POPCHAIN Technical Features Module Document

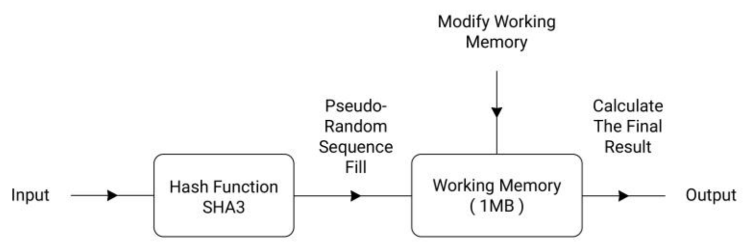
Version：	V1.0.2	No：	TH-TL-POP-01
Security level：	A+	Department/Project:	R&D Department
Product：	POPCHAIN Technical Features Module Document	Subsystem:	

Contents

I. PoW algorithm: Cryptopop

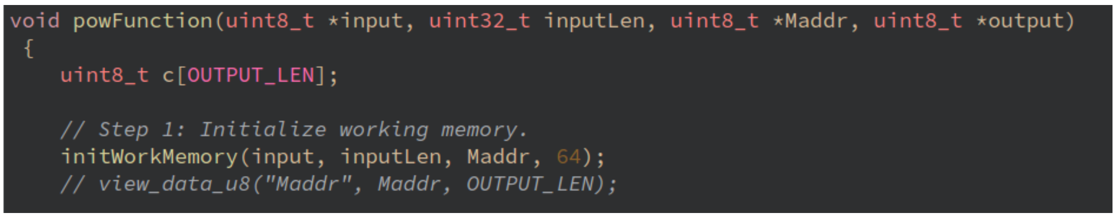
POPCHAIN designs a Proof of Work - Cryptopop, a CPU friendly architecture, but NOT to GPU, FPGA or ASIC. This is very crucial. POPCHAIN, as a public block chain of a pan-entertainment ecosystem, aims to involve every game player, video audience, and consumers. The audience of POPCHAIN should be the general public, not specific individuals or a team. Therefore, overcoming the operation of a large-scale mining using GPU, FPGA, and ASIC is a precondition to allow everyone to participate. Currently, SHA256 used by Bitcoin, X11 used by Dash, and Scrypt algorithm used by Litecoin already have corresponding ASIC mining rigs. Ordinary machines cannot compete with ASIC mining rigs, which will lead to the loss of mass users. It is inappropriate for POPCHAIN to use such algorithms. Therefore, the Cryptopop is born.

We have learned from the memory-hard PoW algorithm design. The method has 3 steps: use a pseudo-random sequence to initialize the working memory, modify the working memory, and produce the final result based on the working memory content.



[PoW algorithm: Cryptopop]

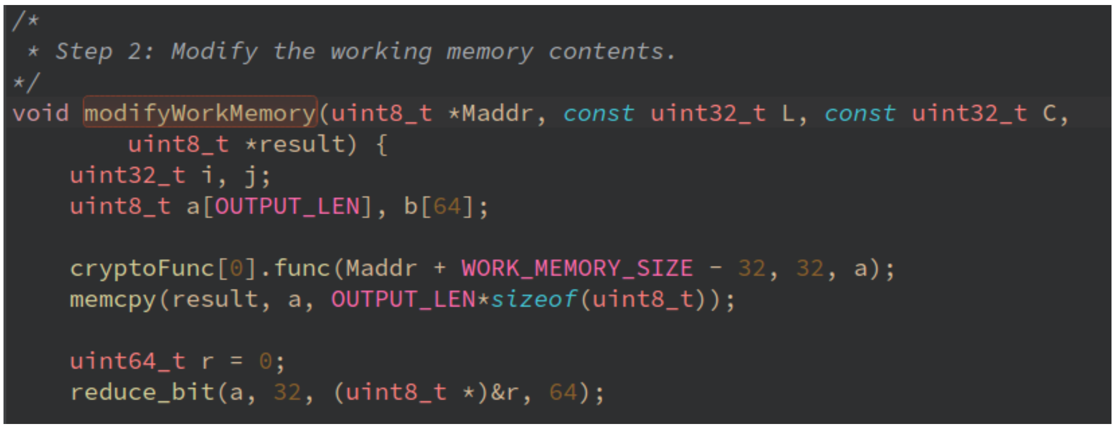
During the initialization of the working memory, an input is calculated using the SHA3 function. It fills 32-byte vector a [0:31] and initializes the linear congruence pseudo-random number generator seed. The working memory is continuously filled with units of K blocks (32 bytes per block), where the former K-1 block is generated by a pseudo-random number generator, combined with 4 8-byte vectors. The last block is filled with a random selection of one of the hash function families (16 different hash functions) based on the vector and the previously filled content and updates the random number generator seed. We can increase the filling speed by increasing k to reduce the number of one-way function calls. This step must be (1) executed in strict order, and (2) the factors that determine the content of each 32-byte memory block should be unpredictable and difficult enough to prevent pre-calculation.



```
void powFunction(uint8_t *input, uint32_t inputLen, uint8_t *Maddr, uint8_t *output)
 {
    uint8_t c[OUTPUT_LEN];

    // Step 1: Initialize working memory.
    initWorkMemory(input, inputLen, Maddr, 64);
    // view_data_u8("Maddr", Maddr, OUTPUT_LEN);
```

[The k is 64]

