



Bitcoin Classic Token - BCT

Whitepaper 1.2

Bitcoin Classic Token: The Decentralized Bitcoin Token on the original Ethereum Chain - Ethereum Classic

Abstract

It should be clarified that Bitcoin is distributed via 'bitcoin mining' and therefore aligns itself as a 'commodity' and not a 'security.' BCT is the first ERC223 mineable token that aligns itself as a 'commodity' since it is distributed only using 'Proof of Work Mining' identical to the Bitcoin model. This token is also transferred on a blockchain in a method very similar to Bitcoin and so therefore interfaces with other software and with the world in a manner which is effectively identical to Bitcoin. This token has several advances that set it apart from Bitcoin such as the ability to directly interact with Ethereum Classic Smart Contracts.

Background

BCT is the implementation of Bitcoin in Solidity and is the first truly decentralized ERC223 token for Ethereum Classic and will be launched soon after launch on a decentralized exchange. It is an open source community project, not led by an official team or corporation, and therefore does not have ICO capital or other vast amounts of currency/capital that a centralized token project would have. We believe as a community that decentralization is the true flavour of the blockchain and that is the architecture that provides open and transparent trust for users. We also believe that Ethereum Classic and ERC223 tokens are a significant segment of the future of blockchain technology.

BCT is designed to be used as a decentralized 'bitcoin-like' token within the Ethereum Classic ecosystem and beyond. It avoids problems related to centralization as seen on Ethereum and security because it is powered by the Ethereum Classic Network and by globally distributed anonymous miners. Since it follows a standard protocol (ERC223), it is stored in a traditional Ethereum Classic wallet such as Classic Ether Wallet or the Saturn Wallet. Since every BCT token has been mined in a completely decentralized manner, there is no central body or central organization which controls or enforces any aspect of BCT. The community owns and operates the token in a flat structure and every individual has the same power over the smart contract as any other individual. This is on purpose in order to follow the same model of Bitcoin and to establish BCT as a commodity.

Contract Location

The BCT contract is located at Ethereum Classic address

0x1be6d61b1103d91f7f86d47e6ca0429259a15ff0

The Decentralized Token

Since BCT is mined like Bitcoin, it acts just like a commodity. The difficulty of 'mining' this commodity automatically adjusts to the total computational power used to mine it..

This powerful mechanism frees individuals from having to use a third party exchange, susceptible to security holes and wallet compromise, and third party escrows. The movement away from centralization is a core tenant of what Satoshi Nakamoto originally intended with classic Bitcoin (Nakamoto, 2009). BCT can be traded permissionlessly within immutable permanent smart contracts which are not able to be censored or restricted by central entities. Considering that BCT will be launching on a decentralized exchange after launch. This adds another clear advantage and is closer to fulfilling Satoshi's complete vision.

Account System

As an ERC223 token, BCT uses a traditional Ethereum Classic account. These accounts are free and are impossible to hack or to steal from, given that the private key has not been exposed. BCT can be stored in a Ledger Nano, Trezor or any other wallet that supports ERC223 tokens.

Mining

There is currently no mineable token on the Ethereum Classic Blockchain. BCT is mined using a simple Keccak256 (Sha3) algorithm using the following methodology:

```
keccak256(nonce, minerEthAddress, challengeNumber) < difficultyTarget
```

The nonce is a random number selected by the mining software. The mining software mines to try to find a valid nonce. If the above statement evaluates to true, then the nonce is a valid solution to the proof of work. The challengeNumber is just a recent Ethereum Classic block hash. Every round, the challengeNumber updates to the most recent Ethereum Classic block hash so future works cannot be mined in the past. The miner's Ethereum Classic Address is part of the hashed solution so that when a nonce solution is found, it is only valid for that particular miner and man in the middle attacks cannot occur. This also enables pool mining. The difficulty target becomes smaller and smaller automatically as more hashpower is added to the network.

Pool Mining

When mining BCT, whenever a miner submits a solution, the miner must pay a small gas fee in order to execute the Ethereum smart contract code for the mint() function.

If the gas fee is too low, the solution will take too long to be mined and if difficulty is not at equilibrium then another mint() solution from another miner will likely be mined first. This renders the original miners solution invalid and the transaction will revert(). To alleviate gas fees for miners, they can instead mine into a pool. This way, the pool will then submit the solutions to the smart contract and pay a gas fee. Then the pool will typically take a small percent of the rewards and give the rest to the miner for providing the PoW solution.

