# BIT-ECC:
# Generate Genesis Block

Ha-young Park, Wahidur, Huisu Kim
8th January 2021
INFONET (Prof. Heung-No Lee), GIST
https://github.com/infonetGIST
https://infonet.gist.ac.kr/

Purpose: The aim of this document is to instruct the readers how to generate BIT-ECC genesis block of at local machine.

# Contents:


1. Change values of genesis block
2. Get hash values of a genesis block and Merkle-root
3. Execute blockchain core


# 1. Change values of genesis block

We will modify some parameters values that are related to genesis block. 'chainparams.cpp' file is used to modify values at '~/bitcoin_ECC/src' folder. In the 'chainparams.cpp', there are two override functions and three classes for each network. The override function is 'CreateGenesisBlock' and the classes of networks are mainnet, testnet(v3) and regression test.

### 1.1 Modify the 'CreateGenesisBlock' function:

Modify the value of ‘txNew.vout[0].nValue’ parameter as 50. It means that the reward point for generating a new block will be 50 BTC.

```
static CBlock CreateGenesisBlock(const char* pszTimestamp, const CScript& genesisOutputScript, uint32_t nTime, uint32_t nNonce, uint32_t nBits, int32_t
nVersion, const CAmount& genesisReward)
{
    CMutableTransaction txNew;
    txNew.nVersion = 1;
    txNew.vin.resize(1);
    txNew.vout.resize(1);
    txNew.vin[0].scriptSig = CScript() << 486604799 << CScriptNum(4) << std::vector<unsigned char>((const unsigned char*)pszTimestamp, (const unsigned
char*)pszTimestamp + strlen(pszTimestamp));
    txNew.vout[0].nValue = 50;
    txNew.vout[0].scriptPubKey = genesisOutputScript;

    CBlock genesis;
    genesis.nTime    = nTime;
    genesis.nBits    = nBits;
    genesis.nNonce   = nNonce;
    genesis.nVersion = nVersion;
    genesis.vtx.push_back(MakeTransactionRef(std::move(txNew)));
    genesis.hashPrevBlock.SetNull();
    genesis.hashMerkleRoot = BlockMerkleRoot(genesis);
    return genesis;
}
```

Modify the value of 'pszTimestamp' parameter. Write any sentence you want or just modify the date as current date.

```
static CBlock CreateGenesisBlock(uint32_t nTime, uint32_t nNonce, uint32_t nBits, int32_t nVersion, const CAmount& genesisReward)
{
    const char* pszTimestamp = "The Times 08/Jan/2021 New hope for Covid patients";
    const CScript genesisOutputScript = CScript() <<
ParseHex("04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61deb649f6bc3f4cef38c4f35504e51ec112de5c384df7ba0b8d578a4c702b6bf11d5f") <<
OP_CHECKSIG;
    return CreateGenesisBlock(pszTimestamp, genesisOutputScript, nTime, nNonce, nBits, nVersion, genesisReward);
}
```

### 1.2 Modify the parameters of 'CreateGenesisBlock' function:

Each class has 5 parameters- ‘nTime’, ‘nNonce’, ‘nBits’, ‘nVersion’ and ‘genesisReward’. Change the first and the second one [‘nTime’, ‘nNonce’]. For UNIX time please follow the mentioned link [https://www.unixtimestamp.com/ ] and put any random integer number for the ‘nNonce’ value.

```
int init_level = 1;
genesis = CreateGenesisBlock(1558627231, 399, init_level, 1, 50 * COIN);
```

```
int init_level = 1;
genesis = CreateGenesisBlock(1610072113, 1012, init_level, 1, 50 * COIN);
//printf("\nmainnet with level = %d\n",init_level);
//printf("set is constructed from %d to %d with step 2\n",ldpc_level_table[init_level].from, ldpc_level_table[init_level].to);
//printf("n : %d\t wc : %d\t wr : %d\n", ldpc_level_table[init_level].n, ldpc_level_table[init_level].wc, ldpc_level_table[init_level].wr);
```

Repeat this process for other networks. If 'init_level' is different with mainnet, change it same as mainnet.

[testnet(v3)]

```
int init_level = 1;
genesis = CreateGenesisBlock(1610072113, 1012, init_level, 1, 50 * COIN);
//printf("\ntestnet with level = %d\n",init_level);
//printf("set is constructed from %d to %d with step 2\n",ldpc_level_table[init_level].from, ldpc_level_table[init_level].to);
//printf("n : %d\t wc : %d\t wr : %d\n", ldpc_level_table[init_level].n, ldpc_level_table[init_level].wc, ldpc_level_table[init_level].wr);
```

[regression test]

```
int init_level = 1;
genesis = CreateGenesisBlock(1610072113, 1012, init_level, 1, 50 * COIN);
//printf("\nregtest with level = %d\n",init_level);
//printf("set is constructed from %d to %d with step 2\n",ldpc_level_table[init_level].from, ldpc_level_table[init_level].to);
//printf("n : %d\t wc : %d\t wr : %d\n", ldpc_level_table[init_level].n, ldpc_level_table[init_level].wc, ldpc_level_table[init_level].wr);
```

# 2. Get hash values of a genesis block and Merkle-root

We will enter the 'bitcoin core' to get new hash values of a genesis block and Merkle-root.