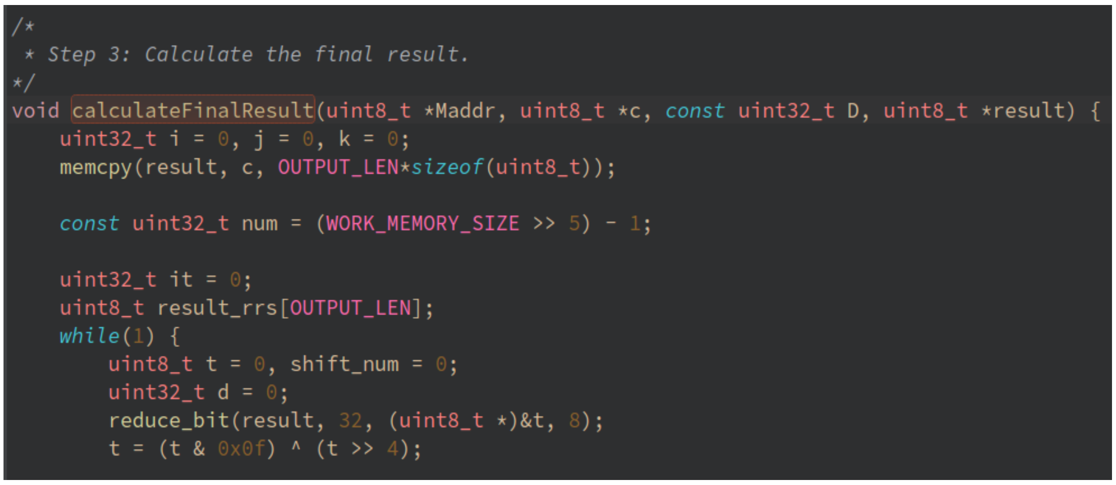
At the memory modification stage, the SHA3 function generates a seed of the random number generator based on the last piece of the working memory and initializes a state variable whose length is L-bytes. This is followed by C main cycles, where L subcycles are performed. In a subcycle, a pair of addresses is generated by the random number generator. The byte data in the two addresses in the working memory and the byte data in the state vector are combined and exchanged using the XOR method to update the memory content and the state vector content. After the end of a subcycle, the function vector of the hash function family is used to update the state vector and re-initialize the random number generator seed. The existing parameter design states that one quarter of the working memory will be modified.



```
/*
 * Step 2: Modify the working memory contents.
*/
void modifyWorkMemory(uint8_t *Maddr, const uint32_t L, const uint32_t C,
        uint8_t *result) {
    uint32_t i, j;
    uint8_t a[OUTPUT_LEN], b[64];

    cryptoFunc[0].func(Maddr + WORK_MEMORY_SIZE - 32, 32, a);
    memcpy(result, a, OUTPUT_LEN*sizeof(uint8_t));

    uint64_t r = 0;
    reduce_bit(a, 32, (uint8_t *)&r, 64);
```

[Modification of the Memory]

In the final result generation phase, the content of d (a random number no greater than D) blocks in the contiguous XOR memory is updated with the hash function family, and d is recalculated to the last block of the working memory. Finally, the SHA3 function forms the final result.



```
/*
 * Step 3: Calculate the final result.
*/
void calculateFinalResult(uint8_t *Maddr, uint8_t *c, const uint32_t D, uint8_t *result) {
    uint32_t i = 0, j = 0, k = 0;
    memcpy(result, c, OUTPUT_LEN*sizeof(uint8_t));

    const uint32_t num = (WORK_MEMORY_SIZE >> 5) - 1;

    uint32_t it = 0;
    uint8_t result_rrs[OUTPUT_LEN];
    while(1) {
        uint8_t t = 0, shift_num = 0;
        uint32_t d = 0;
        reduce_bit(result, 32, (uint8_t *)&t, 8);
        t = (t & 0x0f) ^ (t >> 4);
```

[The Final Result of the Calculation]

The main features of the above algorithm include the followings:

1. The working memory cache capacity optimization is CPU-oriented, while it is difficult for GPU or ASIC to meet the memory capacity requirement for executing a large number of PoW algorithm at the same time;

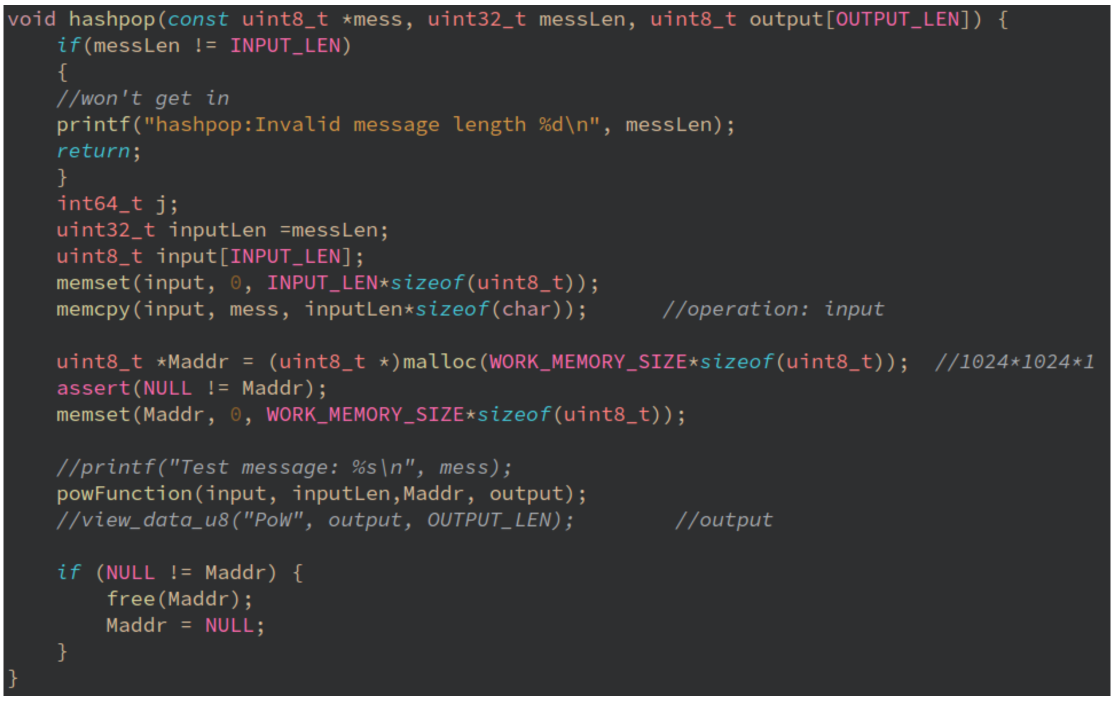
2. A family of 16 hash functions is randomly selected and executed, which will increase the chip area to prevent realization of ASIC, and cause GPU's multi-threaded execution of different paths, reducing its parallelization efficiency;

3. Strict serial execution sequence is used during the working memory modification stage, and they are accessed in units of bytes to control the execution parallelism of the ASIC or GPU and significantly reduce the memory system efficiency;

4. Complex control logic of the entire scheme, large required memory capacity, and irregular memory access address makes it difficult to implement the ASIC by using a conventional method of stacking a large number of acceleration component modules.

