Bramble

(Whitepaper Revision 1.0.2)

Bramble: Token ecosystem for reward based systems

Abstract

There are many reward based system but they don't reward the users with cryptocurrencies. This paper gives implementation details of a token ecosystem that allows applications using it to reward people with a crytocurrency.

Bramble Components

1) A smart contract which rewards Bramble Token on verification. 2) A Bramble Token (Ethereum based cryptocurrency) 3) A Decentralized Exchange which will exchange other Ethereum Tokens / Ether for Bramble Token. 4) A mining software to mine Bramble Tokens.

Bramble Token

The cryptocurrency used by the Bramble community. For incentivizing the people in engaging on applications of the Bramble community(Lets call them Bramblers), they will be paid in BrambleToken / vested BrambleToken. This cryptocurrency token is designed to be used as a decentralized token within the Ethereum ecosystem and beyond. It avoids problems related to centralization and security because it is powered by the Ethereum Network and by globally distributed anonymous miners. Since it follows a standard protocol (ERC20), it is stored in a traditional Ethereum wallet and it is transferred using standard software which supports EIP20/ERC20 tokens. The community owns and operates the token in a fiat structure and every individual has the same power over the smart contract as any other individual.

As an ERC20 token, Bramble Token uses a traditional Ethereum account. These accounts are free and are impossible to hack or to steal from, given that the private key has not been exposed. Bramble Token / vested Bramble Token can be stored in a Ledger Nano, Trezor or any other wallet that supports ERC20 tokens.

Account System for Bramble Token

Bramble Account will be connected to your ethereum address which supports ERC20 tokens. This will be done using Metamask. All payments involving activity of the site will be done in Bramble Token / vested Bramble Token.

Bramble Token Allocation and Supply

After the initial supply of Bramble Tokens which is 250 million BRTs is over with mining, Bramble will follow a inflationary model which begins creating new tokens at an annual inflation rate of 9.5%. The inflation rate decreases at a rate of 0.01% everytime the new supply get finished. The inflation will continue decreasing at this pace until the overall inflation rate reaches 0.95%.

Of the new tokens that are generated, 44% go to fund the reward pool, which is split between problem creators/solvers/action-takers. Another 12% of the new tokens are awarded to holders of vested Bramble Token. The remaining 44% pays for the miners to power the blockchain.

Impact of Token Creation Rate

As stated in the Steem paper "It is often said that a coin with an inflationary model is not sustainable, but we know from countless real-world examples that the quantity of money does not have a direct

and immediate impact on its value, although it certainly plays a role. From August 2008 through January 2009 the US money supply 14 grew from $871B to $1,737B, a rate of over 100% per year, and then continued to grow at about 20% per year for the next six years. All told, the money supply in the US has grown by 4.59x over less than seven years. During that same time, the value of the dollar relative to goods and services has fallen less than 10%, according to the government's price index. This real-world example demonstrates that supply is only one component of price. For the first two years of Bitcoin's life the network sustained an annual inflation rate 16 of over 100%. For the first five years it was over 30%, and for the first eight years it was over 10%. All told, the total "spending". Steem requires to fund content, curation, and block production amounts to less than 10% inflation. The price of a digital commodity like STEEM is driven by both supply and demand. When a long-term holder decides to exit, the supply of STEEM on the market will increase and push the price down. This downward pressure is countered when a new long-term holder decides to buy up the STEEM and convert it back into SP. Additional supply and demand may be added due to the purchases and sales of liquid STEEM by market speculators, based on their predictions of the future market price."

Bramble Token will act similiarly to STEEM and vested Bramble Token will act similar to SP (Steem Power).

Mining for Bramble Token

Mining is not a part of the software application but nonetheless but a major part for generating Bramble Token which will be used by the Bramble community. BrambleToken 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 evalutates to true, then the nonce is a valid solution to the proof of work. The challengeNumber is just a recent Ethereum block hash. Every round, the challengeNumber updates to the most recent Ethereum block hash so future works cannot be mined in the past. The miner's Ethereum 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 of BrambleToken

When mining BrambleToken, 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 equillibrium 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 address is included in the proof of work, pools require that miners mine using the pool's ethereum 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 Proof of Work pool mining. Probability theory states that, given enough close solutions, a full solution will eventually be found.

BrambleToken Smart Contract Functions

name

Returns the name of the token - e.g. `"Bramble 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. `"BRT"`.

