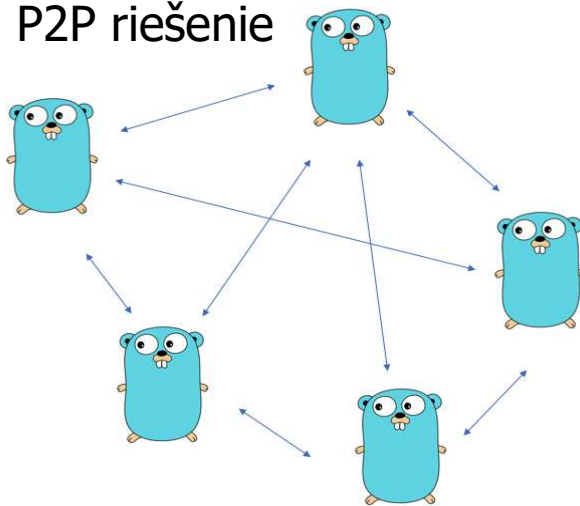
Problém

Vytvorené riešenie je single-node, preto, ak niekto zničí uzol, na ktorom je blockchain uložený, zničil všetko.

V takej banke by ste peniaze nechceli mať.

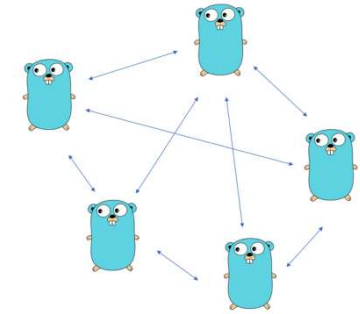
Riešenie:

- broadcast blockchainu
- P2P riešenie



<https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/>

Networking

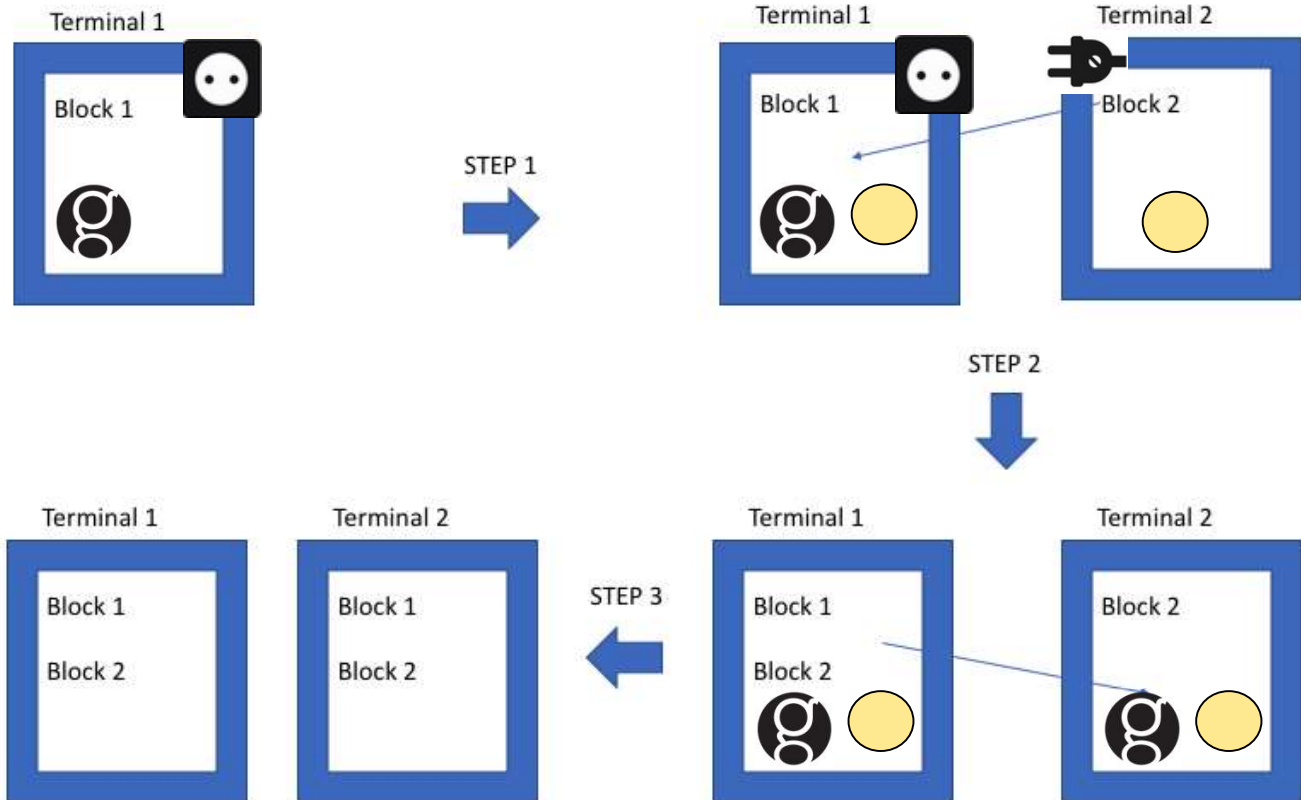


1) T1 má otvorený server socket

2) T2 connect T1
Add Block2 do T1

3) T1 broadcastuje

3) T1,T2 sú sync



Každý uzol počúva z kanála prichádzajúce blockchajny (prenášame celé pole blokov!)

```
var bcServer chan []Block // kanál, na ktorom počúvam  
                           prichádzajúce blockchajny
```




TCP Server Socket

```
func main() { // vytvor genesis-počiatočný block
    genesisBlock := Block{0, time.Now().String(), 0, "", ""}
    genesisBlock.Hash = calculateHash(genesisBlock) // jeho haš
    Blockchain = append(Blockchain, genesisBlock) // |chain|=1

    bcServer = make(chan []Block) // otvor kanál, na ktorom počúva
    // start TCP and serve TCP server
    server := net.Listen("tcp", ":"+os.Getenv("SERVERSOCKETPORT"))
    // otvor Server socket na :9000,1,2
    defer server.Close() // final
    for {
        conn, err := server.Accept()
        go handleConn(conn) // handler pre každého, čo urobí
                             // connect na port :9000+i
    }
}
```


vytvorí 3 vlákna

- 

SERVER

nod320.ad.mstep - PuTTY

nod320.ad.mstep - Pull

verzia_networking

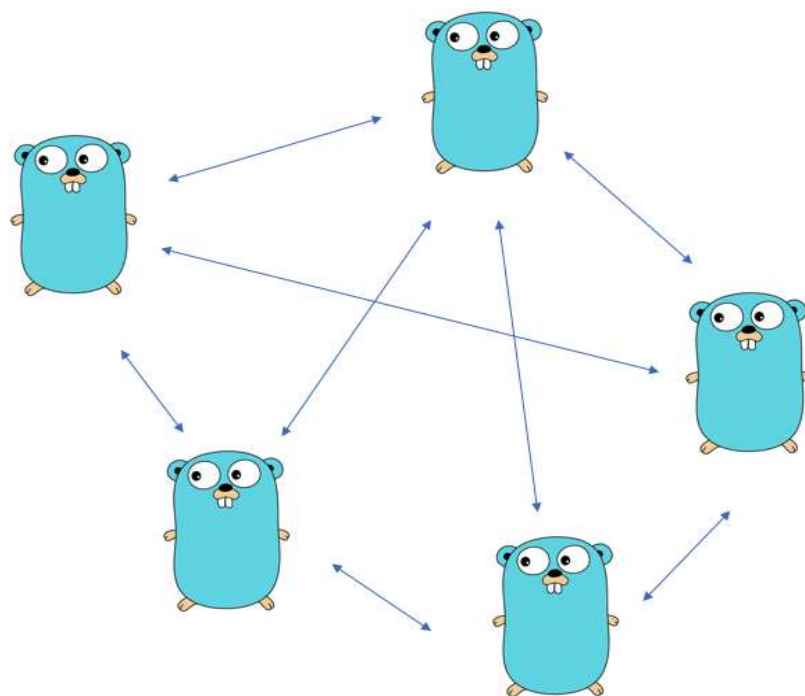


func handleConn(conn net.Conn)

```
defer conn.Close() // handler po connect na 9000
io.WriteString(conn, "Enter a new Data:") // na konzolu vypíše prompt
scanner := bufio.NewScanner(conn) // scanner pre stdin
→ go func() {
    for scanner.Scan() { // čítaj Data z stdin
        data, err := strconv.Atoi(scanner.Text()) // konvertuj
        newBlock, err := generateBlock(Blockchain[len(Blockchain)-1], data)
        if isValid(newBlock, Blockchain[len(Blockchain)-1]) {
            replaceChain(append(Blockchain, newBlock))
        }
        bcServer <- Blockchain // pošle nový blockchain do kanálu
        io.WriteString(conn, "\nEnter a new Data:")
    } }()
→ go func() { // broadcastuje celý blockchain do output ako json
    for { // všetkým klientom, ktorí sa pripojili, raz za 30s.
        time.Sleep(30 * time.Second)
        output, err := json.Marshal(Blockchain) // zabalí Bchain do jsonu
        io.WriteString(conn, string(output))
    } }()
→ for _ = range bcServer { spew.Dump(Blockchain) } // dump na konzolu
```

Problém

Zodpovedá tento obrázok tomu, čo sme vytvorili ?



Čo je P2P ?