The actual test shows that the above performance of PoW on CPU is proportional to its number of threads, and the performance on GPU is basically the same as that of the general CPU server.

We also evaluated the feasibility of the ASIC. By modifying the algorithm, the requirements for the SRAM capacity of the ASIC chip for accelerating computing module is increased, resulting in a gradual reduction in the number of acceleration computing modules that can support the maximum capacity SRAM on the chip. Finally, ASIC accelerated chip exit mining due to cost performance. Meanwhile we have improved the utilization rate of SOC chips with high-speed CPU as the core to achieve higher ASIC R&D costs, longer design and production cycle and inhibit the production of ASIC mining rigs.



```c
void hashpop(const uint8_t *mess, uint32_t messLen, uint8_t output[OUTPUT_LEN]) {
    if(messLen != INPUT_LEN)
    {
    //won't get in
    printf("hashpop:Invalid message length %d\n", messLen);
    return;
    }
    int64_t j;
    uint32_t inputLen =messLen;
    uint8_t input[INPUT_LEN];
    memset(input, 0, INPUT_LEN*sizeof(uint8_t));
    memcpy(input, mess, inputLen*sizeof(char));      //operation: input

    uint8_t *Maddr = (uint8_t *)malloc(WORK_MEMORY_SIZE*sizeof(uint8_t));  //1024*1024*1
    assert(NULL != Maddr);
    memset(Maddr, 0, WORK_MEMORY_SIZE*sizeof(uint8_t));

    //printf("Test message: %s\n", mess);
    powFunction(input, inputLen,Maddr, output);
    //view_data_u8("PoW", output, OUTPUT_LEN);        //output

    if (NULL != Maddr) {
        free(Maddr);
        Maddr = NULL;
    }
}
```

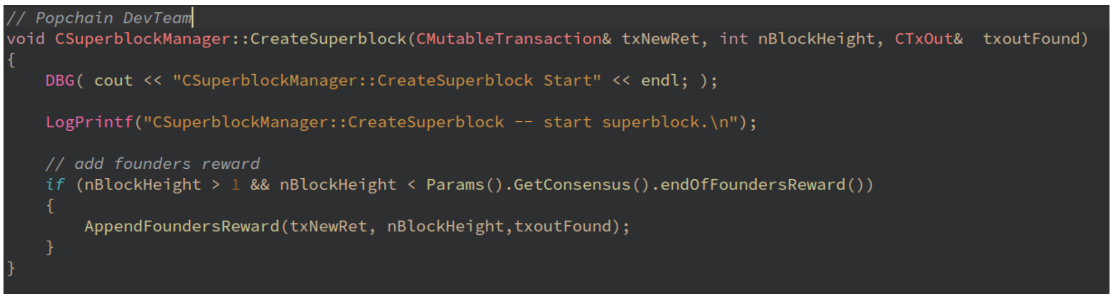
[Cryptopop IALG]

II. POPNODE

POPCHAIN refers to double-layer network approach of Dash and establishes a master node network on the based user's network, Popnode. POPCHAIN team or investors can create Popnode networks in the mainstream operating systems such as Linux/Windows/OS X, providing quick operations such as real-time transactions and anonymous delivery for ordinary users. As a private network for POPCHAIN, Popnode networks can also verify block data, support high-frequency transactions, improve the stability and security of the entire network, and provide stable PoSe services.

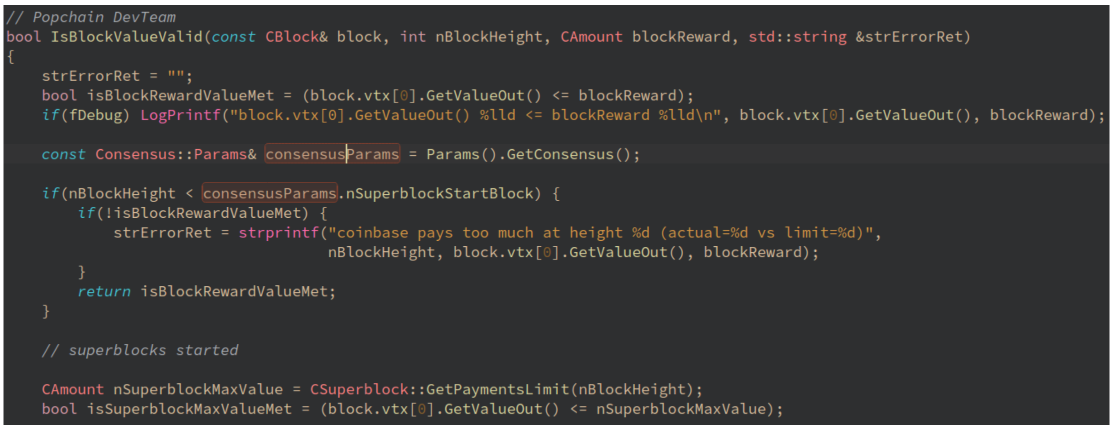
Of course, the Popnode of POPCHAIN and the Masternode of Dash are quite different in terms of their technology and function. First of all, the Masternode of Dash uses dividend mechanism. Anybody can upgrade to have a Masternode with 1000 Dash as a pledge. If the ordinary users want to have a Masternode, they need to pay a lot of money. Therefore, most of the Masternodes are owned by the development team or investors. The Masternode network becomes centralized and is going to be manipulated by a few people for their interests. On the other hand, POPCHAIN changed this approach by removing the benefits of the main node and allowing Popnode to provide free network and content storage services for POPCHAIN. The Popnodes will be provided by the POPCHAIN Foundation in the early stage and by the operation teams of DApps afterwards. The maintainer of the Popnode will take precedence over the benefits of the entire network.