Since the miner's Ethereum classic address is included in the proof of work, pools require that miners mine using the pool's Ethereum Classic address. This way, the miner cannot submit full solutions to the contract while only giving partial solutions to the pool. If the miner is mining on behalf of the pool (using the pools address in the PoW algorithm) then it will not be able to submit any of those solutions to the smart contract without a revert(). This allows pools to operate without being cheated by the miners.

Typically, a pool will accept 'partial solutions' from miners which means the miners will receive 'shares' from the pool for solutions that are close to valid but not quite valid. This follows the same methodology as Bitcoin and Ethereum Classic Proof of Work pool mining. Probability theory states that, given enough close solutions, a full solution will eventually be found.

Smart Contract

Typically, ERC223 tokens will grant all tokens to the owner or will have an ICO which demands that amounts of ETC be sent to the owner for an initial offering of tokens. Instead of granting tokens to the 'contract owner', all BCT tokens are locked within the smart contract initially. These tokens are dispensed, 50 at a time, by calling the function 'mint' and using Proof of Work, similar to mining bitcoin classic.

Token

ERC-223 Interface

name

Returns the name of the token - e.g. "Bitcoin Classic Token".

OPTIONAL - This method can be used to improve usability, but interfaces and other contracts MUST NOT expect these values to be present.

```
function name() constant returns (string name)
```

symbol

Returns the symbol of the token. e.g. "BCT".

OPTIONAL - This method can be used to improve usability, but interfaces and other contracts MUST NOT expect these values to be present.

```
function symbol() constant returns (string symbol)
```

totalSupply

Returns the total token supply.

```
function totalSupply() constant returns (uint256 totalSupply)
```

balanceOf

Returns the account balance of another account with address `_owner`.

```
function balanceOf(address _owner) constant returns (uint256 balance)
```

Mining Operations

mint

Returns a flag indicating a successful hash digest verification. In order to prevent MiTM attacks, it is recommended that the digest include a recent Ethereum Classic block hash and `msg.sender`'s address. Once verified, the mint function calculates and delivers a mining reward to the sender and performs internal accounting operations on the contract's supply.

```
function mint(uint256 nonce, bytes32 challenge_digest) public returns (bool success)
```

Mint Event

Upon successful verification and reward the mint method dispatches a Mint Event indicating the reward address, the reward amount, the epoch count and newest challenge number.

```
event Mint(address indexed from, uint reward_amount, uint epochCount, bytes32 newChallengeNumber);
```

getChallengeNumber

Recent ethereum block hash, used to prevent pre-mining future blocks.

```
function getChallengeNumber() public constant returns (bytes32)
```

getMiningDifficulty

The number of digits that the digest of the PoW solution requires which typically auto adjusts during reward generation. Return the current reward amount. Depending on the algorithm, typically rewards are divided every reward era as tokens are mined to provide scarcity.

```
function getMiningDifficulty() public constant returns (uint)
```

getMiningReward

Return the current reward amount. Depending on the algorithm, typically rewards are divided every reward era as tokens are mined to provide scarcity.

```
function getMiningReward() public constant returns (uint)
```

Mining Debug Operations

getMintDigest

Returns a test digest using the same hashing scheme used when minting new tokens.

```
function getMintDigest(uint256 nonce, bytes32 challenge_digest, bytes32 challenge_number) public view returns (bytes32 digesttest)
```

OPTIONAL - This method can be used to improve usability, but interfaces and other contracts MUST NOT expect these values to be present.

checkMintSolution

Verifies a sample solution using the same scheme as the mint method.

```
function checkMintSolution(uint256 nonce, bytes32 challenge_digest, bytes32 challenge_number, uint testTarget) public view returns (bool success)
```

OPTIONAL - This method can be used to improve usability, but interfaces and other contracts MUST NOT expect these values to be present.

Minting New BCT Tokens

The Bitcoin Classic Token was deployed to the Ethereum Classic blockchain on xxxxxxxx, with the following attributes:

- No pre-mine
- No ICO
- 21,000,000 tokens total supply
- Difficulty target auto-adjusts with PoW hashrate
- Rewards decrease as more tokens are minted
- ERC223 compatibility