<https://medium.com/@mycoralhealth/code-a-simple-p2p-blockchain-in-go-46662601f417>

out of scope...

Mining algo

(čo je ťažba)



Bitcoin nemá žiaden centrálny FED, ktorý by tlačil (a znehodnotil) ₿

Bitcoinov bude len 21 mil.

Jeden bitcoin sa uro(b/d)í každých cca 10min, 6/h, 144/d, 52560/y, ~210000/4y

Toto je riadene bitcoin protokolom.

Ako aj zlata ubúda, tak aj voľných ₿, preto aj ťažba je čoraz umelo ťažšia, komplikovanejšia a výpočtovo náročnejšia (ak nie ste poslanec).

Ťažba je výpočet a očakáva sa nejaký Proof of Work (napr. Bitcoin, Ethereum).

Poslanec pirátů „šel příkladem“.

V služebním bytě s elektřinou

zdarma těžil kryptoměnu

Na Slovensku sú desiatky kryptofariem, ťažil aj Glváč

<https://domov.sme.sk/c/20855672/na-slovensku-su-desiatky-kryptofariem-tazil-aj-qlvac.html>

zdroj: <https://medium.com/@mycoralhealth/code-your-own-blockchain-mining-algorithm-in-go-82c6a71aba1f>



Satoshi



Satoshi Nakamoto – zakladateľ BTC, ale nikto nevie, kto to je...
Version 0.1 of bitcoin, Sourceforge on 9 January 2009.

Odmena za jeden nový blok (*block reward*) bola 50 BTC pri genesis bloku #1.

Delí sa dvomi po každých 210 tisíc blokoch

Logaritmické zdraženie

Block Reward History

Exponenciálny rast výkonu

Jeden blok sa ťaží raz za 10 min.

Del'ba na 1/2 je každé cca 4 roky.

Odmena za blok je teraz

~~12.50 BTC.~~

6.25 BTC

Date reached	Block	BTC/block	Year (estimate)	BTC Added	End % of Limit
1/3/09	0	50	2009	2625000	12.50%
4/22/10	52500	50	2010	2625000	25.00%
1/28/11	105000	50	2011	2625000	37.50%
12/14/11	157500	50	2012	2625000	50.00%
11/28/12	210000	25	2013	1312500	56.25%
10/9/13	262500	25	2014	1312500	62.50%
8/11/14	315000	25	2015	1312500	68.75%
7/29/15	367500	25	2016	1312500	75.00%
	420000	12.5	2017	656250	78.13%
	472500	12.5	2018	656250	81.25%
	525000	12.5	2019	656250	84.38%
	577500	12.5	2020	656250	87.50%
	630000	6.25	2021	328125	89.06%
	682500	6.25	2022	328125	90.63%

<https://www.bitcoinblockhalf.com/>

Proof of work

(čo je ťažba)

Čo je dôkaz prácou ?

riešiť ťažký matematický problém.

sha-256: PARA\n → c29d55d407abfa5a5a88ec07d0e424724985e1890da8f1c294b9c006b8c21155

Ale uhádnuť z ec95943926e7348a596157dff..., že vzorom je PARA je prakticky nemožné !!!

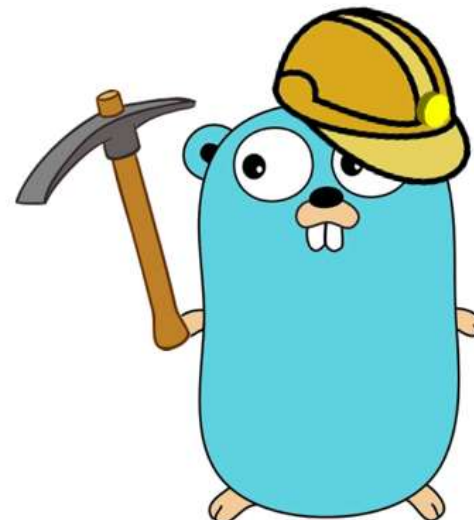
Tu je SHA-256 kalkulačka

Viete nájsť x (nazýva sa **nonce**) také, že sha-256(x) začína

- 0
- 00
- 000 napr. 886, 1039, 3633, 5848, ...
- 0000 napr. 88484, APPLICANT, KYE
- 00000 napr. WIDDIES

Lenže googliť vám nepomôže, ak máte najst' x také, že sha-256("PARA"+x) začína

- 0
- 00
- 000
- 0000





$\text{sha-256}(\text{PARA}+x) \rightarrow 000\dots$

666 000a644865962faadcaf0000324538567109c6387aa31cb1e798df9e59df645f -

Prémia Banícka: Zahrajte sa na para ťažiča – tzv. paraminer.

