

PurrCoin

Mu Xi

May 14, 2021

Abstract

Decentralized blockchain systems with turing complete smart contracts enable a multitude of programs. Here we discuss a simple program called a token. A token, or sub-currency (to ether), can be distributed to any number of entities and used as a means of exchange or governance. The methods of distribution are defined in the smart contract and there are various techniques that result in lower or higher decentralization. Here we investigate the *50/50 Burn + Pool* method used to initialize PurrCoin, an ERC-20 token contract that lives on the Ethereum blockchain. We found this method to be successful in distributing the token evenly across multiple holders and resulted in high decentralization – a fundamental quality of PurrCoin.

1 Ethereum

1.1 Network

Ethereum is a vast network of computers sharing information to each other in a peer-to-peer (p2p) fashion. It is essential that some of the computers on the network are *full nodes*, and store an append-only database, called the blockchain. A message sent by a computer with the intent of appending to the database is called a *transaction*. This transaction gets picked up by a miner via the p2p message stream, included in a block, and worked on. Once the puzzle has been solved, the miner can broadcast this solved block to the network, where it's work is validated by other nodes, and included at the end of the chain.

1.2 Smart Contract

Blockchain systems like Bitcoin favor security over functionality and opt for a much simpler programming language (BitcoinScript). With a limited set of OP CODES, there is less attack surface and fewer bugs. Contrastingly, Ethereum contracts have access to a *turing complete*[1] set of OP CODES, and with higher level languages like Solidity, it becomes quite easy to create full-fledged programs. With great power comes great responsibility!

2 Token

2.1 Simple Contract

A token contract at it's core is as simple as a mapping of address→amount:

Alice	10 PURR
Bob	4 PURR

All functions like transfer, burn, and mint just modify the number of tokens held at that memory location. For example, if we wanted to transfer 5 PURR from Alice to Bob the resulting map would be:

Alice	5 PURR
Bob	9 PURR

2.2 ERC-20 Contract

As tokens became more widely used and understood, common functionality was determined and standardized. In November of 2015, a standard interface for tokens was defined in ERC-20 Token Standard [2]. PurrCoin uses this standard through the well tested library OpenZeppelin [3].

3 Tokenomics

3.1 Definition

A "coined" word in the cryptocurrency space that is combination of *token* and *economics*. In essence it encapsulates all game theory and incentives of using the token. It is also used to describe the distribution method of the token.

3.2 Goals

To enable high security and integrity for downstream projects, the token **must be** highly decentralized.

3.3 No Minter

The contract has had its minting feature completely removed. There will never be more PurrCoin than what was initially created in the constructor of the contract. PurrCoin supply is hard limited at a generous amount.

3.4 50/50 Burn + Pool

To validate the transfer function and limit supply even more, half the tokens are immediately burned by sending to the Ethereum burn address:

0x00dead

The remaining tokens are then forfeited to a pool, giving us a token distribution of:

Burn Address	50%
Uniswap Pool	50%

After 3 days of being a live contract, the token pool has been distributed across more than 50 holders, with no single holder having majority.


Txn Hash	From	To	Quantity
0xa298c689710881ffb93...	0xb88dd65ee618704a68...	 Uniswap V2: PURR	500,000,000,000,000
0xf56456175b14383138...	0xb88dd65ee618704a68...	Burn Address	500,000,000,000,000
0x94c809bda7a8f8926d...	0x00000000000000000000...	0xb88dd65ee618704a68...	1,000,000,000,000,000

Figure 1: Genesis transactions for PurrCoin (newest to oldest)

4 Conclusion

In order to enable highly decentralized apps, the source utility token itself must be highly decentralized. PurrCoin uses a robust and battle-hardened token library provided by OpenZeppelin that implements ERC-20 [2, 3]. We see that by holding no pre-sale or pre-distribution of any kind, and following a disciplined method of 50/50 Burn + Pool, a highly decentralized token was created.

References

- [1] Turing completeness, “Turing completeness — Wikipedia, the free encyclopedia,” 2021. [Online; accessed 10-May-2021].
- [2] Fabian Vogelsteller, Vitalik Buterin, “Eip-20: Erc-20 token standard,” *Ethereum Improvement Proposals*, no. 20, November 2015. [Online serial]. Available: <https://eips.ethereum.org/EIPS/eip-20>.
- [3] “ERC20.” <https://docs.openzeppelin.com/contracts/4.x/erc20>. Accessed: 2021-05-01.