The super blocks in POPCHAIN and Dash are also very different. The Masternode of Dash can initiate a proposal and a vote. The top proposal gets the requested fund. The funds are distributed once a month and packed in a superblock at a fixed block height. However, there is a defect that if the Dash development team and investors have more than 10% of the total Masternodes, they can make their proposals pass and get a lot of funds. Therefore, few multiple Masternode owners can obtain additional revenue besides the revenue from the ownership of the Masternodes. The general public does not have the right to give negative votes, but the funds distributed by the proposal are indeed accumulated from the 10% of each block for the superblock. This is obviously unfair to ordinary participants and the reason for POPCHAIN to remove the governance vote. The Popnodes cannot initiate the proposal. Funds for the community management, the only component of the super block, are provided by the Foundation, and are distributed to the Foundation once a month at a fixed time and not from the blocks dug out by the miners.



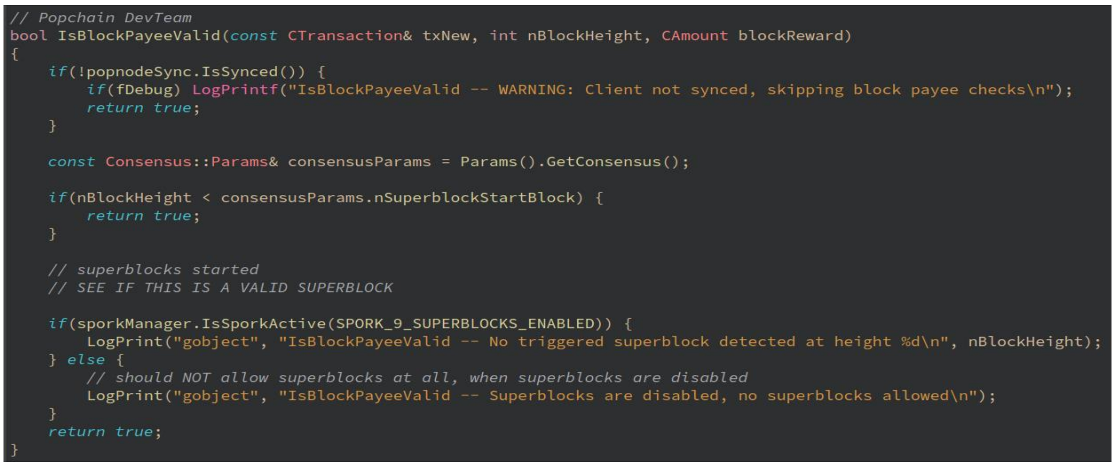
```
// Popchain DevTeam
void CSuperblockManager::CreateSuperblock(CMutableTransaction& txNewRet, int nBlockHeight, CTxOut&  txoutFound)
{
    DBG( cout << "CSuperblockManager::CreateSuperblock Start" << endl; );

    LogPrintf("CSuperblockManager::CreateSuperblock -- start superblock.\n");

    // add founders reward
    if (nBlockHeight > 1 && nBlockHeight < Params().GetConsensus().endOfFoundersReward())
    {
        AppendFoundersReward(txNewRet, nBlockHeight,txoutFound);
    }
}
```

[POPCHAIN creates a superblock]



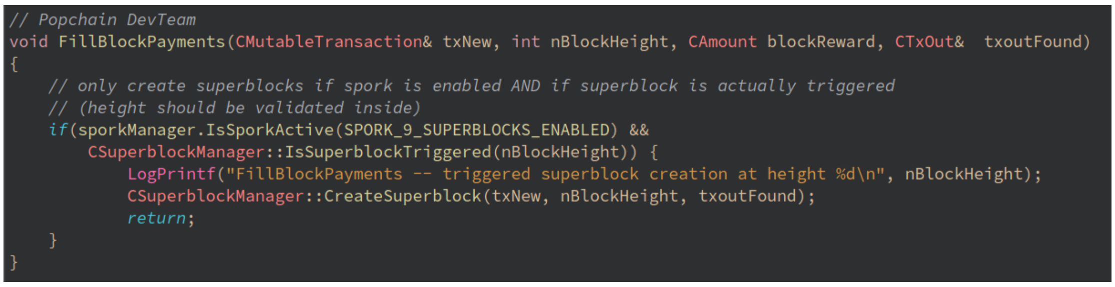
```
// Popchain DevTeam
bool IsBlockValueValid(const CBlock& block, int nBlockHeight, CAmount blockReward, std::string &strErrorRet)
{
    strErrorRet = "";
    bool isBlockRewardValueMet = (block.vtx[0].GetValueOut() <= blockReward);
    if(fDebug) LogPrintf("block.vtx[0].GetValueOut() %lld <= blockReward %lld\n", block.vtx[0].GetValueOut(), blockReward);

    const Consensus::Params& consensusParams = Params().GetConsensus();

    if(nBlockHeight < consensusParams.nSuperblockStartBlock) {
        if(!isBlockRewardValueMet) {
            strErrorRet = strprintf("coinbase pays too much at height %d (actual=%d vs limit=%d)",
                                    nBlockHeight, block.vtx[0].GetValueOut(), blockReward);
        }
        return isBlockRewardValueMet;
    }

    // superblocks started

    CAmount nSuperblockMaxValue = CSuperblock::GetPaymentsLimit(nBlockHeight);
    bool isSuperblockMaxValueMet = (block.vtx[0].GetValueOut() <= nSuperblockMaxValue);
```

[Popnode verifies a block]



```
// Popchain DevTeam
bool IsBlockPayeeValid(const CTransaction& txNew, int nBlockHeight, CAmount blockReward)
{
    if(!popnodeSync.IsSynced()) {
        if(fDebug) LogPrintf("IsBlockPayeeValid -- WARNING: Client not synced, skipping block payee checks\n");
        return true;
    }

    const Consensus::Params& consensusParams = Params().GetConsensus();

    if(nBlockHeight < consensusParams.nSuperblockStartBlock) {
        return true;
    }

    // superblocks started
    // SEE IF THIS IS A VALID SUPERBLOCK

    if(sporkManager.IsSporkActive(SPORK_9_SUPERBLOCKS_ENABLED)) {
        LogPrint("gobject", "IsBlockPayeeValid -- No triggered superblock detected at height %d\n", nBlockHeight);
    } else {
        // should NOT allow superblocks at all, when superblocks are disabled
        LogPrint("gobject", "IsBlockPayeeValid -- Superblocks are disabled, no superblocks allowed\n");
    }
    return true;
}
```