Nájdite x , aby $\text{sha-256}(\text{"PARA"}+x)$ malo aspoň

- 4 úvodné nuly, [prvý má 0.125 bodu, ďalší nič]
- 5 úvodných núl, [prvý má 0.25 bodu, ďalší nič]
- 6 úvodných núl, [prvý má 0.5 bodu, ďalší nič]
- 7 úvodných núl, [prvý má 1 bod, ďalší nič]
- 8 úvodných núl, [prvý má 2 body, ďalší nič]
- 9 úvodných núl, [prvý má 4 body, ďalší nič]
- 10 úvodných núl, [prvý má 8 bodov, ďalší nič]
- 11 úvodných núl, [prvý má 16 bodov, ďalší nič]
- 12 úvodných núl, [prvý má 32 bodov, ďalší nič]
- 13 úvodných núl, [prvý má 64 bodov, ďalší nič]
- 14 úvodných núl, [prvý má instantne Ačko do indexu]



Priložiť musíte váš script, čo to vytiažil. Rozhoduje čas submitu do L.I.S.T.u

Block #411940

Summary

Number Of Transactions	1098
Output Total	7,253.02558914 BTC
Estimated Transaction Volume	933.02422913 BTC
Transaction Fees	0.23578732 BTC
Height	411940 (Main Chain)
Timestamp	2016-05-16 00:22:09
Received Time	2016-05-16 00:22:09
Relayed By	BTCC Pool
Difficulty	194,254,820,283.44
Bits	403024122
Size	814.78 kB
Weight	3258.868 kWU
Version	0x20000001
Nonce	506565497
Block Reward	25 BTC

Hashes

Hash	0000000000000000057fcc708cf0130d95e27c5819203e9f967ac56e4df598ee
Previous Block	00000000000000000337fc316c605be3392b8b26aaf387c9756915feecb4780f
Next Block(s)	0000000000000000019d7f37d7b9dd5e4d36b16dd0355f1cc30933fabf4dcc12
Merkle Root	f7b19ebf61f25feba68f7e3b139a846f2b6d7a4eb442085007bccf76b523849c


€7,198.96 +€7,101.57 (69.6K%)

1H 24H 1W 1M 1Y ALL



18

Block 528249

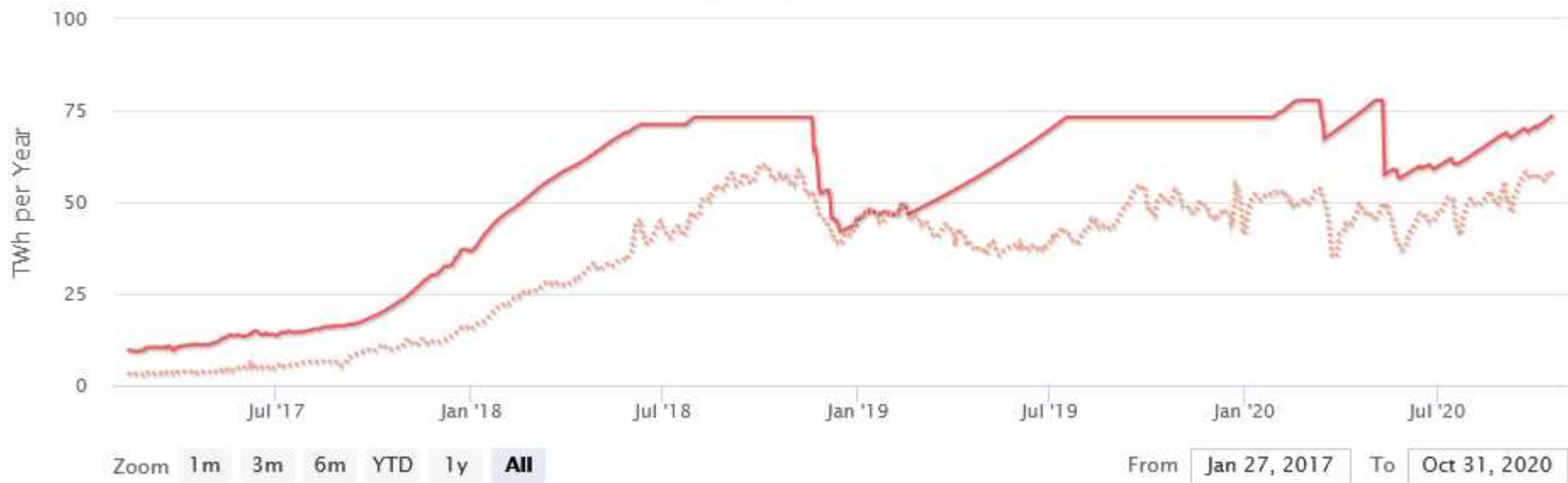
Hash	000000000000000000021e800c1e8df51b22c1588e5a624bea17e9faa34b2dc4a 
Confirmations	125,149
Timestamp	2018-06-19 21:32
Height	528249
Miner	BTC.TOP
Number of Transactions	407
Difficulty	5,077,499,034,879.02
Merkle root	53c2c974e45311de5c34047a5e4c9bb9b938f5d2a9eb1cd3c98a1dbea370a377
Version	0x20000000
Bits	389,508,950



73TWh/y

Bitcoin Energy Consumption Index Chart

Click and drag in the plot area to zoom in



Annual Total Footprint

Carbon Footprint

34.73 Mt CO₂



Comparable to the carbon footprint of
Denmark.

