

ADVANCE CYBER PROJECT

Topic : Creating Cryptocurrency using Blockchain Technology



SUBMITTED BY

SUBMITTED TO

Rishab Banthia - 18BCN7076

**Prof. Ajith Jubilson
Dept. of Computer Science**

INTRODUCTION

A cryptocurrency is a digital or virtual currency that is secured by cryptography, which makes it nearly impossible to counterfeit or double-spend. Many cryptocurrencies are decentralized networks based on blockchain technology—a distributed ledger enforced by a disparate network of computers. A defining feature of cryptocurrencies is that they are generally not issued by any central authority, rendering them theoretically immune to government interference or manipulation.

KEY TAKEAWAYS

- A cryptocurrency is a form of digital asset based on a network that is distributed across a large number of computers. This decentralized structure allows them to exist outside the control of governments and central authorities.
- The word “cryptocurrency” is derived from the encryption techniques which are used to secure the network.
- Blockchains, which are organizational methods for ensuring the integrity of transactional data, is an essential component of many cryptocurrencies.
- Many experts believe that blockchain and related technology will disrupt many industries, including finance and law.
- Cryptocurrencies face criticism for a number of reasons, including their use for illegal activities, exchange rate volatility, and vulnerabilities of the infrastructure underlying them. However, they also have been praised for their portability, divisibility, inflation resistance, and transparency.

TYPES OF CRYPTOCURRENCY

Coins refer to cryptocurrencies built on their independent blockchain network. The most famous example is Bitcoin (BTC), which is also the world’s largest cryptocurrency by market capitalization.

Bitcoin is powered by its native blockchain network. Similarly, Litecoin (LTC) and Ethereum (ETH) function on their respective blockchains. These blockchains may differ in their size, rules, miners, performance, etc.

Some of the popular coins are Bitcoin (BTC), Ripple (XRP), Ethereum (ETH), Dogecoin (DOGE), and Litecoin (LTC).

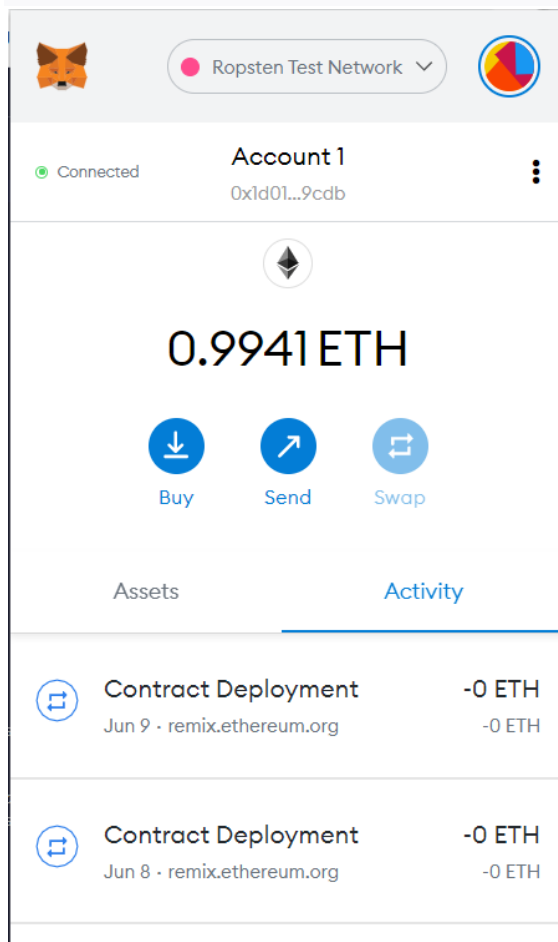
In the meantime, tokens refer to cryptocurrencies that don't have a blockchain network of their own. Instead, these cryptocurrencies are built on another blockchain. Users can create digital tokens using one of the many platforms in the DeFi (Decentralized Finance) ecosystem.

Ethereum is one of the most popular choices, thanks to its support for smart contracts. Most of the digital tokens found today are ERC-20 tokens since the Ethereum platform easily enables creating tokens on top of the Ethereum blockchain.