[POPCHAIN verifies the block revenue]



```
// Popchain DevTeam
void FillBlockPayments(CMutableTransaction& txNew, int nBlockHeight, CAmount blockReward, CTxOut&  txoutFound)
{
    // only create superblocks if spork is enabled AND if superblock is actually triggered
    // (height should be validated inside)
    if(sporkManager.IsSporkActive(SPORK_9_SUPERBLOCKS_ENABLED) &&
        CSuperblockManager::IsSuperblockTriggered(nBlockHeight)) {
            LogPrintf("FillBlockPayments -- triggered superblock creation at height %d\n", nBlockHeight);
            CSuperblockManager::CreateSuperblock(txNew, nBlockHeight, txoutFound);
            return;
    }
}
```

[POPCHAIN fills the block reward]

III. GHOST

1. Introducing the background of GHOST protocol

To support more than 4,000 transactions per second, we are introducing the GHOST protocol ("Greedy Heaviest Observed SubTree") to the original chain of POPCHAIN based on the Bitcoin framework to ensure security and increase the block generation rate.

Blockchain has a contradiction between the high speed of block generation rate and the security. Here are the reasons:

High-speed block generation rate often means a large number of "wasted blocks." As the time for a block to propagate in the network can be assumed to be fixed, if a block generation and propagation time ratios are not coordinated, a lot of nodes will mine for old blocks, resulting in a large ratio of wasted blocks.

It also causes centralization. For example, let's say a mining node A is a mining pool that accounts for 30% of the total network hashrate, and that mining node B occupies 10% of the total network hashrate. Node A will have 70% probability of generating a waste block, and node B will have a 90% probability of generating a waste block. If the interval between the new and the old block is too short, node A will be more efficient than node B because of the scale effect.

Therefore, the high-speed block generation rate will inevitably lead to a single mining pool leading the mining process of the whole network. GHOST protocol was first published by Yonatan Sompolinsky and Aviv Zohar in 2013 on PoW memory expansion. It initially proposes an effective way to shorten the block generation time. GHOST protocol is a main chain selection protocol.

GHOST protocol is based on the maximum number of subtrees, not the longest chain. What is the principle of the maximum number of subtrees? For example, the classic Proof-of-Work protocol takes the longest main chain as the basic principle for the next block selection; while GHOST protocol takes the maximum number of subtrees as the basic principle. Under the control of GHOST protocol, there is no necessary relationship between "block releasing too fast" and "51% attack continuously controlling the initiative".

After the introduction of the GHOST protocol, Ethereum successfully solved the problem of a large increase in the waste block rate and centralization and shortened the block generation time to 14 seconds. However, the Ethereum network carries both transactions and contracts, which caused the Ethereum network to be filled with junk transactions, and sometimes prevented from providing normal service. When the Ethereum network was congested, many transactions were rejected regardless of transaction fees consumed. In this case, the original chain of the

Bitcoin architecture, can quickly confirm the transaction fee. Therefore, we introduce the GHOST protocol to POPCHAIN to integrate the advantages of the Bitcoin framework and the GHOST protocol, so to improve the security and the block generation rate of the original chain.

2. How to introduce the GHOST protocol

The introduction of the GHOST protocol to POPCHAIN is a major change in the development of the original chain. Because it involves redefinition of the most basic block structure of the original chain, the impacted domain includes consensus, verification, etc. The work involved may contain these aspects:

1) Redefining the main chain's selecting algorithm

2) Centralizing risk simulation

3) Redefining the block award consensus

4) Redefining the block verification method

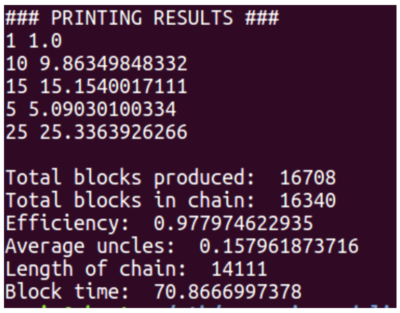
5) Redefining the orphan block treatment

6) Increasing the uncle block selecting algorithm

7) The uncle block's transaction process and impact analysis.

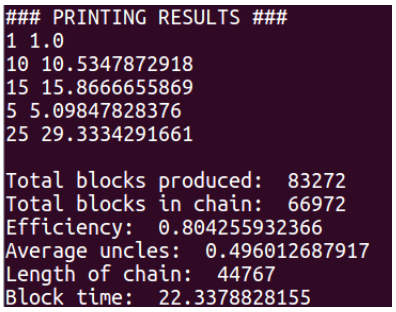
8) The study on the original chain's improvement impacting SPV

Currently, the initial efficiency of the introduction of GHOST has been tested, and the main chain efficiency is approximately estimated by changing the block generation time, propagation time, uncle block depth and bonus allocation:



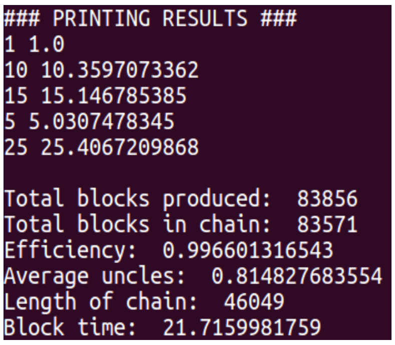
```
### PRINTING RESULTS ###
1 1.0
10 9.86349848332
15 15.1540017111
5 5.09030100334
25 25.3363926266

Total blocks produced:  16708
Total blocks in chain:  16340
Efficiency:  0.977974622935
Average uncles:  0.157961873716
Length of chain:  14111
Block time:  70.8666997378
```