Electrical Energy

73.12 TWh



Comparable to the power
consumption of Austria.

Electronic Waste

10.74 kt



Comparable to the e-waste generation
of Luxembourg.

Single Transaction Footprints

Carbon Footprint

296.68 kgCO₂



Equivalent to the carbon footprint of
741,703 VISA transactions or 49,447
hours of watching Youtube.

Electrical Energy

624.59 kWh



Equivalent to the power consumption
of an average U.S. household over
21.11 days.

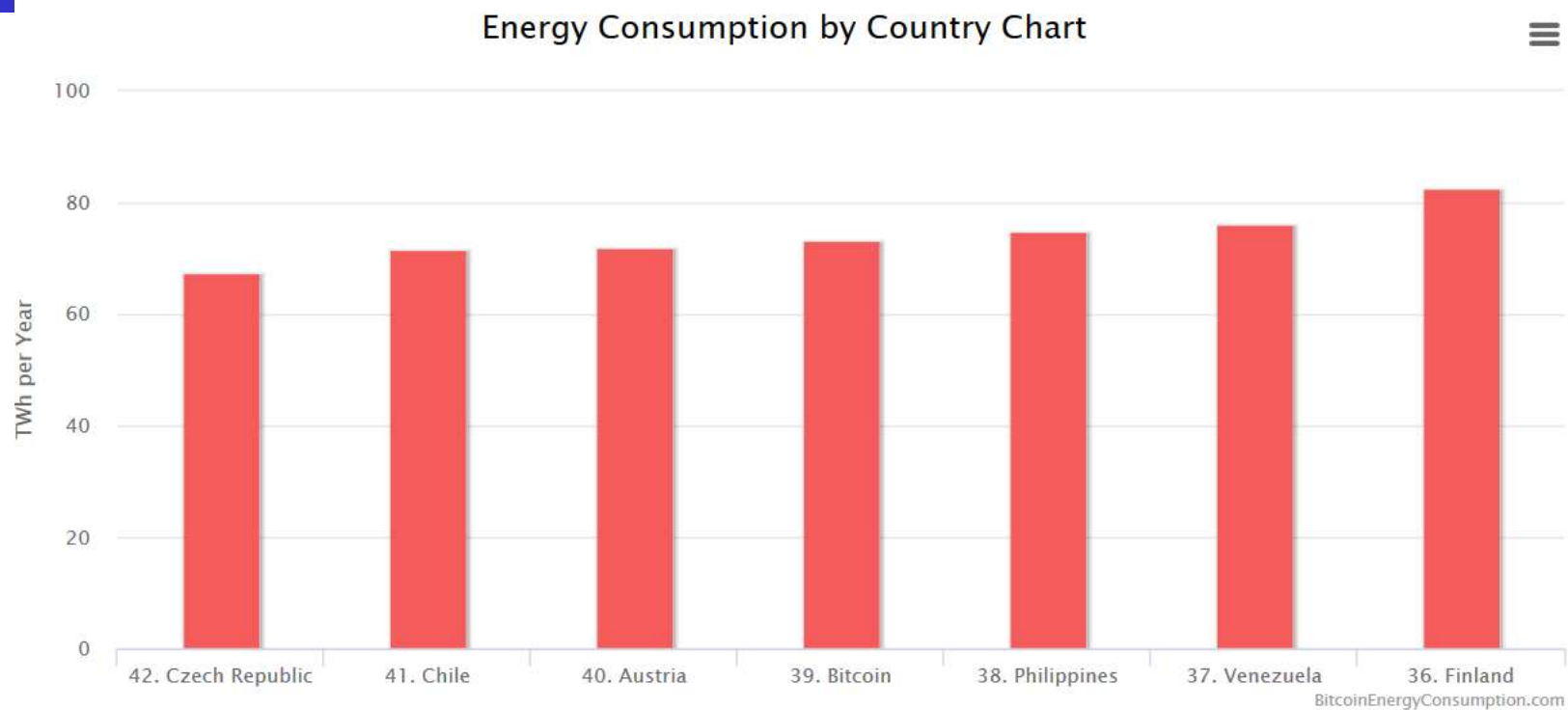
Electronic Waste

91.80 grams



Equivalent to the weight of 1.41 'C'-
size batteries or 2.00 golf balls. (Find
more info on e-waste [here.](#))

