



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
ELEKTRONINIŲ SISTEMŲ KATEDRA

BLOCKCHAIN TECHNOLOGIJA

Atliko: AKSfm-16 grupės studentas Andrius Adamonis

Vilnius, 2017

BLOCKCHAIN TINKLO KŪRIMAS

LiteCoin blockchain tinklas yra kuriamas, kompiliuojamas ir paleidžiamas Linux operacinėje sistemoje. Norint, jog operacinėje sistemoje pavyktų atlikti blockchain kodo kompiliavimą reikia paruošti operacinę sistemą ir į ją įrašyti bibliotekas. Komandinėje eilutėje (terminal) vedame tokį kodą:

- *sudo apt install g++*
- *sudo apt-get install libboost-all-dev*
- *sudo apt-get install libssl-dev*
- *sudo apt-get install libdb++-dev*

Tokiu pat būdu įrašome ir kitus Linux operacinei sistemai reikalingas bibliotekas ir įskiepius (tools):

- *sudo apt-get install build-essential libtool autotools-dev autoconf pkg-config libssl-dev*
- *sudo apt-get install libboost-all-dev*
- *sudo add-apt-repository ppa:bitcoin/bitcoin*
- *sudo apt-get update*
- *sudo apt-get install libminiupnpc-dev*

Paruošus operacinę sistemą darbui su blockchain tinklu parisiunčiame Litecoin .zip failą ir Litecoin github (<https://github.com/litecoin-project/litecoin/releases>) versija 0.8.7.4

Sekantis žingsnis, pervadiname visus Litecoin į savo pasirinktą vardą. Šiam veiksmui atlikti komandinėje eilutėje vedame tokias komandas:

- *find ./ -type f -readable -writable -exec sed -i "s/Litecoin/Venuscoin/g" {} \;*
- *find ./ -type f -readable -writable -exec sed -i "s/LiteCoin/VenusCoin/g" {} \;*
- *find ./ -type f -readable -writable -exec sed -i "s/LTC/VEN/g" {} \;*
- *find ./ -type f -readable -writable -exec sed -i "s/litecoin/venuscoin/g" {} \;*
- *find ./ -type f -readable -writable -exec sed -i "s/litecoind/venuscoind/g" {} \;*

Atlikus šiuos pakeitimus, turime pakeitę visus pavadinimus, tinklo parametrai išliko tokie patys. Mums būtina pakeisti genesis block hash value – motinio bloko hash'o reikšmę; merkle root – merkle šaknį ir nNonce laiką. Šie parametrai suteikia mūsų tinklui unikalumo. Šiuos pakeitimus atliksime main.cpp, net.cpp ir checkpoints.cpp failuose.

Pirmasis žingsnis norint padaryti tinklą unikaliu yra pakeisti pradžios žinučių reikšmes. Main.cpp faile pakeičiame:

```
bool LoadBlockIndex()
{
    if (fTestNet)
    {
        pchMessageStart[0] = 0xd2; // (čia keista)
        pchMessageStart[1] = 0xc3; // (čia keista)
        pchMessageStart[2] = 0xb7;
        pchMessageStart[3] = 0xdc;
        hashGenesisBlock = uint256("0x"); // (čia keisim į mūsų genesis block hash value)
    }
}
```

Antras žingsnis yra ištrinti visas esamas Litecoin motininio bloko reikšmes. Visame main.cpp faile ieškome „HashGenesisBlock“ ištriname reikšmes iki „0x“, kaip tai padaryta viršuje pateiktame pavyzdyje (hashGenesisBlock = uint256("0x")); // (cia keisim į mūsų genesis block hash value)).

Trečias žingsnis yra atsikratyti laiko žymės (timestamp), tiksliau tariant ją pakeičiame sava. Taip pat vietą, kurioje būna generuojama motininio bloko hash reikšmė pakeičiame įvesdami bet kokius ženklus, jog hash generuojama reikšmė pasikeistų. Šioje funkcijoje aš pridėjau ženklus: 1234.