### 2.1 Modify hash values to pass assert () function:

Replace the source code for each class with the new code as written inside the box below.

```
consensus.hashGenesisBlock = genesis.GetHash();
assert(consensus.hashGenesisBlock == uint256S("01a92cfc6a9949ae39ae5c58918138414beedad0e5a4f921612bc182bef280d2"));
assert(genesis.hashMerkleRoot == uint256S("0873377ff3078f39a4f1470e7e15d1234c4be9980b41724e714fa953ba04824b"));
```

```
consensus.hashGenesisBlock = genesis.GetHash();
assert(consensus.hashGenesisBlock == genesis.GetHash();
assert(genesis.hashMerkleRoot == BlockMerkleRoot(genesis));
```

Save 'chainparams.cpp' file, compile it and execute the bitcoin core using the following commands.

```
1) $ cd bitcoin_ECC
2) $ ./autogen.sh
3) $ ./configure
4) $ make
5) $ sudo make install
6) $ bitcoind -txindex -daemon
```

```
hisu@hisu-virtual-machine:~/bitcoin_ECC$ bitcoind -txindex -daemon
Bitcoin server starting
```

### 2.2 Get hash values:

Get hash value of the genesis block which block height would be 0.

```
1) $ bitcoin-cli getblockhash 0
```

Use the hash value of the genesis block to get Merkle-root hash value.

```
1) $ bitcoin-cli getblock
b71a7c0e8cfb0e11b14c8625ed65e4a60f270009a6114530fb9edb5718782
458
```

```
hisu@hisu-virtual-machine:~/bitcoin_ECC$ bitcoin-cli getblockhash 0
b71a7c0e8cfb0e11b14c8625ed65e4a60f270009a6114530fb9ebd5718782458
hisu@hisu-virtual-machine:~/bitcoin_ECC$ bitcoin-cli getblock b71a7c0e8cfb0e11b14c8625ed65e4a60f270009a6114530fb9ebd5718782458
{
  "hash": "b71a7c0e8cfb0e11b14c8625ed65e4a60f270009a6114530fb9ebd5718782458",
  "confirmations": 1,
  "strippedsize": 265,
  "size": 265,
  "weight": 1060,
  "height": 0,
  "version": 1,
  "versionHex": "00000001",
  "merkleroot": "2bb12aa4c91367fabe71acfe5b763b7c2070944953d7c7b228a0ce541867a098",
  "tx": [
    "2bb12aa4c91367fabe71acfe5b763b7c2070944953d7c7b228a0ce541867a098"
  ],
  "time": 1610072113,
  "mediantime": 1610072113,
  "nonce": 1035,
  "bits": "00000001",
  "difficulty": 4.523059468369196e+74,
  "chainwork": "0000000000000000000000000000000000000000000000000000000000007ee8",
  "nTx": 1
}
```

Save both the hash values [genesis block and Merkle-root].

### 2.3 Replace parts with hash values:

```
consensus.hashGenesisBlock = genesis.GetHash();
assert(consensus.hashGenesisBlock == genesis.GetHash());
assert(genesis.hashMerkleRoot == BlockMerkleRoot(genesis));
```

Replace the source code for each class with the new hash values as written inside the box.

```
consensus.hashGenesisBlock = genesis.GetHash();
assert(consensus.hashGenesisBlock ==
uint256S(''b71a7c0e8cfb0e11b14c8625ed65e4a60f270009a6114530fb
9edb5718782458''));
assert(genesis.hashMerkleRoot ==
uint256S(''2bb12aa4c91367fabe71acfe5b763b7c2070944953d7c7b228
a0ce541867a098''));
```

# 3. Execute blockchain core

### 3.1 Compile and execute bitcoin server:

Compile the code again using same procedures as mentioned in 2.1

### 3.2 Check the number of blocks in the blockchain:

First command line returns the number of blocks and Second one returns the information of the blockchain we accessed.

1) `$ bitcoin-cli getblockcount`
2) `$ bitcoin-cli getblockchaininfo`

```
hisu@hisu-virtual-machine:~/bitcoin_ECC$ bitcoind -txindex -daemon
Bitcoin server starting
hisu@hisu-virtual-machine:~/bitcoin_ECC$ bitcoin-cli getblockcount
0
hisu@hisu-virtual-machine:~/bitcoin_ECC$ bitcoin-cli getblockchaininfo
{
  "chain": "main",
  "blocks": 0,
  "headers": 0,
  "bestblockhash": "b71a7c0e8cfb0e11b14c8625ed65e4a60f270009a6114530fb9ebd5718782458",
  "difficulty": 4.523059468369196e+74,
  "mediantime": 1610072113,
  "verificationprogress": 1,
  "initialblockdownload": false,
  "chainwork": "0000000000000000000000000000000000000000000000000000000007ee8",
  "size_on_disk": 273,
  "pruned": false,
  "softforks": [
    {
      "id": "bip34",
      "version": 2,
      "reject": {
        "status": false
      }
    },
    {
      "id": "bip66",
```