73TWh/y



asi najväčší problém ťažby - väčšina TWh je z čínskej špinavej/uholnej elektriny

Ge

(BTC – Block 0, resp. Block 1)

Genesis Block:

```
GetHash() = 0x00000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f -10leading zeros
```

```
hashMerkleRoot = 0x4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b
```

```
txNew.vin[0].scriptSig = 486604799 4
```

0x736B6E616220726F662074756F6C69616220646E6F63657320666F206B6E697262206E6F20726F6C6C65636E6

1684320393030322F6E614A2F33302073656D695420656854

```
txNew.vout[0].nValue = 5000000000
```

```
txNew.vout[0].scriptPubKey =
```

0x5F1DF16B2B704C8A578D0BBAF74D385CDE12C11EE50455F3C438EF4C3FBCF649B6DE611FEAE06279A60939E02

8A8D65C10B73071A6F16719274855FEB0FD8A6704 OP_CHECKSIG

Convert epoch to human-readable

```
block.nVersion = 1
```

```
block.nTime = 1231006505
```

```
block.nBits = 0x1d00ffff
```

```
block.nNonce = 2083236893
```

```
CBlock(hash=00000000019d6, ver=1, hashPrevBlock=00000000000000, hashMerkleRoot=4a5e1e,
      nTime=1231006505, nBits=1d00ffff, nNonce=2083236893, vtx=1)
```

```
CTransaction(hash=4a5e1e, ver=1, vin.size=1, vout.size=1, nLockTime=0)
```

```
CTxIn(COutPoint(000000, -1), coinbase
```

04ffff001d0104455468652054696d65732030332f4a616e2f32303039204368616e63656c6c6f72206f6e20627

2696e6b206f66207365636f6e64206261696c6f757420666f722062616e6b73)

```
CTxOut(nValue=50.00000000, scriptPubKey=0x5F1DF16B2B704C8A578D0B)
```

```
vMerkleTree: 4a5e1e
```

1231006505

Timestamp to Human date

- Supports Unix timestamps in seconds, milliseconds, micro

Assuming that this timestamp is in **seconds**:

GMT

: Saturday, January 3, 2009 6:15:05 PM

Čo sú dáta v Bloku 1

CTxIn(COutPoint(000000, -1), coinbase

04ffff001d0104455468652054696d65732030332f4a616e2f32303039204368616e63656c6c6f72
206f6e206272696e6b206f66207365636f6e64206261696c6f757420666f722062616e6b73)

Je hypotéza, že Satoshi Nakamoto asi žil v Spojenom kráľovstve

5468652054696d65732030332f4a616e2f32303039204368616e63656c6c6f72206f6e206272696e6b206f66207365636f6e64206261696c6f757420666f722062616e6b73)

Character encoding:

ASCII

Convert Reset Swap

The Times 03/Jan/2009 Chancellor on brink of second bailout for banks



<https://www.rapidtables.com/convert/number/hex-to-ascii.html>



Bitcoin Consensus Algorithm

Ako sa v distribuovanom prostredí, kde

- uzly môžu zlyhávať
- komunikácia vypadávať
- pôsobia klamári, záškodníci „mokujúci“ falošnú komunikáciu

dohodnúť na niečom, presnejšie

- na čomkoľvek
- ktorý blok je pridaný korektne do blockchainu.

Ako sa zabezpečiť, aby sa blok vytlačil raz za 10 minút, pričom výkon rastie ?

Ako sa zabezpečiť, aby niekto nemal tajne pred-rátané haše ?

In cryptography, a **nonce** is an arbitrary number that can be used just once. It is often a random or pseudo-random number issued in an authentication protocol to ensure that old communications cannot be reused in replay attacks. -- wiki

Nonce

```
func calculateHash(block Block) string { // vypočíta SHA 256 haš pre blok
    h := sha256.New()
    h.Write([]byte(string(block.Index) + block.Timestamp + string(block.Data) +
        block.PrevHash + block.Nonce))
    hashed := h.Sum(nil)
    return hex.EncodeToString(hashed)
}
```

Hľadanie nonce

```
func generateNewBlock(oldBlock Block, Data int) (Block) {
    newBlock := Block{oldBlock.Index+1, time.Now().String(), Data, "", oldBlock.Hash, ...
    newBlock.Hash = calculateHash(newBlock)
    for nonce := 0; ; nonce++ { // naivné hľadanie nonce v cykle 0...
        newBlock.Nonce = fmt.Sprintf("%x", nonce) // kandidát sa zapíše do bloku
        if !strings.HasPrefix(calculateHash(newBlock), strings.Repeat("0", difficulty)){
            fmt.Println(calculateHash(newBlock), " do more work!") // netrafili sme sa :(
            time.Sleep(time.Second) // zabráni prehrievaniu :)
            continue
        } else {
            fmt.Println(calculateHash(newBlock), " work done!") // hurá, máme nový blok :)
            newBlock.Hash = calculateHash(newBlock)
            break
        }
    }
}
```