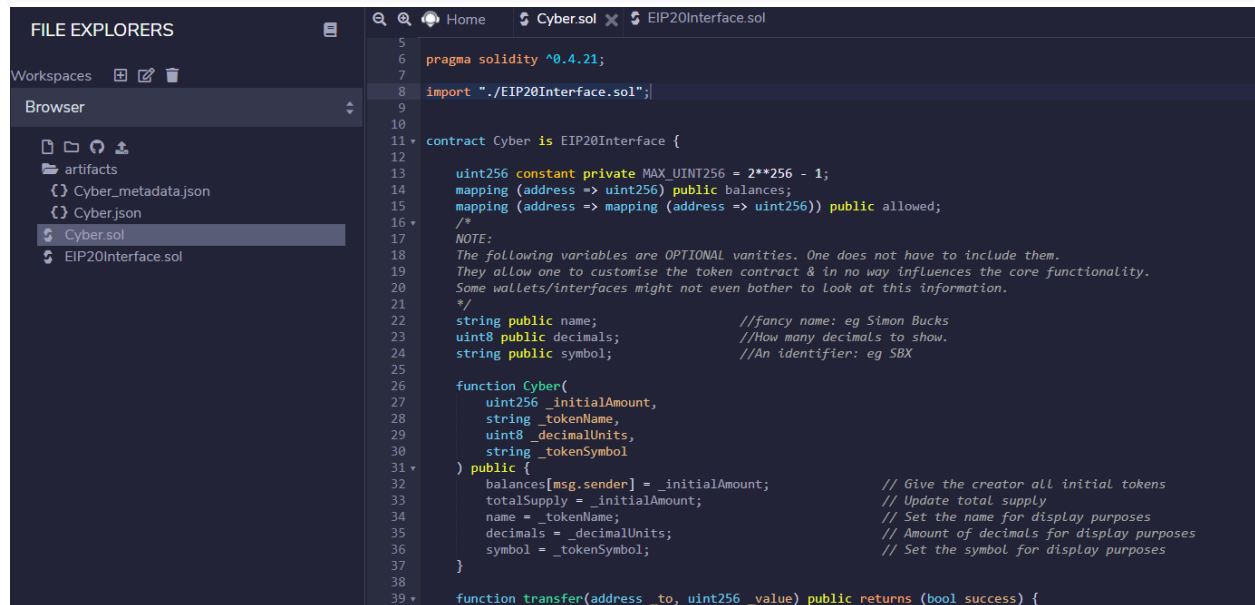
Currently, thousands of tokens exist in the market. Tether (USDT), USD Coin (USDC), DAI, UMA, and Basic Attention Token (BAT) are some of the commonly-used digital tokens out there. These tokens may have powers other than value transfer.

IMPLEMENTATION

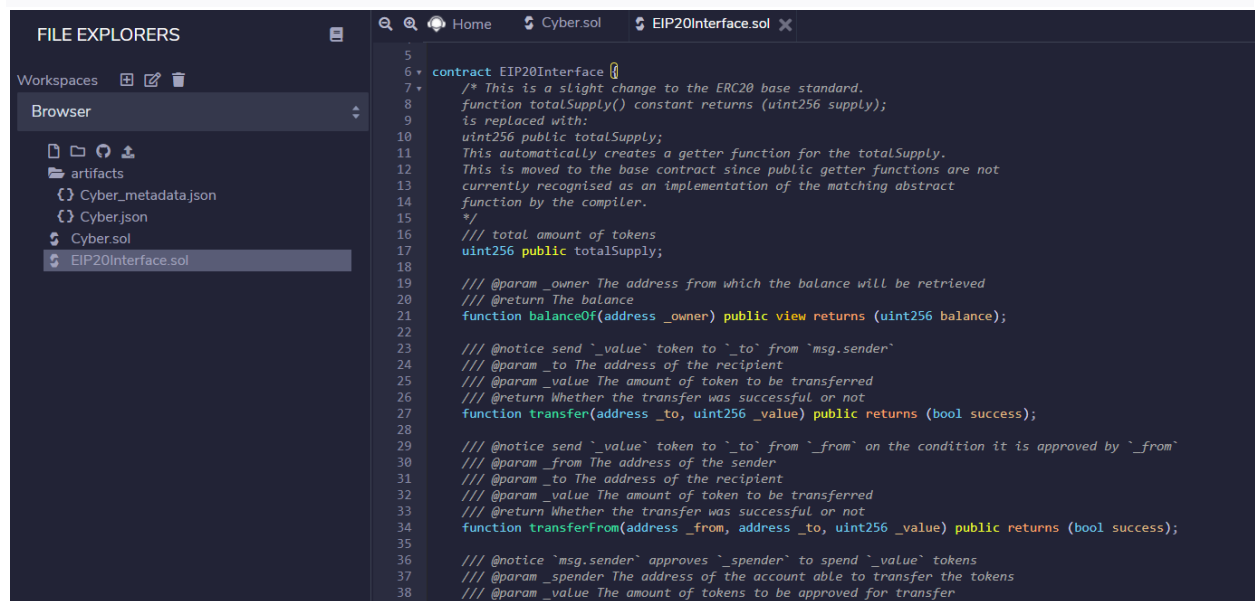
Creating a meta mask wallet and having a test ether in the ropsten test network.



