BanksCoin: A High Difficulty Crypto Commodity

Banks
bankscoin@gmail.com
https://etherscan.io/token/0xbc9d0f9969b4ba0bc7e4f6459b7393fe0e661620

Abstract. Satoshi Nakamoto created the Bitcoin protocol with the maximum target set to 2^224. At this level any CPU could solve for a valid challenge digest and receive new Bitcoin; Satoshi is said to have mined over 1,000,000 Bitcoin starting in early 2009. BanksCoin takes the fundamentals of Bitcoin with proof-of-work mining and operates on top of Ethereum for securing the network thus solving for the challenge digest is the only math done by the miner. BanksCoin has a total supply of 1,000,000 programmed to mint one every two days over thousands of years, there are no halvings. The maximum target is 2^204, relative to Bitcoin at launch, this means that the challenge digest is close to 1.05 million times harder than Bitcoin was in 2009 (2^20 times harder). Since 2009 technology has dramatically improved, specifically in FPGA chips made by Xilinx. These logic cells can be programmed and run the Solidity SHA-3 bitstream efficiently. BanksCoin is set to adjust difficulty every 10 blocks up or down based on the rolling average of time to solve for a block. The large increase in time (ten minutes with Bitcoin to two days with BanksCoin) allows the chips extended time to compute a valid digest, pinning the value of BanksCoin to the economic resource of electricity, computer hardware, and Ethereum gas.

1. Introduction

Mathematically sound money has proved to be one of the greatest inventions of the 21st century. Bitcoin revolutionized the way two individuals can transact value without a central entity. Bitcoin was inspired by Hashcash and BanksCoin is inspired by Bitcoin. BanksCoin seeks to take the fundamentals of Bitcoin and inject steroids to the proof-of-work process. Changing the maximum target to 2^204 makes the challenge digest a much harder computation. To compensate for this dramatic increase in the number of hashes needed per valid digest, the average block time is two days for chips to have time to compute a digest. With a FPGA hashing 24/7 (essentially running a microwave 24/7 in terms of electricity), the cost of production needed to solve for one block is well over one USD. This makes it a difficult task even for early adopters. Satoshi wrote about “one-CPU-one-vote” [1], BanksCoin is similar but one-FPGA-one vote. This will bypass the initial stages of a mineable token when CPU and GPU miners can solve for a digest. BanksCoin miners will need modern FPGA machines to have success. With one new BanksCoin scheduled to mint every two days, this creates a scarce, low float high difficulty crypto commodity.

If something is easy to get, how can it store any value? Silver and gold for example, are well-known valuable commodities; they are scarce, require work, and have acted as a store of value for thousands of years. BanksCoin is a mathematical store of value in the form of a cryptographic hash function. There is no pre-mine or ICO. The only way to get BanksCoin is through mining or buying. BanksCoin is an ERC-20 token and the process of mining unlike Bitcoin does not need to secure the network. The proof-of-work is only for computing the challenge digest therefore miners are not validating transactions. 0xBTC was the first coin to implement the EIP918 protocol as a mineable token. BanksCoin is similar to the code of 0xBTC with changes to parameters: maximum target, target eth blocks per diff period, blocks per adjustment, minimum target, reward size, total supply and there are no halvings scheduled. The contract will continuously mint one BanksCoin every valid challenge digest. These changes make the functionality of the smart contract fundamentally different, but anyone curious about the internals should check out the 0xBTC whitepaper [2].

2. The Evolution of Crypto Mining

Crypto mining started with CPU power as Satoshi wrote in his whitepaper [1]. After CPUs were GPUs, ASICs were introduced with pooled mining. In 2020, ASICs are used for mining Bitcoin in combination with mining pools while FPGAs are used to mine bitstreams that do not have ASIC implementation. BanksCoin uses the 0xToken bitstream and currently, an ASIC implementation doesn't exist but FPGAs are available. These machines are incredible at hashing while running at a relatively low energy cost.

> "Keccak, the winner of the SHA-3 competition, is blazing fast when implemented on dedicated (ASIC) or programmable (FPGA) hardware. Its throughput for a given circuit area is an order of magnitude higher than SHA-2 or any of the SHA-3 finalists. And if you care beyond plain speed, note that it also consumes much less energy per bit. In this sense, Keccak is a green cryptographic primitive" [3].

0xToken bitstream uses the Keccak sponge construction and this makes for efficient hashing for a challenge digest. The goal for BanksCoin is to bypass the CPU stage and GPU stage, then move into the pooled mining stage with FPGA hardware at launch. Starting BanksCoin at these target levels is like starting Bitcoin after over six and a half million have been mined in June 2011 algorithmically [4]. This results in a higher computational cost per challenge digest making the cost of production per BanksCoin above one USD. Historically small mineable crypto currencies have high volatility and are prone to dumping. Currently, no other mineable token has no halvings, one per block

and one block programmed to take two days. This should act as a line of defense and it should create less volatility in the future market.