Generovanie nového bloku

- pustite server verzia PoW, na porte 8080 počúva HTTP POST a GET, ako prvá verzia
- cez HTTP klienta generujte POST request, v tele requestu zadajte json, napr. { "Data":1000}
- uvidíte, ako sa na „serveri“ háda/hľadá nonce v cykle od 0...
- dáva si dramatizujúci sleep 1sek po neúspechu

2019/10/19 17:33:04 Listening on 8080

```
(main.Block) {  
  Index: (int) 0, ... }
```

4e7c55f05b8e2f6f8f37fe297c780bb543dc6db92239dc3b688e580efa269f64 do more work!

..... toto trvá

b3af8d27de02e9ef0957bd916bc8684cd6446338087a5a95d63d3263a5c5aefc do more work!

00705c8dd7f56f91c452d8be6aabaee84727f5fd5d7b0c7457b38ce0f5d8a5d72 work done!

Hash/s: 0.99950210.5

```
(main.Block) {  
  Index: (int) 1,  
  Timestamp: (string) (len=54) "2019-10-19 17:33:32.1453664 +0200 CEST",  
  Data: (int) 1000,  
  Difficulty: (int) 2,  
  Nonce: (string) (len=2) "1b"
```



BlockChain dĺžky 5, diff=6

Vyhod'te sleep !!!

pomocou zjednodušeného algoritmu na ťažbu si skúste vytážiť blockchain, teda sekvenciu klov, maximálnej dĺžky pre difficulty level 6.

Môžete využiť terminálku, ale nepriznajte sa, že ste z PARA.

Viete vytážiť nejaký rozumne dlhý blokchain pre difficulty 7 ?

Takto nejak by mal vyzerat' váš výstup (ilustrácia pre diff=6):

```
[ { "Index": 0, "Timestamp": "2019-10-19 17:48:30.9753317 +0200 CEST m=+0.003989201", "Data": 0, "Hash":  
  "96a296d224f285c67bee93c30f8a309157f0daa35dc5b87e410b78630a09cfc7", "PrevHash": "", "Difficulty": 6, "Nonce": "" },  
{ "Index": 1, "Timestamp": "2019-10-19 17:48:37.4570445 +0200 CEST m=+6.485702001", "Data": 1000, "Hash":  
  "0000001c4b6cd9031885393192a0981ba2006a9f3a54c9e58fbfa9ea4e0f897d", "PrevHash":  
  "96a296d224f285c67bee93c30f8a309157f0daa35dc5b87e410b78630a09cfc7", "Difficulty": 6, "Nonce": "4b2df5" },  
{ "Index": 2, "Timestamp": "2019-10-19 17:50:51.7591878 +0200 CEST m=+140.787845301", "Data": 1001, "Hash":  
  "0000005a45587f38bf8d8ec9cd800eb8480501bfbdd92065eba4d671b40b617c", "PrevHash":  
  "0000001c4b6cd9031885393192a0981ba2006a9f3a54c9e58fbfa9ea4e0f897d", "Difficulty": 6, "Nonce": "16fb42b" },  
{ "Index": 3, "Timestamp": "2019-10-19 17:53:56.0622716 +0200 CEST m=+325.090929101", "Data": 1001, "Hash":  
  "000000d6e57c9bc5395ec315a584c8e078ac6c443a38f4910bd051d62a583c5f", "PrevHash":  
  "0000005a45587f38bf8d8ec9cd800eb8480501bfbdd92065eba4d671b40b617c", "Difficulty": 6, "Nonce": "60ec28" },  
{ "Index": 4, "Timestamp": "2019-10-19 17:54:39.8128602 +0200 CEST m=+368.841517701", "Data": 1001, "Hash":  
  "00000072bc6e1afcfda7a5ca9fc75faac0c73a98575c7ea0d705b9925ed39002", "PrevHash":  
  "000000d6e57c9bc5395ec315a584c8e078ac6c443a38f4910bd051d62a583c5f", "Difficulty": 6, "Nonce": "353d619" },  
{ "Index": 5, "Timestamp": "2019-10-19 18:00:26.3314238 +0200 CEST m=+715.360081301", "Data": 1001, "Hash":  
  "0000001f33c04174b7452b0e5c78d8820ba53377b735f5c7cdca30ce1c74f7c", "PrevHash":  
  "00000072bc6e1afcfda7a5ca9fc75faac0c73a98575c7ea0d705b9925ed39002", "Difficulty": 6, "Nonce": "c88bfe" } ]
```



Ako to funguje ?

