Build Your Own Cryptocurrency

Project Advisors: Simeon Wuthier and Sang-Yoon Chang

Summer Research Camp
Silicon Valley Cybersecurity Institute

Bhupatiraju, Ashvini (UCSC)
arbhupat@ucsc.edu
Lim, Jaehyun (Sophomore)
jaehyun0618@gmail.com
Jain, Aashvi (Sophomore)
jain.aashvi@gmail.com

Introduction

Silicon Valley Crypto for Charity is an Ethereum token currently deployed to the Ropsten Testnet. The goal of this project was to bridge the gap between cryptocurrencies and charities, and design a fully featured deflationary cryptocurrency that sends 0.5% of every transaction to a charity of the user's choice.

Our source code is located at the following URL
https://github.com/Cryptocurrency-CyberSec-Summer-Camp/charity-coin

Preliminary Information

Background

Bitcoin was proposed in 2008 by Satoshi Nakamoto [4] as a simple fund management scheme. While sufficient at the time, the decentralized finance space wanted more interactivity and functionality out of the crypto space.

Vitalik Buterin proposed Ethereum in 2013, and released the whitepaper in 2015 [5]. Ethereum is a blockchain platform that hosts smart contract-based decentralized applications (dApps), which has become the second largest currency by market capitalization as of July 2021.

Token Standardization

The smart contract functionality of Ethereum has enabled smart contracts to execute Turing-complete code on the blockchain. The Ethereum Request for Comments under proposal identification twenty (ERC-20) is a standard that dictates exactly how decentralized exchanges (DeXs) will be able to interact with the contract, thus making a smart contract that is also a token.

Technical Requirements and Our Work

Ethereum Programmability

The programming language designed by Ethereum and used by other smart-contract-based cryptocurrencies is called Solidity. Solidity is an object-oriented programming language built to run on a distributed system, namely the Ethereum Virtual Machine (EVM). Solidity is also statically-typed, and designed based off ECMAScript with some notable differences such as type management and syntax of contracts/objects.

ERC-20 Technical Requirements

Any smart contract that contains the functionalities defined by the ERC-20 standard is considered a token.

```
abstract contract ERC20Interface {
    function totalSupply() public virtual view returns (uint256);
    function balanceOf(address tokenOwner) public virtual view returns (uint256 balance);
    function allowance(address tokenOwner, address spender) public virtual view returns (uint256 remaining);
    function transfer(address to, uint256 tokens) public virtual returns (bool success);
    function approve(address spender, uint256 tokens) public virtual returns (bool success);
    function transferFrom(address from, address to, uint256 tokens) public virtual returns (bool success);

    event Transfer(address indexed from, address indexed to, uint256 tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint256 tokens);
}
```

Because Solidity is object-oriented, and contracts are objects which can have inheritance with other contracts, the ERC-20 standard is implemented as an abstract object, such that any object that inherits it will be required to also implement the functions and events defined in the ERC-20 standard. These functions are as follows:

- **totalSupply()** - Returns the total number of tokens in supply, as defined by the contract creator.
- **balanceOf(tokenOwner)** - Given an address, returns the total number of tokens in that address.
- **allowance(tokenOwner)** - Used to grant an external entity tokens owned by a different address.
- **transfer(to, tokens)** - Transfer tokens from the sender's address to the provided address.
- **approve(spender, tokens)** - The approval function required by the allowance function.
- **transferFrom(from, to, tokens)** - Transfer tokens from an external address to the provided address, assuming the transaction has been approved.

Events allow anybody to subscribe to specific blockchain activities and be notified when the event is triggered. The two events required by the ERC-20 standard are as follows:

- **Transfer(from, to, tokens)** - Triggered when any amount of tokens is sent from one user to another.
- **Approval(tokenOwner, spender, tokens)** - Triggered when funds have been approved to be spent by an external entity.

By following this standard, tokens become spendable and tradeable with other tokens and coins via Uniswap [2] and other swapping platforms.

Our Contributions

The cryptocurrency space is home to a vibrant set of currencies, platforms, dApps, and other DeFi-based services. Similarly, there exist many vibrant charities that support various causes and help incentivise improvement. We bridge the gap between DeFi and charities by building an ERC-20 token that simultaneously functions as a tradeable token and a charity-supporting platform/interface.

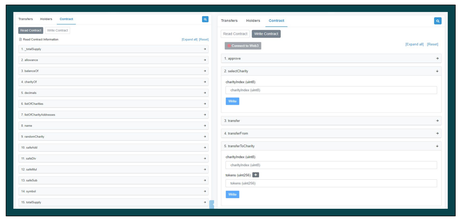
We introduce the Silicon Valley Crypto for Charity (SVCFC) token built in Solidity 0.8.6 [3], which uses Giveeth [1]. For every transaction, there are two additional required fees that help facilitate the scalability and global consensus of our service:

1. The burn fee, where 0.5% is sent to address(0), which limits the supply and incentivises holding the token. The token is deflationary from this.
2. The charity-donation fee, where 0.5% is sent through our charity management scheme.

Charity Management Scheme

We provide token holders with various charity donation options. A few of the most notable functions we incorporate into our token are as follows:

- **Charity selection** - Every account holds a token balance, as defined by ERC-20, however we also include a *charityState* mapping to each address, where every user has the ability to select a charity through our *selectCharity(charityIndex) function.* By default, an account will dynamically select a random charity for each transaction that is made.
- **Random charity selection** - We use a hash-based pseudo-random number generator (PRNG) in smart contracts by hashing the transaction's block difficulty and timestamp with *keccak256*. We then use modulo to derive an index into our array of Ethereum charities.
- **Direct transfers to a charity** - For users that prefer directly transferring tokens to a charity of the user's choice, our *transferToCharity(charityIndex)* function bypasses all fees to enable a direct transfer to any supported charity.



The Etherscan interface for these functionalities is shown in the above figure.

Tools Used

The smart contract is published on both GitHub and Etherscan. Remix was the tool used to deploy the contract to the Ethereum's Ropsten Testnet. The link to our Etherscan and GitHub repositories are as follows.

- https://github.com/Cryptocurrency-CyberSec-Summer-Camp/charity-coin
- https://ropsten.etherscan.io/token/0x4C8C16B4a5c5866fe3BBE0f0cb1d9950A3fE947E

Summary/Conclusions

In this work, we introduce our token named SVCFC. This is a charity-supporting token that includes many additional features and functionalities to help with the token's deflationary nature, as well as charity support in the form of multiple interactive functions.

Currently, this token is deployed on the Ropsten testnet, however after additional testing, we plan to migrate this to the mainnet and link it with exchanges.

Key References

[1] Giveth. [Online]. Available: https://giveth.io/. [Accessed: 01-Aug-2021].

[2] Hayden Adams, Noah Zinsmeister, and Dan Robinson. 2020. Uniswap v2 Core. Technical Report.

[3] Solidity. [Online]. Available: https://docs.soliditylang.org/en/v0.8.6/. [Accessed: 01-Aug-2021].

[4] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, May 2008, [online] Available: https://bitcoin.org/bitcoin.pdf.

[5] V. Buterin, Ethereum: A next-generation smart contract and decentralized application platform, Zug, Switzerland, 2014, [online] Available: https://github.com/ethereum/wiki/wiki/White-Paper.

Acknowledgements

We would like to acknowledge the Cybersecurity Summer Camp speakers and faculty for their efforts. Without their knowledge and expertise this project would not have been made possible.