3. Distribution Schedule

BanksCoin has a supply distribution that is hard coded to aim for one BanksCoin every two days. For example, if 10 BanksCoin are minted in less than 20 days, the difficulty adjusts up to make the challenge digest harder to solve. Likewise if it takes more than 20 days to solve for 10 BanksCoin the contract will get easier. This design is intentional, the purpose is to limit the amount distributed and retain a low circulating supply. Each year, close to 183 BanksCoin are designed to be minted, but if the FPGA adoption increases this number could be many more. It all depends on the amount of hashrate on the contract for when the adjustments occur. This distribution schedule makes BanksCoin unique from any crypto commodity made before.

Many coins are generated on the Ethereum blockchain with a few clicks of a mouse. Yes, by clicking the mouse BanksCoin is created but the coins need to be mined. Mineable crypto, in my opinion, is true crypto and truly decentralized. BanksCoin is unique in regards to the internal functionality of the contract, this distribution schedule is designed to protect long term holders from large hash rate miners dumping their holdings onto exchanges. In this case, the increase in hash rate will only further the increase in difficulty, validating the store of value narrative around computational costs in the free market. Bitcoin would not be worth $9,000 if it didn't have enormous built in computational costs in the free market.

4. The Effect of One BanksCoin per Two Days

In a free market, the supply and demand determine price. If a commodity has a high supply, the price will be low because the sellers will sell into the market buy orders, plummeting the price. The vision for BanksCoin is to have such a low supply being generated, making it difficult to destroy the order book because the proof-of-work will take time. For example, if a buyer wants to spend one USD per one BanksCoin but there is no sell order listed, then the buyer must wait until either more is mined and the miner sells or a holder wants to sell. If someone else comes along with the intention of buying one BanksCoin at two USD during this time, then they will outbid the other buy order and wait until it’s filled. The effect of one BanksCoin per two days makes a commodity with a low float. Because the amount of units produced are small, theoretically, less sell pressure helps maintain liquidity in the market. The market can grow where buyers and sellers can transact at a healthy rate of production, pinning the price of BanksCoin to the cost of electricity, computer hardware, and Ethereum gas. In the free market of decentralized mining, if the orderbook has enough value for miners to make money with their rigs they will colloquially “*cash grab*”. People all around the world are looking for alpha (ways

to generate returns). The free market will take over BanksCoin if the orderbook gets enough value, leading to a cycle of more miners, increased difficulty adjustments, and increased production costs. Just like Bitcoin, there should be expansionary times and contractionary times based on the market's demand.

Besides liquidity in the market, another added benefit to this distribution is on the price. Theoretically, if cash flow coming from the buy side outpaces the sell side, then the price is bound to increase. With difficulty adjustments, the computational costs of one block plus the Ethereum gas increases. This is why there is strong confidence that BanksCoin will be superior to the USD throughout its existence.

5. Purpose

Many cryptocurrencies/crypto commodities promise the world but never deliver. BanksCoin is not intended to be the world's reserve currency, it looks to run alongside fiat systems as a long term algorithmic store of value. Each BanksCoin is private property that can be owned on the Ethereum blockchain by any individual. Only time will tell the evolution of the difficulty adjustments, but BanksCoin looks to change the way people view proof-of-work and the value added component it brings. BanksCoin difficulty of one million is around 1.05 trillion in Bitcoin difficulty. The long term vision is to provide these machines time to solve for a valid challenge digest that will only get harder and harder assuming miner adoption over time.

Bitcoin puzzles most economists on how can something that has no intrinsic value to start with be worth anything? This assumption defies all economic theory. Bitcoin was able to bootstrap itself to many machines doing the work. A protocol on one computer is now a massive network computing a difficult challenge digest. As Hayes states in *Cryptocurrency Value Formation: An Empirical Analysis Leading to a Cost of Production Model for Valuing Bitcoin*, “The more aggregate computational power employed in mining for a cryptocurrency, the higher the value” [5]. There is a direct correlation between production costs of mineable cryptocurrency and the minimum level of a valid digest. Essentially, BanksCoin is starting two and a half years ahead of Bitcoin in the algorithmic process to begin bootstrapping machines to the network [4].

6. Conclusion

Ethereum miners secure the network, while BanksCoin miners solve for a valid challenge digest. BanksCoin miners are purely running to solve the cryptographic puzzle allowing for a pure mined crypto commodity like 0xBTC. With changes to parameters of the 0xBTC contract, BanksCoin behaves fundamentally different while using the same proven built in functions. It also starts years ahead of Bitcoin algorithmically with a lower reward per block. BanksCoin is a new, open-source, and decentralized high difficulty crypto commodity backed by mathematics.