- protokol každých cca 10 minút vyhlási bitku o nový blok,
- Bitcoin miesto SHA-256 používa zdvojený SHA-256², len 2 x opakujete SHA,
- keď ťažič nájde správnu hodnotu $x = \text{nonce}$, aby SHA-256 začínalo daným počtom núl, ťažič vyhráva blok, a tým aj BTC odmenu za neho mínus poplatky,
- všetkým sa broadcastuje informácia o novom bloku, a všetci si ho zapíšu,
- začína boj o nový blok,
- tí, čo neuspeli, nemajú nič, len spálili elektriku,
- väčšiu šancu majú tí, urobia viac *nezmyselných* výpočtov SHA, teda tí, čo majú viac HW sily (veľkosť a výkon)
- počet hašov za sec na vašom gride/rig, typicky udávané v GHash
- inak je to skoro lotéria,
- asi...? oveľa viac ľudí sa živí obchodovaním BTC, ako jeho dolovaním

Hashfast Sierra 1.2Th/s Bitcoin Miner

500 USD

HashFast Sierra 1.2TH/s Bitcoin Miner

Contains 3 HashFast Golden Nounce Asic

Performance:

1,200 Ghash/s at nominal clock speed.
(That's 1.2 Thash/s in one mining unit.)

Power: 1300 W

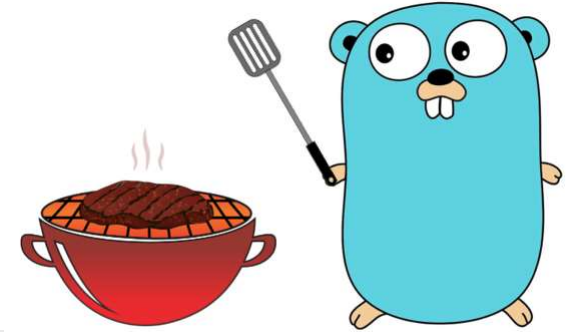
At the wall, this unit consumes 1 watt per gigahash +/- 20%

Chassis: 4U rack mounted

Môj „rig“ má výkon Hash/s: 156908 = 0.15 GHash/s



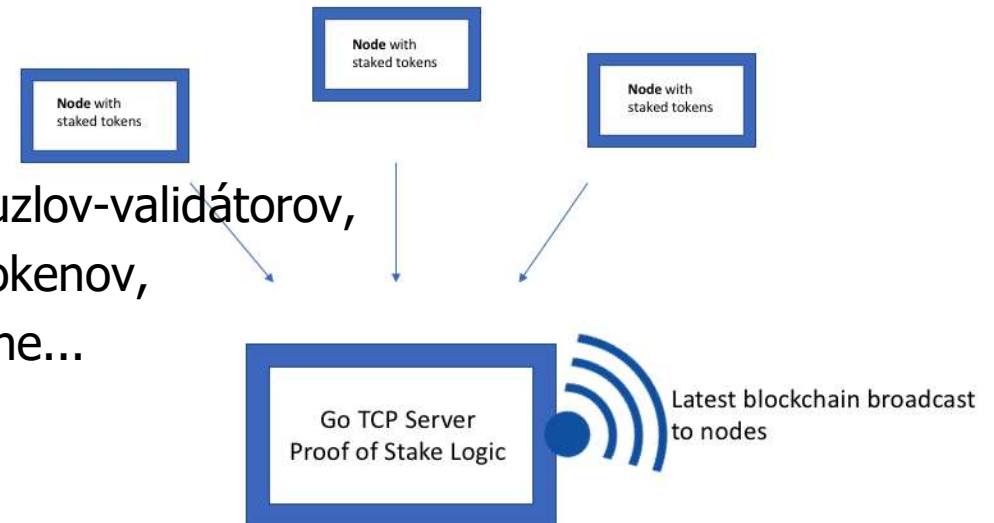
Proof of Stake



- napr. Ethereum s projektom Casper
- čím viac (tokenov) každý uzol vsadí do transakcie, tým väčšiu šancu na výhru má
- keďže nevyhráva veľkosť HW, máte šancu aj s laptopom, ak máte dost tokenov

Implementácia zjednodušuje:

- jeden blockchain server-veľa uzlov-validátorov,
- nemáme vlastnú peňaženku tokenov, takže môžeme vsadiť ľubovoľne...
- a väčší čatejšie berie...



Výber víťaza

- validátor navrhuje nový blok a vsadí do toho nejaký počet vlastných tokenov
- víťaz kola sa určuje náhodne, podľa váh vsadených tokenov
- blok víťaza sa zapíše do blockchainu, ten sa broadcastuje všetkým