// Genesis block

```
const char* pszTimestamp = "This is VenusCoin"; // (čia atlikom pakeitimą)
CTransaction txNew;
txNew.vin.resize(1);
txNew.vout.resize(1);
txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned
char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp +
strlen(pszTimestamp));
txNew.vout[0].nValue = 50 * COIN;
txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("123484710fa689ad5023690c80f3a49c8f13f8d45b8c857fbc8bc4a8e4d3eb4b10f4d4604fa
08dce601aaf0f470216fe1b51850b4acf21b179c45070ac7b03a9") << OP_CHECKSIG; // (čia atlikom
pakeitimą)
```

Ketvirtas žingsnis, pakeičiame tinklo parametrus. Tai yra bloko sukūrimo laiką ir vėliau įrašysime pirmą kartą paleidžiant tinklą debug.log faile atsiradusią nNonce reikšmę.

CBlock block;

```
block.vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
block.nVersion = 1;
block.nTime = 1495778930; // (Čia atlikome pakeitimus)
block.nBits = 0x1e0ffff0;
block.nNonce = 0; // (čia įkelsime duomenis sugeneruotus debug.log faile)
if (fTestNet)
{
    block.nTime = 1495778930; // (čia atlikome pakeitimą)
    block.nNonce = 176860; // (čia atlikome pakeitimą)
}
```

Penktas žingsnis ištriname senąsias Litecoin merkle šaknis. Jas taip pat užpildome „0x“ o jų tikroji, mums reikiama reikšmė bus sugeneruota debug.log faile.

//// debug print

```
uint256 hash = block.GetHash();
printf("%s\n", hash.ToString().c_str());
printf("%s\n", hashGenesisBlock.ToString().c_str());
printf("%s\n", block.hashMerkleRoot.ToString().c_str());
assert(block.hashMerkleRoot == uint256("0x")); // (ištrynėme merkle root)
```

Šeštas žingsnis pakeičiame bloko patvirtinimo laiką:

```
static const int64 nTargetTimespan = 10 * 30; // Venuscoin: 5 minutes (cia atlikome pakeitimus)
```

```
static const int64 nTargetSpacing = 1 * 30; // Venuscoin: 30 sec (cia atlikome pakeitimus)
```

```
static const int64 nInterval = nTargetTimespan / nTargetSpacing;
```

Toliau atsikratome visų egzistuojančių LiteCoin checkpoint'ų. Atliekame pakeitimus checkpoint.cpp faile.

```
static MapCheckpoints mapCheckpoints =
```

```
    boost::assign::map_list_of
```

```
    ( 0, uint256("0x")) // (šioje vietoje ištrynėme labai daug tokio pat stiliaus eilučių ir palikome tik vieną, taip pat ištriname genesis block hash value)
```

```
    ;
```

```
static const CCheckpointData data = {
```

```
    &mapCheckpoints,
```

```
    //1410516073, // (užkomentavome šią eilutę) * UNIX timestamp of last checkpoint block
```

```
    //4896865, // (užkomentavome šią eilutę) * total number of transactions between genesis and last checkpoint
```

```
    // (the tx=... number in the SetBestChain debug.log lines)
```

```
    //7000.0 // (užkomentavome šią eilutę) * estimated number of transactions per day after checkpoint
```

```
};
```

```
static MapCheckpoints mapCheckpointsTestnet =
```

```
    boost::assign::map_list_of
```

```
    ( 0, uint256("0x")) // (ištrynėme genesis block hash reikšmę)
```

```
    ;
```

```
static const CCheckpointData dataTestnet = {
```

```
    &mapCheckpointsTestnet,
```

```
    // 1365458829, // (užkomentavome šią eilutę)
```

```
    // 547, // (užkomentavome šią eilutę)
```

```
    // 576 // (užkomentavome šią eilutę)
```

```
};
```

Atliekame pakeitimus net.cpp faile. Šiame faile nurodžiau savo virtualų serverį (localhost, ip 127.0.0.1). Šioje vietoje reikėtų įrašyti serverio kuriame bus kodas ir į kurį kreipsis adresas.

```
static const char *strMainNetDNSSeed[][2] = {
```

```
    {"venuscointools.com", "127.0.0.1"}, // (ištrynėme keltą tokių pat eilučių ir palikome tik vieną kur nuročiau localhost serverio adresą) cia vedam adresa valiutos
```

```
    {NULL, NULL}
```

```
};
```