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 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)
```

Minting New Bramble Tokens

The Bramble Token will be deployed to the Ethereum blockchain, with the following attributes:

- No pre-mine
- No ICO
- Unlimited supply with controlled emission.
- Difficulty target auto-adjusts with PoW hashrate
- ERC20 compatibility

As such, the only way for a user to acquire Bramble is to mine them or engage with the community on the Bramble software application. The mint function is responsible for verifying the validity of the hash solution, updating the contracts internal state and issuing new Bramble.

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

            //the PoW must contain work that includes a recent ethereum block hash
(challenge number) and the msg.sender's address to prevent MITM attacks
            bytes32 digest =  keccak256(abi.encodePacked(challengeNumber, msg.sender,
nonce));

            //the challenge digest must match the expected
            if (digest != challenge_digest) revert();

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


            //only allow one reward for each challenge
            bytes32 solution = solutionForChallenge[challengeNumber];
            solutionForChallenge[challengeNumber] = digest;
            if(solution != 0x0) revert();  //prevent the same answer from awarding
twice

            reward_amount = getMiningReward();

            miners_cut = reward_amount.mul(44).div(100);
```

```
            reward_pool_cut = reward_pool_cut.add(reward_amount.mul(44).div(100));
            z_holders_cut = z_holders_cut.add(reward_amount.mul(12).div(100));

            balanceOfRewardPool[address(this)] =
balanceOfRewardPool[address(this)].add(reward_pool_cut);
            balanceOfVestingInContract[address(this)] =
balanceOfVestingInContract[address(this)].add(z_holders_cut);
            balanceOfVestingInUser[msg.sender] =
balanceOfVestingInUser[msg.sender].add(miners_cut);

            if(unfreezeDate[msg.sender] == 0){
                unfreezeDate[msg.sender] = now + timeUntilUnlocked;
                emit TokensFrozen(msg.sender, miners_cut, now);
            }

            tokensMinted = tokensMinted.add(reward_amount);
            circulatingSupply = circulatingSupply + tokensMinted;

            //Cannot mint more tokens than there are
            // assert(tokensMinted <= maxSupplyForEra);
            //set readonly diagnostics data

            lastRewardTo = msg.sender;
            lastRewardAmount = miners_cut;
            lastRewardEthBlockNumber = block.number;

             _startNewMiningEpoch();

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

           return true;

        }
```

figure 1. Bramble Token Smart Contract mint() function

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 1024 mined blocks.

Frequently Asked Questions

Does BrambleToken have its own Blockchain?

No. Bramble Token exists on the Ethereum Blockchain as a Smart Contract. This allows it to leverage a faster, more secure and modern crypto environment.

How does pool mining work with BrambleToken?

Essentially the same way that pool mining works for classic Bitcoin, except BrambleToken pools must pay gas fees to the Ethereum network.

How often does difficulty update of BrambleToken?

Every 1024 blocks.

How does the difficulty update of BrambleToken?

It increases up to 100% or down 50% with fractional changes in between in an effort to be approximately 5x slower than eth block rate, or roughly 1 minute.

Whitepaper Contributors

1. Rahul Soshte(2015rahul.soshte@ves.ac.in)

References

Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2009. http://www.bitcoin.org/bitcoin.pdf.

Logelin J and 0xBitcoin communitiy members. ERC 541 - Mineable Token Standard Draft, 2018. https://github.com/ethereum/EIPs/pull/918

Fabian Vogelsteller and Vitalik Buterin. ERC-20 Token Standard, 2015. URL https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md.

TrustNodes. The First PoW Bitcoin Like Token Launches on Ethereum, February 16, 2018. https://www.trustnodes.com/2018/02/16/first-pow-bitcoin-like-token-launches-ethereum

Vitalik Buterin. Ethereum White Paper, 2014. https://github.com/ethereum/wiki/wiki/White-Paper

Epstien J. Why Proof of Work in Bitcoin Means Proof of Value in the Real World, December 20, 2017. https://www.neverstopmarketing.com/proof-work-bitcoin-means-proof-value-real-world/

Bitfury Group Limited. "Proof of Stake versus Proof of Work", 2015. http://bitfury.com/content/5-white-papers-research/pos-vs-pow-1.0.2.pdf

https://en.bitcoin.it/wiki/Controlled_supply

Dai W. "b-money", 1998. http://www.weidai.com/bmoney.txt

Back A. "Hashcash - a denial of service counter-measure", 2002. http://www.hashcash.org/papers/hashcash.pdf

Cunningham A, Ethereum Co-Founder Announces DAICO, a new ICO Fundraising Model (January 15, 2018). https://discover.coinsquare.io/investing/daico-new-ico-fundraising-model/

0xBitcoin: The Decentralized Bitcoin Token for Ethereum. https://github.com/0xbitcoin/white-paper

Steem. An incentivized, blockchain-based, public content platform. https://github.com/steemit/whitepaper