Code

```
pragma solidity ^0.4.18;
library SafeMath {
    function add(uint a, uint b) internal pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    function sub(uint a, uint b) internal pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
    function mul(uint a, uint b) internal pure returns (uint c) {
        c = a * b;
        require(a == 0 || c / a == b);
    }
    function div(uint a, uint b) internal pure returns (uint c) {
        require(b > 0);
        c = a / b;
    }
}
library ExtendedMath {
    function limitLessThan(uint a, uint b) internal pure returns (uint c) {
        if(a > b) return b;
        return a;
    }
}
contract ERC20Interface {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant returns (uint balance);
    function allowance(address tokenOwner, address spender) public constant returns (uint remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public returns (bool success);
    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}
contract ApproveAndCallFallBack {
    function receiveApproval(address from, uint256 tokens, address token, bytes data) public;
}
contract Owned {
    address public owner;
    address public newOwner;
    event OwnershipTransferred(address indexed _from, address indexed _to);
    function Owned() public {
        owner = msg.sender;
    }
    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }
    function transferOwnership(address _newOwner) public onlyOwner {
        newOwner = _newOwner;
    }
    function acceptOwnership() public {
        require(msg.sender == newOwner);
        OwnershipTransferred(owner, newOwner);
```

```
        owner = newOwner;
        newOwner = address(0);
    }
}
contract BanksCoin is ERC20Interface, Owned {
    using SafeMath for uint;
    using ExtendedMath for uint;
    string public symbol;
    string public  name;
    uint8 public decimals;
    uint public _totalSupply;
    uint public latestDifficultyPeriodStarted;
    uint public epochCount;
    uint public _BLOCKS_PER_READJUSTMENT = 10;// 10 blocks per adjustment, scheduled for every 20 days
    uint public  _MINIMUM_TARGET = 2**10;
    uint public  _MAXIMUM_TARGET = 2**204;// BTC started at 2^224, BKC starts at 2^204= 2^20 relative to BTC: 1.05m
    uint public miningTarget;
    bytes32 public challengeNumber;
    uint public rewardEra;
    uint public maxSupplyForEra;
    address public lastRewardTo;
    uint public lastRewardAmount;
    uint public lastRewardEthBlockNumber;
    bool locked = false;
    mapping(bytes32 => bytes32) solutionForChallenge;
    uint public tokensMinted;
    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allowed;
    event Mint(address indexed from, uint reward_amount, uint epochCount, bytes32 newChallengeNumber);
    function BanksCoin() public onlyOwner{
        symbol = "BKC";
        name = "BanksCoin";
        decimals = 8;
        _totalSupply = 1000000 * 10**uint(decimals);// Total supply is 1,000,000
        if(locked) revert();
        locked = true;
        tokensMinted = 0;
        rewardEra = 0;
        maxSupplyForEra = _totalSupply; // 1 million to mint in 1 era
        miningTarget = _MAXIMUM_TARGET;
        latestDifficultyPeriodStarted = block.number;
        _startNewMiningEpoch();
    }
        function mint(uint256 nonce, bytes32 challenge_digest) public returns (bool success) {
            bytes32 digest =  keccak256(challengeNumber, msg.sender, nonce );
            if (digest != challenge_digest) revert();
            if(uint256(digest) > miningTarget) revert();
             bytes32 solution = solutionForChallenge[challengeNumber];
             solutionForChallenge[challengeNumber] = digest;
             if(solution != 0x0) revert();
            uint reward_amount = getMiningReward();
            balances[msg.sender] = balances[msg.sender].add(reward_amount);
            tokensMinted = tokensMinted.add(reward_amount);
            assert(tokensMinted <= maxSupplyForEra);
            lastRewardTo = msg.sender;
            lastRewardAmount = reward_amount;
            lastRewardEthBlockNumber = block.number;
```

```
            _startNewMiningEpoch();
             Mint(msg.sender, reward_amount, epochCount, challengeNumber );
           return true;
         }
    function _startNewMiningEpoch() internal {
      if( tokensMinted.add(getMiningReward()) > maxSupplyForEra && rewardEra < 1) // This will be evoked in thousands of years
      {
       rewardEra = rewardEra + 1;
      }
      maxSupplyForEra = _totalSupply;
      epochCount = epochCount.add(1);
      if(epochCount % _BLOCKS_PER_READJUSTMENT == 0)
      {
       _reAdjustDifficulty();
      }
      challengeNumber = block.blockhash(block.number - 1);
    }