Toliau pakeičiame qt.pro failą, visa tai atliekame Linux komandinėje eilutėje, komanda: mv bitcoin-qt.pro venuscoin-qt.pro

Labai svarbus naujo blockchain tinklo kūrimo procesas naujos merkle šaknies generavimo kodas. Ši kodą talpiname main.cpp faile.

```
if (true && block.GetHash() != hashGenesisBlock)
```

```

{
    printf("Searching for genesis block...\n");
    // This will figure out a valid hash and Nonce if you're
    // creating a different genesis block:
    uint256 hashTarget = CBigNum().SetCompact(block.nBits).getuint256();
    uint256 thash;
    char scratchpad[SCRIPT_SCRATCHPAD_SIZE];

    loop
    {
        script_1024_1_1_256_sp(BEGIN(block.nVersion), BEGIN(thash), scratchpad);
        if (thash <= hashTarget)
            break;
        if ((block.nNonce & 0xFFF) == 0)
        {
            printf("nonce %08X: hash = %s (target = %s)\n", block.nNonce, thash.ToString().c_str(),
hashTarget.ToString().c_str());
        }
        ++block.nNonce;
        if (block.nNonce == 0)
        {
            printf("NONCE WRAPPED, incrementing time\n");
            ++block.nTime;
        }
    }
    printf("block.nTime = %u \n", block.nTime);
    printf("block.nNonce = %u \n", block.nNonce);
    printf("block.GetHash = %s\n", block.GetHash().ToString().c_str());
}

```

Atliekame pirmąjį kompiliavimą. Atlikus kompiliavimą sėkmingai src folderyje turėtų atsirasti venuscoind failas, kuris reikalingas tinklo paleidimui. Tai patikriname src folderyje suvedę komandą *ls*. Kompiliuojame komandinėje eilutėje vesdami komandą: ***make -f makefile.unix USE//_UPNP=-***

```

blockchain@ubuntu: ~/venuscoin/src
blockchain@ubuntu:~/venuscoin$ cd src
blockchain@ubuntu:~/venuscoin/src$ ls
addrman.cpp      hash.cpp          netbase.cpp       scrypt-sse2.cpp
addrman.h        hash.h            netbase.h         serialize.h
alert.cpp        init.cpp          net.cpp           sync.cpp
alert.h          init.h            net.cpp~          sync.h
allocators.h     icon              net.h             test
base58.h         key.cpp           noui.cpp          threadsafety.h
bignum.h         key.h             obj               txdb.cpp
bitcoinrpc.cpp   keystore.cpp      obj-test          txdb.h
bitcoinrpc.h     keystore.h        protocol.cpp      ui_interface.h
bloom.cpp        leveldb.cpp       protocol.h        uint256.h
bloom.h          leveldb.h         rpcblockchain.cpp util.cpp
checkpoints.cpp  limitedmap.h      rpcdump.cpp       util.h
checkpoints.h    main.cpp          rpcmining.cpp     venuscoind
checkqueue.h     main.cpp~         rpcnet.cpp        version.cpp
clientversion.h  main.h            rpcwallet.cpp     version.h
coincontrol.h    makefile.linux-mingw  rpcwtransaction.cpp wallet.cpp
compat.h         makefile.mingw      script.cpp         walletdb.cpp
crypter.cpp      makefile.osx        script.h           walletdb.h
crypter.h        makefile.unix       scrypt.cpp        wallet.h
db.cpp           mruset.h           scrypt.h
db.h
blockchain@ubuntu:~/venuscoin/src$