[Single-level GHOST, 60s propagation time, 0.875 times uncle block reward, efficiency 97.79%]



```
### PRINTING RESULTS ###
1 1.0
10 10.5347872918
15 15.8666655869
5 5.09847828376
25 29.3334291661

Total blocks produced:  83272
Total blocks in chain:  66972
Efficiency:  0.804255932366
Average uncles:  0.496012687917
Length of chain:  44767
Block time:  22.3378828155
```

[Modified propagation time is 12s, efficiency drops to 80.42%, 25% hash rate gains 29% revenue]



```
### PRINTING RESULTS ###
1 1.0
10 10.3597073362
15 15.146785385
5 5.0307478345
25 25.4067209868

Total blocks produced:  83856
Total blocks in chain:  83571
Efficiency:  0.996601316543
Average uncles:  0.814827683554
Length of chain:  46049
Block time:  21.7159981759
```

[-4+3-uncle block level, 12s block generation time, 12s propagation time, efficiency 99.6%]

IV. Cross-chain Atomic Swap

Among many problems, the interoperability of blockchain greatly limits the blockchain application. Whether the chain is public or private, each blockchain project is an independent value network. Cross-chain technology is the key to realize the Internet value, and to connect the future exploration and development. Hash-locking technology in many cross-chain technologies can greatly enhance the blockchain transaction capability of Bitcoin type. Introducing the Cross-chain atomic swap protocol with the hash-locking technology to the POPCHAIN's public blockchain effectively supports the exchange of POPCHAIN and other digital currencies.

The Cross-chain atomic swap protocol algorithm is not complicated. Imagine a scenario where there is a special safe, the safe is transparent, and there are two locks on the safe: lock 1 and lock 2.

There is a time and a fingerprint lock on the lock 1. If the time is not up, the lock 1 can be unlocked with the fingerprint. When the time is up, the fingerprint on the lock 1 becomes invalid.

The lock 2 is a password lock. After entering a password, the password lock cannot be toggled. Therefore, others can see the password after the lock is opened. And others can completely clone the same lock as the lock 2 without knowing the password, including the password setting.

There are two businessmen A and B who do not believe each other. A has gold, B has silver. A want to exchange gold for silver, and B just wants to exchange silver for gold. They can reach this deal with the two safes.

B puts silver for A in the safe 1, sets the safe to be opened by A for 2 days, and by B after 2 days. B then sets the password of lock 2 of the safe 1. A does not know the password.

A check the safe 1 and find that he can open the lock 1 within 2 days and that the amount of silver in the safe is correct. He puts the gold for B in the safe 2 and sets the lock 1 of the safe 2 to be opened by B for only 1 day, and by A after 1 day. A then clones the lock 2 of the safe 1 and adds to safe 2.

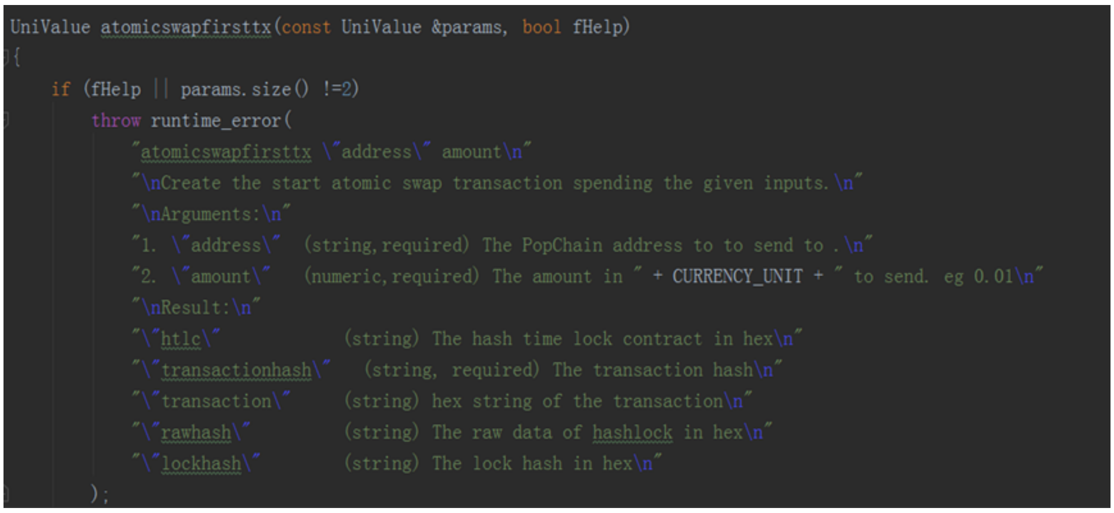
B can open the lock 1 of the safe 2 with his fingerprint and the lock 2 with the password, as B knows the password to the lock. B then can take the gold. The lock 2 of the safe 2 is opened and the password is left.

Now A knows the password of the lock 2 of the safe 1. A can open the lock 1 of the safe 1 with the fingerprint, open the lock 2 of the safe 1 with the password, and take the silver.

If B refuses to open the safe 2, A can't get silver, and B can't get the gold. After 1 day, A gets back his gold. B will get back his silver after 2 days.

In the POPCHAIN public block chain that introduces the cross-chain atomic swap protocol, the B user only needs to perform the following steps:

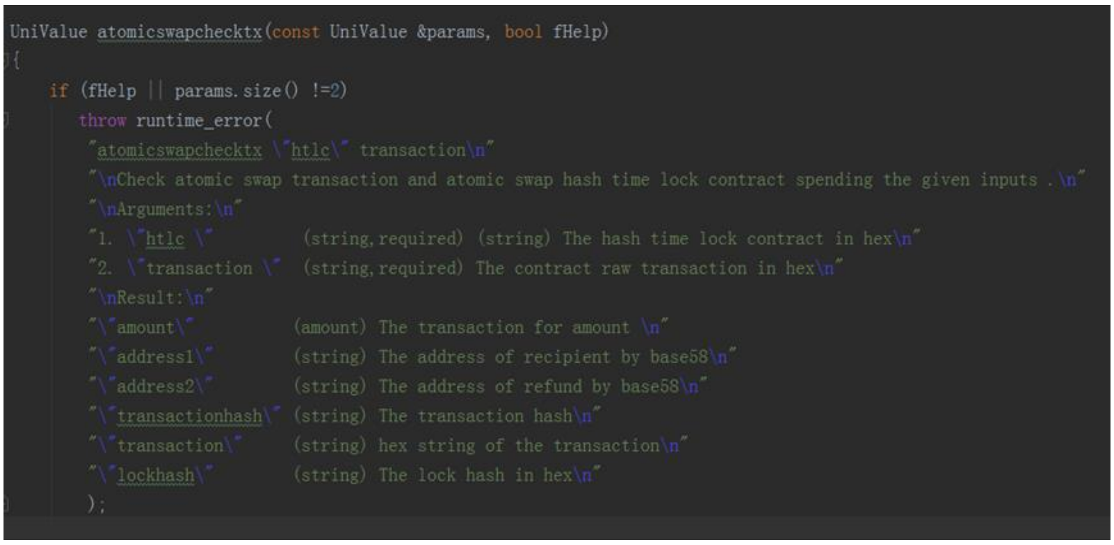
1. Initialize the transaction from B to A



```
UniValue atomicswapfirsttx(const UniValue &params, bool fHelp)
{
    if (fHelp || params.size() !=2)
        throw runtime_error(
            "atomicswapfirsttx \"address\" amount\n"
            "\nCreate the start atomic swap transaction spending the given inputs.\n"
            "\nArguments:\n"
            "1. \"address\"  (string,required) The PopChain address to to send to .\n"
            "2. \"amount\"   (numeric,required) The amount in " + CURRENCY_UNIT + " to send. eg 0.01\n"
            "\nResult:\n"
            "\"htlc\"               (string) The hash time lock contract in hex\n"
            "\"transactionhash\"    (string, required) The transaction hash\n"
            "\"transaction\"        (string) hex string of the transaction\n"
            "\"rawhash\"            (string) The raw data of hashlock in hex\n"
            "\"lockhash\"           (string) The lock hash in hex\n"
        );
```

[Atomic swap transaction initialization]

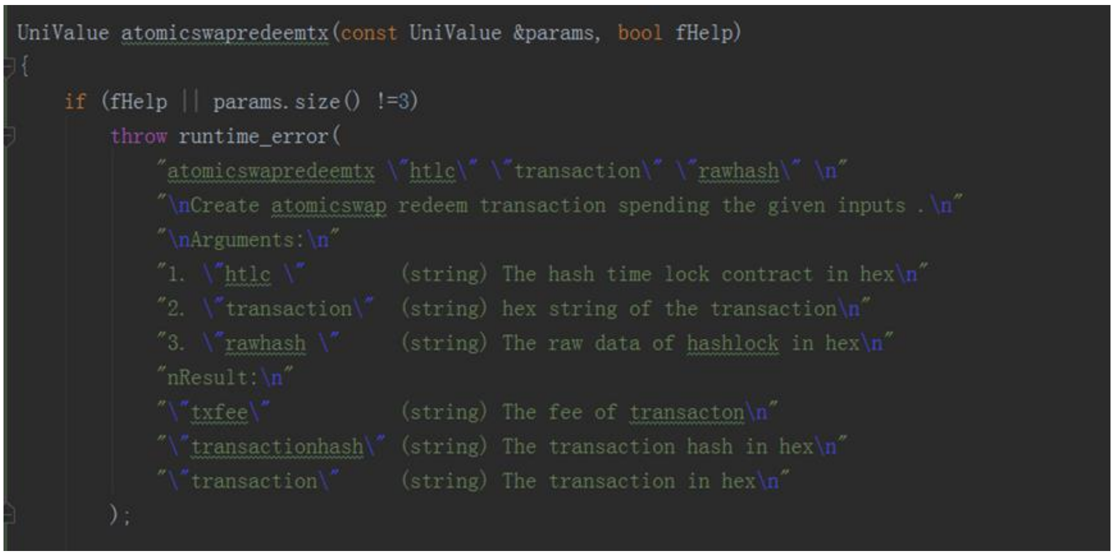
2. Check the transaction from A to B



```
UniValue atomicswapchecktx(const UniValue &params, bool fHelp)
{
    if (fHelp || params.size() !=2)
        throw runtime_error(
        "atomicswapchecktx \"htlc\" transaction\n"
        "\nCheck atomic swap transaction and atomic swap hash time lock contract spending the given inputs .\n"
        "\nArguments:\n"
        "1. \"htlc \"          (string,required) (string) The hash time lock contract in hex\n"
        "2. \"transaction \"   (string,required) The contract raw transaction in hex\n"
        "\nResult:\n"
        "\"amount\"          (amount) The transaction for amount \n"
        "\"address1\"        (string) The address of recipient by base58\n"
        "\"address2\"        (string) The address of refund by base58\n"
        "\"transactionhash\" (string) The transaction hash\n"
        "\"transaction\"     (string) hex string of the transaction\n"
        "\"lockhash\"        (string) The lock hash in hex\n"
        );
```

[Atomic swap transaction inspection]