As such, the only way for a user to acquire BCT is to mine them or purchase them from miners on decentralized exchanges. The mint function is responsible for verifying the validity of the hash solution, updating the contracts internal state and issuing new BCT.

```

function mint(uint256 nonce, bytes32 challenge_digest) public returns (bool success) {
    uint reward_amount = getMiningReward();

    bytes32 digest = keccak256(challengeNumber, msg.sender, nonce );

    if (digest != challenge_digest) revert();

    //the digest must be smaller than the target
    if(uint256(digest) > miningTarget) revert();

    uint hashFound = rewardHashesFound[digest];
    rewardHashesFound[digest] = epochCount;
    if(hashFound != 0) revert(); //prevent the same answer from awarding twice

    balances[msg.sender] = balances[msg.sender].add(reward_amount);

    tokensMinted = tokensMinted.add(reward_amount);

    //set readonly diagnostics data
    lastRewardTo = msg.sender;
    lastRewardAmount = reward_amount;
    lastRewardEthBlockNumber = block.number;

    //start a new round of mining with a new 'challengeNumber'
    _startNewMiningEpoch();

    Mint(msg.sender, reward_amount, epochCount, challengeNumber );

    return true;
}

```

figure 1. Bitcoin Classic Token Smart Contract mint() function

The mining reward is initially gathered and follows the same algorithm as Bitcoin classic. Essentially following the paradigm of a fully decentralized monetary system, whereby the tokens are created by the nodes of a peer to peer network. The BCT algorithm defines how the token will be created and at what rate.

As with Bitcoin, BCT's are generated every time a user discovers a new block by being the first to submit Proof of Work for each round. The rate of the block creation is adjusted every 1024 to aim for a relatively constant adjustment period equal to approximately 6 BCT blocks per hour. The number of BCT generated per block is set to decrease logarithmically, having a 50% reduction every time half of the remaining supply has been mined. This ensures that the number of BCT in existence will never exceed 21 million.

A unique 'nonce' has to be passed into the mint function along with the hash solution digest in order for tokens to be dispensed. To find this special number, it is necessary to run a mining program. More specifically, the PoW includes a recent Ethereum Classic block hash combined with the wallet sender's address in order to prevent man in the middle attacks when minting new coins. The challenge and nonce are validated in solidity using the keccak256 hashing algorithm to decipher the challenge's digest. Once the digest has been extracted, it is validated to match the

expected challenge result and then check to ensure that it is smaller than the mining target difficulty.

The mining reward is calculated based on the logarithmic halving algorithm making the BCT token a reliably deflationary asset. The award is immediately assigned to the sender's wallet address and the 'tokens minted count' is incremented within the smart contract for any other software to monitor. Notably, the contract then validates that the tokens minted count is less than or equal to the maximum supply or the given halving era that transaction is taking place. Next, the contract records diagnostics reflecting reward address, amount and ether block number for the purpose of public transparency and for other software to monitor.

Difficulty Calculation and Adjustment

After every block is minted, the smart contract will determine if it is time to adjust the difficulty. This occurs every 2016 mined blocks. Just before this occurs, the contract increments the reward era if necessary - this is, if the tokens minted count has exceeded the maximum era supply which is calculated via a simple halving algorithm:

$$\text{max_era_supply} = \text{total_supply} - (\text{total_supply} / (2 * (\text{reward_era} + 1)))$$

This means that the first era supply is 10500000 tokens, the second era supply is 15750000 tokens, the third era supply is 18375000 tokens and so forth. During the first era, the block reward for a mint() is 50 tokens. During the second era, the reward is 25 tokens. During the third era, the reward is 12.5 tokens and so forth. There are forty eras total until the mining will halt. This is expected to take about 100 years at which time BCT can be used as a decentralized digital currency for Ethereum Classic.

The reward era is used to calculate the mining reward. Next, the BCT smart contract adjusts the difficulty by first determining how many Ethereum Classic blocks had been mined since the last adjustment. If less than 2016*60 Ethereum Classic blocks had been mined, BCT is being mined too quickly and the difficulty will increase. This is accomplished by reducing the size of the 'target'. When the target is smaller, valid nonces for minting are more rare and are harder to find for future mining rounds. Alternatively if BCT is being mined too slowly the target will increase in value in order to make minting more easy to accomplish. All difficulty targets are bound within minimum and maximum difficulties of 216 and 2234 respectively.

Calculating Mining Hashrate

To calculate approximate hashrate or approximate time to find a solution, the following equation can be used:

$$\text{TimeToSolveBlock (seconds)} = (\text{difficulty} * 2 ^ {22}) / \text{hashrate (hashes per second)}$$

Risks and Challenges

BCT is implemented as an Ethereum Classic ERC223 token and so its success is largely dependent on the success of the Ethereum Classic Network.