```

Pav. 6. Pirmojo kompiliavimo rezultatas

Paleidžiame tinklą, komandinėje eilutėje vedame komandą: ./venuscoind
Gauname klaidą:

```

andrius@ubuntu: ~/venuscoin/src
-D_FORTIFY_SOURCE=2 -MMD -MF obj/txdb.d -o obj/txdb.o txdb.cpp
g++ -O2 -pthread -Wall -Wextra -Wformat -Wformat-security -Wno-unused-parameter
-g -DBOOST_SPIRIT_THREADSafe -D_FILE_OFFSET_BITS=64 -I/home/andrius/venuscoin/src
-I/home/andrius/venuscoin/src/obj -DUSE_IPV6=1 -I/home/andrius/venuscoin/src/level
ldb/include -I/home/andrius/venuscoin/src/leveldb/helpers -DHAVE_BUILD_INFO -D
fno-stack-protector -fstack-protector-all -Wstack-protector -U_FORTIFY_SOURCE -D
_FORTIFY_SOURCE=2 -o venuscoind leveldb/libleveldb.a obj/alert.o obj/version.o
obj/checkpoints.o obj/netbase.o obj/addrman.o obj/crypter.o obj/key.o obj/db.o o
bj/init.o obj/keystore.o obj/main.o obj/net.o obj/protocol.o obj/bitcoinrpc.o ob
j/rpcdump.o obj/rpcnet.o obj/rpcmining.o obj/rpcwallet.o obj/rpcblockchain.o obj
/rpcwtransaction.o obj/script.o obj/scrypt.o obj/sync.o obj/util.o obj/wallet.
o obj/walletdb.o obj/hash.o obj/bloom.o obj/noui.o obj/leveldb.o obj/txdb.o -Wl,
-z,relro -Wl,-z,now -Wl,-Bdynamic -l boost_system -l boost_filesystem -l boost
_program_options -l boost_thread -l db_cxx -l ssl -l crypto -Wl,-Bdynamic -l z -
l dl -l pthread /home/andrius/venuscoin/src/leveldb/libleveldb.a /home/andrius/v
enuscoin/src/leveldb/libmemenv.a
andrius@ubuntu:~/venuscoin/src$ cd
andrius@ubuntu:~$ cd venuscoin
andrius@ubuntu:~/venuscoin$ cd src
andrius@ubuntu:~/venuscoin/src$ ./venuscoind
venuscoind: main.cpp:2809: bool InitBlockIndex(): Assertion `block.hashMerkleRoot
 == uint256("0x")' failed.
Aborted (core dumped)
andrius@ubuntu:~/venuscoin/src$

```

Pav. 7. Pirmasis tinklo paleidimas (nesėkmingas)

Kompiliuojant buvo sukurtas paslėptas (hidden) folderis (.venuscoin). Taškas prieš pavadinimą reiškia, kad jis yra paslėptas. Atsidarome paslėptą folderį ir jame atsidarome debug.log failą. Šio failo apačioje yra sukurta merkle root reikšmė.

Įkeliamo merkle root į main.cpp failą:

```
assert(block.hashMerkleRoot ==  
uint256("0x541c9c1d1fe02385a9e47214879cb9050fc64574ac275cffbdbd673fd5da274c")); // (mūsų  
debug.log faile sugeneruota nauja merkle root reikšmė)
```

Kompiliuojam antrą kartą. Komandinėje eilutėje vedame komandą: **make -f makefile.unix USE_UPNP=-** Vėl gauname klaidą. Klaidos esmė, jog kompiuteris bandė kurti (minint) motininį bloką, kurio hash reikšmės mes nenurodėme main.cpp faile, toje vietoje palikome tik įrašą 0x. Tačiau debug.log faile gavome informaciją, kurios mums reikia, tai yra nNonce ir genesis block hash reikšmės.

GENESIS: 3abc3fb42e89d52eb6d9ad5a9c41e5a6ec0e5c4dc25f55c8dd1f069f719e4

nNonce: 176860

Šias reikšmes įrašome į checkpoints.cpp:

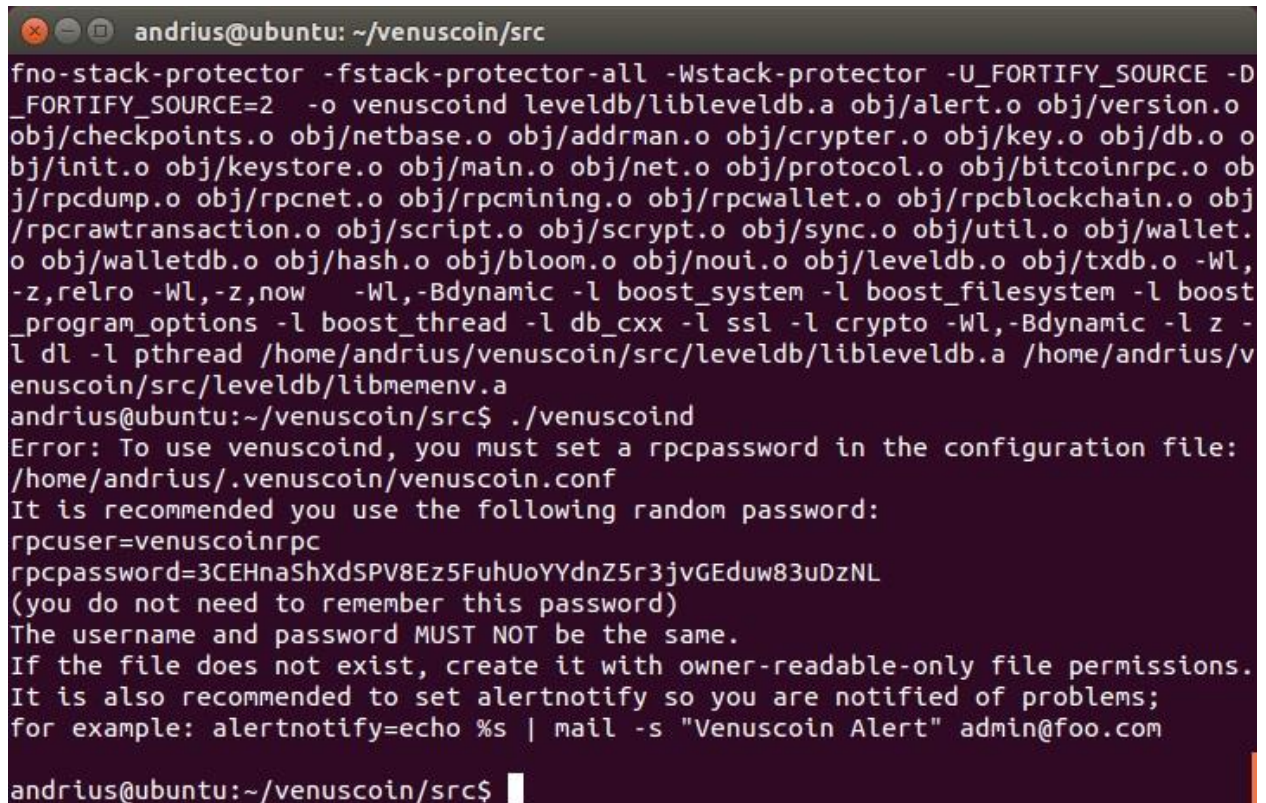
```
static MapCheckpoints mapCheckpoints =  
    boost::assign::map_list_of  
        ( 0, uint256("0x3abc3fb42e89d52eb6d9ad5a9c41e5a6ec0e5c4dc25f55c8dd1f069f719e4")) //  
        (genesis block hash value)  
    ;  
Ir main.cpp:  
map<uint256, CBlockIndex*> mapBlockIndex;  
uint256  
hashGenesisBlock("0x3abc3fb42e89d52eb6d9ad5a9c41e5a6ec0e5c4dc25f55c8dd1f069f719e4");  
// (GenesisBlockHashValue)  
<.....>  
if (fTestNet)  
{  
    pchMessageStart[0] = 0xd2; // (čia keitim)  
    pchMessageStart[1] = 0xc3; // (čia keitim)  
    pchMessageStart[2] = 0xb7;  
    pchMessageStart[3] = 0xdc;  
    hashGenesisBlock =  
uint256("0x3abc3fb42e89d52eb6d9ad5a9c41e5a6ec0e5c4dc25f55c8dd1f069f719e4"); //  
    (GenesisBlockHashValue)  
}  
<.....>  
    block.hashMerkleRoot = block.BuildMerkleTree();  
    block.nVersion = 1;  
    block.nTime = 1495778930; // (čia keitim)
```

```

    block.nBits    = 0x1e0ffff0;
    block.nNonce   = 176860; //(nnonce)
if (fTestNet)
{
    block.nTime    = 1495778930; // (čia keitim)
    block.nNonce   = 176860; //(nnonce)
}