    function _reAdjustDifficulty() internal {
        uint ethBlocksSinceLastDifficultyPeriod = block.number - latestDifficultyPeriodStarted;
        uint epochsMined = _BLOCKS_PER_READJUSTMENT;// 10 every 20 days
        uint targetEthBlocksPerDiffPeriod = epochsMined * 13190; // 13.1 seconds per block, 4.58 per minute, 274.81 per hour, 13190.83 per two
days
        if( ethBlocksSinceLastDifficultyPeriod < targetEthBlocksPerDiffPeriod )
        {
          uint excess_block_pct = (targetEthBlocksPerDiffPeriod.mul(100)).div( ethBlocksSinceLastDifficultyPeriod );
          uint excess_block_pct_extra = excess_block_pct.sub(100).limitLessThan(1000);
          miningTarget = miningTarget.sub(miningTarget.div(2000).mul(excess_block_pct_extra));
        }else{
          uint shortage_block_pct = (ethBlocksSinceLastDifficultyPeriod.mul(100)).div( targetEthBlocksPerDiffPeriod );
          uint shortage_block_pct_extra = shortage_block_pct.sub(100).limitLessThan(1000);
          miningTarget = miningTarget.add(miningTarget.div(2000).mul(shortage_block_pct_extra));
        }
        latestDifficultyPeriodStarted = block.number;
        if(miningTarget < _MINIMUM_TARGET)
        {
          miningTarget = _MINIMUM_TARGET;
        }
        if(miningTarget > _MAXIMUM_TARGET)
        {
          miningTarget = _MAXIMUM_TARGET;
        }
    }
    function getChallengeNumber() public constant returns (bytes32) {
        return challengeNumber;
    }
     function getMiningDifficulty() public constant returns (uint) {
        return _MAXIMUM_TARGET.div(miningTarget);
    }
    function getMiningTarget() public constant returns (uint) {
      return miningTarget;
   }
    function getMiningReward() public constant returns (uint) {
         return (1 * 10**uint(decimals) ).div( 2**rewardEra ) ; // One BKC per reward, 2^0=1
    }
    function getMintDigest(uint256 nonce, bytes32 challenge_number) public view returns (bytes32 digesttest) {
        bytes32 digest = keccak256(challenge_number,msg.sender,nonce);
        return digest;
```

```
    }
    function checkMintSolution(uint256 nonce, bytes32 challenge_digest, bytes32 challenge_number, uint testTarget) public view returns (bool success) {
        bytes32 digest = keccak256(challenge_number,msg.sender,nonce);
        if(uint256(digest) > testTarget) revert();
        return (digest == challenge_digest);
      }
  function totalSupply() public constant returns (uint) {
    return _totalSupply  - balances[address(0)];
  }
  function balanceOf(address tokenOwner) public constant returns (uint balance) {
    return balances[tokenOwner];
  }
  function transfer(address to, uint tokens) public returns (bool success) {
    balances[msg.sender] = balances[msg.sender].sub(tokens);
    balances[to] = balances[to].add(tokens);
    Transfer(msg.sender, to, tokens);
    return true;
  }
  function approve(address spender, uint tokens) public returns (bool success) {
    allowed[msg.sender][spender] = tokens;
    Approval(msg.sender, spender, tokens);
    return true;
  }
  function transferFrom(address from, address to, uint tokens) public returns (bool success) {
    balances[from] = balances[from].sub(tokens);
    allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
    balances[to] = balances[to].add(tokens);
    Transfer(from, to, tokens);
    return true;
  }
  function allowance(address tokenOwner, address spender) public constant returns (uint remaining) {
    return allowed[tokenOwner][spender];
  }
  function approveAndCall(address spender, uint tokens, bytes data) public returns (bool success) {
    allowed[msg.sender][spender] = tokens;
    Approval(msg.sender, spender, tokens);
    ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, this, data);
    return true;
  }
  function () public payable {
    revert();
  }
  function transferAnyERC20Token(address tokenAddress, uint tokens) public onlyOwner returns (bool success) {
    return ERC20Interface(tokenAddress).transfer(owner, tokens);
  }
}
```

References

[1] Nakamoto, S. “Bitcoin: A Peer-to-Peer Electronic Cash System.” *Bitcoin.org,* 31 October 2008, https://bitcoin.org/bitcoin.pdf

[2] Infernal_toast, and Jay Logelin. “0xbitcoin/White-Paper.” *GitHub*, 18 July 2018, github.com/0xbitcoin/white-paper.

[3] Peeters, Michaël, et al. “Team Keccak.” *Keccak Team*, 12 June 2017, keccak.team/2017/is_sha3_slow.html.

[4] “Bitcoin Difficulty Chart.” *BitInfoCharts*, bitinfocharts.com/comparison/bitcoin-difficulty.html.

[5] Hayes, Adam. “Cryptocurrency Value Formation: An Empirical Analysis Leading to a Cost of Production Model for Valuing Bitcoin.” *Semanticscholar.org*, 2015, pdfs.semanticscholar.org/2ba6/3113a5675ebb6bb58d5f261c6117f95cb379.pdf.