3. Redeem atomic swap transaction



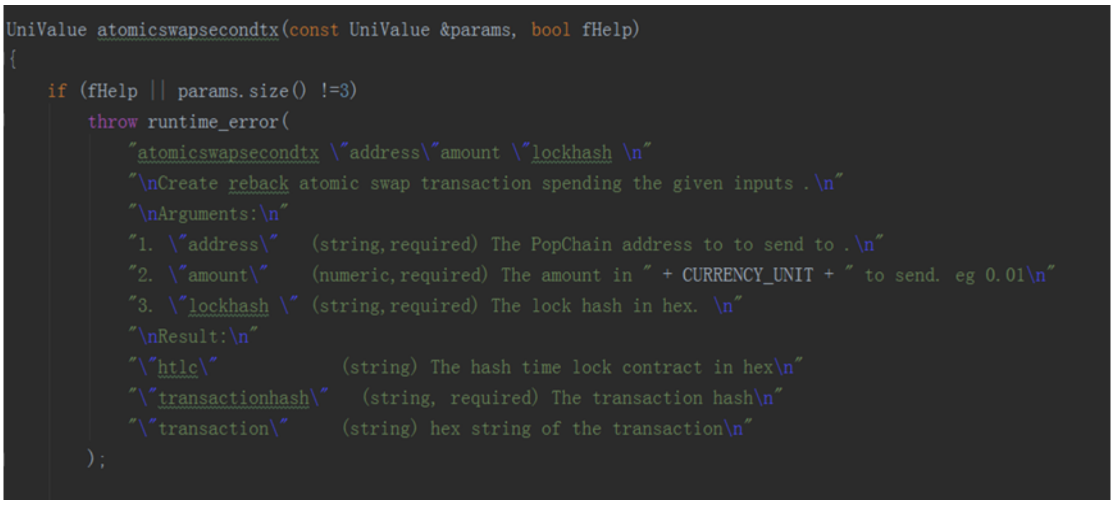
```
UniValue atomicswapredeemtx(const UniValue &params, bool fHelp)
{
    if (fHelp || params.size() !=3)
        throw runtime_error(
            "atomicswapredeemtx \"htlc\" \"transaction\" \"rawhash\" \n"
            "\nCreate atomicswap redeem transaction spending the given inputs .\n"
            "\nArguments:\n"
            "1. \"htlc \"          (string) The hash time lock contract in hex\n"
            "2. \"transaction\"    (string) hex string of the transaction\n"
            "3. \"rawhash \"       (string) The raw data of hashlock in hex\n"
            "nResult:\n"
            "\"txfee\"             (string) The fee of transacton\n"
            "\"transactionhash\"   (string) The transaction hash in hex\n"
            "\"transaction\"       (string) The transaction in hex\n"
        );
```

[Atomic swap transaction redemption]

In the POPCHAIN public blockchain that introduces the cross-chain atomic swap protocol, User A only needs to perform the following steps:

1) Check the transactions from B to A

2) Generate the transaction from A to B

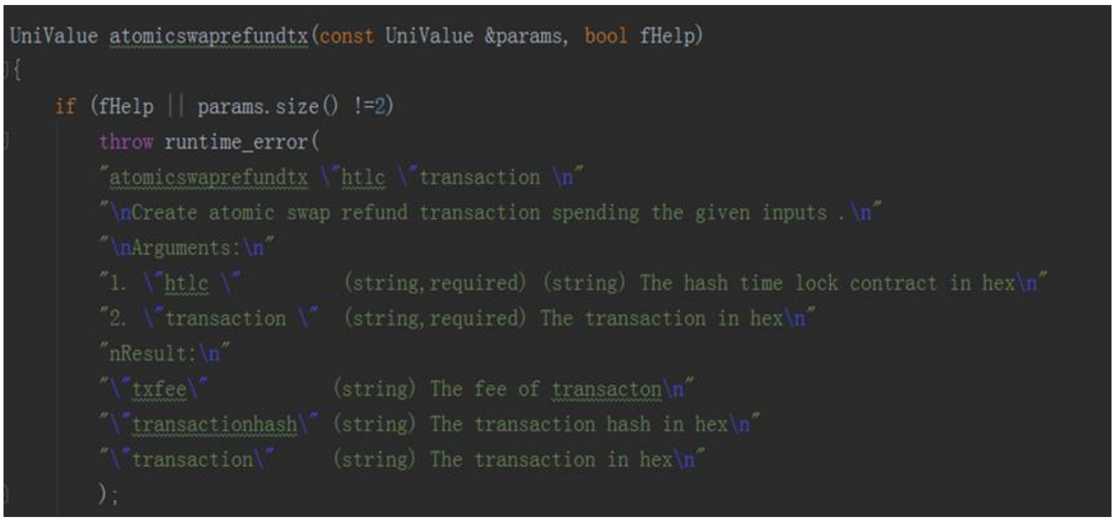


```
UniValue atomicswapsecondtx(const UniValue &params, bool fHelp)
{
    if (fHelp || params.size() !=3)
        throw runtime_error(
            "atomicswapsecondtx \"address\"amount \"lockhash \n"
            "\nCreate reback atomic swap transaction spending the given inputs .\n"
            "\nArguments:\n"
            "1. \"address\"   (string,required) The PopChain address to to send to .\n"
            "2. \"amount\"    (numeric,required) The amount in " + CURRENCY_UNIT + " to send. eg 0.01\n"
            "3. \"lockhash \" (string,required) The lock hash in hex. \n"
            "\nResult:\n"
            "\"htlc\"             (string) The hash time lock contract in hex\n"
            "\"transactionhash\"   (string, required) The transaction hash\n"
            "\"transaction\"      (string) hex string of the transaction\n"
        );
```

[Atomic swap transaction generation]

3) Redeem the transactions from B to A

   In the POPCHAIN public blockchain that introduces a cross-chain atomic swap protocol, if a party does not execute a transaction, he only needs to refund the transaction sent to the other party after the hash-lock time.



```
UniValue atomicswaprefundtx(const UniValue &params, bool fHelp)
{
    if (fHelp || params.size() !=2)
        throw runtime_error(
        "atomicswaprefundtx \"htlc \"transaction \n"
        "\nCreate atomic swap refund transaction spending the given inputs .\n"
        "\nArguments:\n"
        "1. \"htlc \"          (string,required) (string) The hash time lock contract in hex\n"
        "2. \"transaction \"   (string,required) The transaction in hex\n"
        "nResult:\n"
        "\"txfee\"            (string) The fee of transacton\n"
        "\"transactionhash\" (string) The transaction hash in hex\n"
        "\"transaction\"     (string) The transaction in hex\n"
        );
```

[Atomic swap transaction refund]

   The cross-chain atomic swap protocol makes both parties to be interdependent. We have created a programming logic in the implementation process, in which stops tractions if they are not passed. The transaction process does not require a third party. Users can keep and use their own funds to trade for other currencies (such as BTC, LTC, etc.). There is no need to deposit coins in exchanges, and the entire transaction process is recorded on blockchain. This solves the problems of safety vulnerabilities and digital asset custodies.