```

Kompiliuojant trečią kartą (*make -f makefile.unix USE_UPNP=-*) ir vėl paleidžiame serverį. Ši kartą gavome klaidą, jog trūksta .conf failo:



```

andrius@ubuntu: ~/venuscoin/src
fno-stack-protector -fstack-protector-all -Wstack-protector -U_FORTIFY_SOURCE -D
_FORTIFY_SOURCE=2 -o venuscoind leveldb/libleveldb.a obj/alert.o obj/version.o
obj/checkpoints.o obj/netbase.o obj/addrman.o obj/crypter.o obj/key.o obj/db.o o
bj/init.o obj/keystore.o obj/main.o obj/net.o obj/protocol.o obj/bitcoinrpc.o ob
j/rpcdump.o obj/rpcnet.o obj/rpcmining.o obj/rpcwallet.o obj/rpcblockchain.o obj
/rpcrawtransaction.o obj/script.o obj/scrip.o obj/sync.o obj/util.o obj/wallet.
o obj/walletdb.o obj/hash.o obj/bloom.o obj/noui.o obj/leveldb.o obj/txdb.o -Wl,
-z,relro -Wl,-z,now -Wl,-Bdynamic -l boost_system -l boost_filesystem -l boost
_program_options -l boost_thread -l db_cxx -l ssl -l crypto -Wl,-Bdynamic -l z -
l dl -l pthread /home/andrius/venuscoin/src/leveldb/libleveldb.a /home/andrius/v
enuscoin/src/leveldb/libmemenv.a
andrius@ubuntu:~/venuscoin/src$ ./venuscoind
Error: To use venuscoind, you must set a rpcpassword in the configuration file:
/home/andrius/.venuscoin/venuscoin.conf
It is recommended you use the following random password:
rpcuser=venuscoinrpc
rpcpassword=3CEHnaShXdSPV8Ez5FuhUoYYdnZ5r3jvGEduw83uDzNL
(you do not need to remember this password)
The username and password MUST NOT be the same.
If the file does not exist, create it with owner-readable-only file permissions.
It is also recommended to set alertnotify so you are notified of problems;
for example: alertnotify=echo %s | mail -s "Venuscoin Alert" admin@foo.com
andrius@ubuntu:~/venuscoin/src$

```

Pav. 8. Nesėkmingas kompiliavimas, trūksta .conf failo

Paslėptame folderyje .venuscoin sukuriame venuscoin.conf failą su tokiais parametrais:

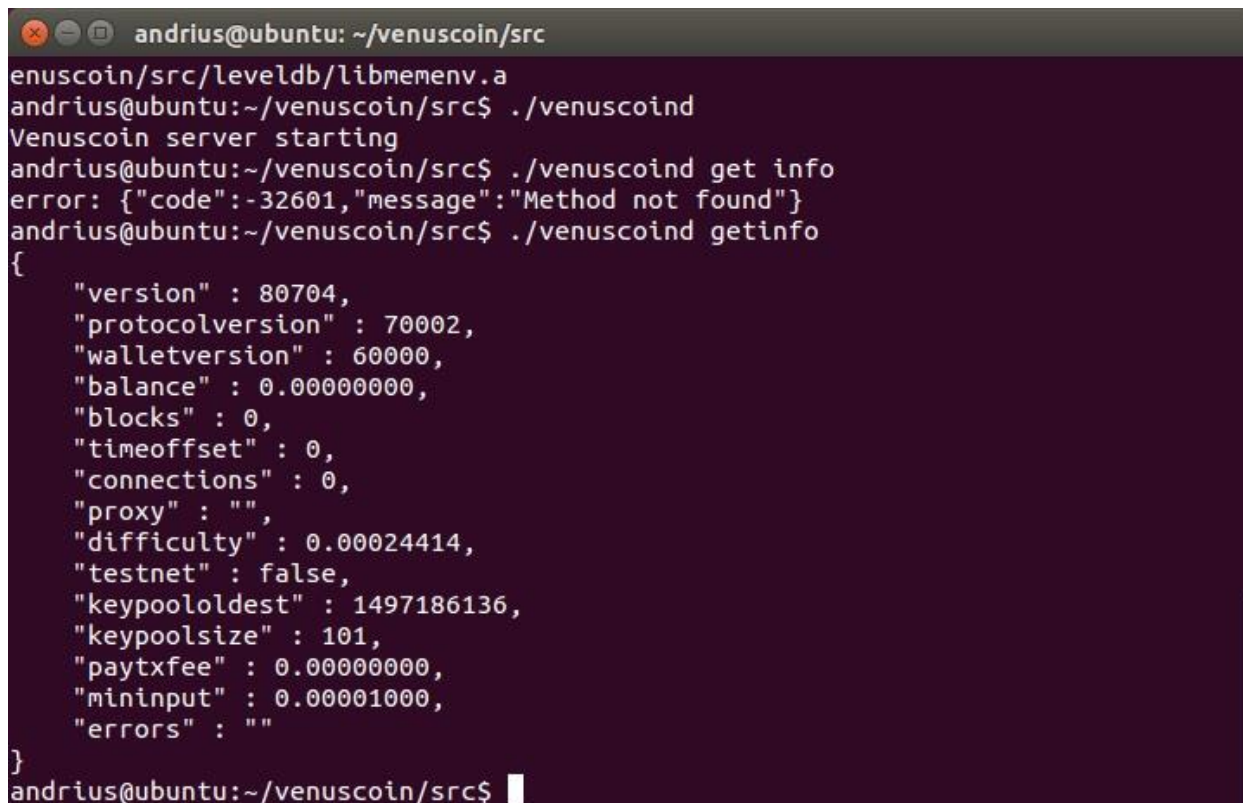
```

rpcuser=username
rpcpassword=password
rpcallowip=127.0.0.1
rpcport=9332
daemon=1
server=1
gen=0

block_nTime=1317972665
block_nNonce=2084524493

```


Kompiliuojame ketvirtą kartą (make -f makefile.unix USE_UPNP=-), vėl paleidžiame tinklą (./venuscoind) komandiniame lange matome jog tinklas pasileido sėkmingai „server starting“. Komandinėje eilutėje vedame komandą: ./venuscoind getinfo ir sužinome informaciją apie tinklą.



```
andrius@ubuntu: ~/venuscoin/src
enuscoin/src/leveldb/libmemenv.a
andrius@ubuntu:~/venuscoin/src$ ./venuscoind
Venuscoin server starting
andrius@ubuntu:~/venuscoin/src$ ./venuscoind get info
error: {"code":-32601,"message":"Method not found"}
andrius@ubuntu:~/venuscoin/src$ ./venuscoind getinfo
{
  "version" : 80704,
  "protocolversion" : 70002,
  "walletversion" : 60000,
  "balance" : 0.00000000,
  "blocks" : 0,
  "timeoffset" : 0,
  "connections" : 0,
  "proxy" : "",
  "difficulty" : 0.00024414,
  "testnet" : false,
  "keypoololdest" : 1497186136,
  "keypoolsize" : 101,
  "paytxfee" : 0.00000000,
  "mininput" : 0.00001000,
  "errors" : ""
}
andrius@ubuntu:~/venuscoin/src$
```

Pav. 9. Sėkmingai paleistas blockchain tinklas

Gavus tokį užrašą, reikia atlikti bene patį svarbiausią tinklo kūrimo žingsnį, padaryti decentralizaciją. Decentralizacijai sukurti naudojame gitian-builder.

Gitian yra saugi į šaltinį orientuota programinė įranga išskirstytumo metodui. Tai reiškia, mes galime parsisiųsti patikimus dvejetainius kodus (binaries), kurie yra patvirtinti daugelio tinklo dalyvių, kūrėjų (builders).

Gitian naudoja deterministinį kūrimo procesą leisti keletui kūrėjų kurti vienodus dvejetainius kodus (binaries). Tai leidžia skirtingiems dalyviams patvirtinti, jog medžiaga buvo naudojama, kuri yra kilusi iš to paties šaltinio. Gitian-builder panaikina vieno klaidos vietos galimybę (single point of failure).

Taigi galime teigti, jog gitian builderiai sukuria decentralizaciją.

1. Kad sukurtume gitian-builder instaliuojame apache virtualią mašiną:

sudo apt-get install git apache2 apt-cacher-ng python-vm-builder ruby

2. Kolonuojam gitian-builder ir gitian-builder github. Tai darome komandinėje eilutėje vesdami komandą: ***git clone <https://github.com/devrandom/gitian-builder.git>***
3. Parsiunčiam reikiamus failus, kurie aprašyti gitian-descriptors. Ten kaip pat aprašyta ir procesoriaus architektūra. Taip pat kai kurie, gitian parsisiųsti failai yra naujesni nei aprašyti šioje Litecoin versijoje, taigi deps-win32.yml aprašyme pakeičiame jų versijas. Gitian-win32.yml

aprašyti failai kuriuos turime turėti norint sukompiliuoti gitian-builderį. Tuos failus sukursime paleisdami kiekvieną iš gitian-descriptors:

- "qt-win32-4.8.5-gitian-r7.zip"
- "boost-win32-1.55.0-gitian-r6.zip"
- "bitcoin08-deps-win32-gitian-r12.zip"

Kad sukompiliuotume šiuos failus turime žiūrėti į deps-win32.yml, ten aprašyti failai, kuriuos reikia parsisiųsti, kad būtų sukompiliuoti šie trys aukščiau išvardinti failai.

Tam folderyje gitian-builder/inputs turime turėti tokius failus:

- "openssl-1.0.1i.tar.gz"
- "db-4.8.30.NC.tar.gz"
- "miniupnpc-1.9.20140401.tar.gz"
- "zlib-1.2.11.tar.gz"
- "libpng-1.6.8.tar.gz"
- "qrencode-3.4.3.tar.bz2"

Juos parsisiunčiame iš interneto.

4. Paleidžiame virtualią mašiną gitian-builder foldery: `sudo bin/make-base-vm --suite precise --arch i386`
5. Patikrinam, kad mūsų kodo (venuscoin) folderis būtų gitian-builder/inputs foldery
Komandinėje eilutėje vedame tokią komandą `sudo bin/gbuild ../venuscoin/contrib/gitian-descriptors/deps-win32.yml`, skirtą sukurti "boost-win32-1.55.0-gitian-r6.zip" failui.

IŠVADOS

Šiuo metu manoma, kad blockchain technologija turi milžinišką potencialą būti pritaikoma įvairiose srityse, tokiose kaip sveikatos priežiūros sistema, registruose, ypatingai didelis potencialas išvelgiamas finansinėse technologijose. Aspektai kurie žavi ir padaro šią technologiją tokia patrauklia yra decentralizacija, nepaneigiamumas, kriptografinis saugumas ir tai jog nėra vieno klaidos vektoriaus. Nepaisant visų pliusų ši technologija yra labai ankstyvoje vystymosi stadijoje ir kol kas sėkmingiausias jos pritaikymo atvejis yra tik vienas Bitcoin'as. Ši technologija kol kas nėra pritaikyta dideliems duomenims, pakankamai lėta. Tačiau į šios technologijos vystymą ir tyrimus investuoja didžiausios pasaulio kompanijos bei valstybės, todėl jos tobulėjimas vyksta labai greitai ir pritaikomumas šiuolaikiniams duomenų kiekiams ar greičiams atrodo vis artimesnis.

Beveik visi šiuo metu veikiantys blockchain tinklai/platformos yra atvirojo kodo, taip jų tobulėjimas vysta dar sparčiau. Pasinaudojęs pagal atvirojo kodo licenciją platinamu Litecoin blockchain tinklu, paėmiau šaltinio kodą ir sukūriau savo visiškai naują blockchain tinklą. Kuriame galėjau laisvai keisti parametrus, optimizuoti jo veikimą siekdamas pritaikyti savo poreikiams.

Kadangi technologija yra pačiame pike, o jos pritaikomumas pasieks savo klestėjimo laikotarpį 5-10 metų laikotarpyje, artimiausiu metu išvysime vis daugiau ir daugiau sistemų, kurių veikimas bus perkeliamas į blockchain tinklą ir galėsime stebėti spartų technologijos vystymąsi.

LITERATŪROS SĄRAŠAS

1. Imran Bashir „Mastering Blockchain“
2. Blockchain-Putting Theory into Practice. Goldman Sachs report
3. https://hyperledger-fabric.readthedocs.io/en/latest/build_network.html
4. <https://github.com/litecoin-project/litecoin/releases>
5. https://ripple.com/files/ripple_consensus_whitepaper.pdf
6. https://en.wikipedia.org/wiki/Smart_contract
7. <http://www.the-blockchain.com/docs/Hyperledger%20Whitepaper.pdf>