Next going to remix and create two files Cyber. sol and EIP20Interface.sol



```
5
6 pragma solidity ^0.4.21;
7
8 import "./EIP20Interface.sol";
9
10
11 contract Cyber is EIP20Interface {
12
13     uint256 constant private MAX_UINT256 = 2**256 - 1;
14     mapping (address => uint256) public balances;
15     mapping (address => mapping (address => uint256)) public allowed;
16     /*
17     NOTE:
18     The following variables are OPTIONAL vanities. One does not have to include them.
19     They allow one to customise the token contract & in no way influences the core functionality.
20     Some wallets/interfaces might not even bother to look at this information.
21     */
22     string public name; //fancy name: eg Simon Bucks
23     uint8 public decimals; //How many decimals to show.
24     string public symbol; //An identifier: eg SBX
25
26     function Cyber(
27         uint256 _initialAmount,
28         string _tokenName,
29         uint8 _decimalUnits,
30         string _tokenSymbol
31     ) public {
32         balances[msg.sender] = _initialAmount; // Give the creator all initial tokens
33         totalSupply = _initialAmount; // Update total supply
34         name = _tokenName; // Set the name for display purposes
35         decimals = _decimalUnits; // Amount of decimals for display purposes
36         symbol = _tokenSymbol; // Set the symbol for display purposes
37     }
38
39     function transfer(address _to, uint256 _value) public returns (bool success) {
```



```
5
6 contract EIP20Interface {
7     /* This is a slight change to the ERC20 base standard.
8     function totalSupply() constant returns (uint256 supply);
9     is replaced with:
10     uint256 public totalSupply;
11     This automatically creates a getter function for the totalSupply.
12     This is moved to the base contract since public getter functions are not
13     currently recognised as an implementation of the matching abstract
14     function by the compiler.
15     */
16     /// total amount of tokens
17     uint256 public totalSupply;
18
19     /// @param _owner The address from which the balance will be retrieved
20     /// @return The balance
21     function balanceOf(address _owner) public view returns (uint256 balance);
22
23     /// @notice send `_value` token to `_to` from `msg.sender`
24     /// @param _to The address of the recipient
25     /// @param _value The amount of token to be transferred
26     /// @return Whether the transfer was successful or not
27     function transfer(address _to, uint256 _value) public returns (bool success);
28
29     /// @notice send `_value` token to `_to` from `_from` on the condition it is approved by `_from`
30     /// @param _from The address of the sender
31     /// @param _to The address of the recipient
32     /// @param _value The amount of token to be transferred
33     /// @return Whether the transfer was successful or not
34     function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);
35
36     /// @notice `msg.sender` approves `_spender` to spend `_value` tokens
37     /// @param _spender The address of the account able to transfer the tokens
38     /// @param _value The amount of tokens to be approved for transfer
```

Next, compiling the Cyber. sol

SOLIDITY COMPILER

COMPILER

0.4.26+commit.4563c3fc

☐ Include nightly builds

LANGUAGE

Solidity

EVM VERSION

compiler default

COMPILER CONFIGURATION

☐ Auto compile

☐ Enable optimization 200

☐ Hide warnings

Compile Cyber.sol

CONTRACT

Cyber (Cyber.sol)

Publish on Swarm

Publish on Ipfs

Compilation Details

```
5
6 pragma solidity ^0.4.21;
7
8 import "./EIP20Interface.sol";
9
10
11 contract Cyber is EIP20Interface {
12
13     uint256 constant private MAX_UINT256 = 2**256 - 1;
14     mapping (address => uint256) public balances;
15     mapping (address => mapping (address => uint256)) public allowed;
16     /*
17     NOTE:
18     The following variables are OPTIONAL vanities. One does not have to include them.
19     They allow one to customise the token contract & in no way influences the core functionality.
20     Some wallets/interfaces might not even bother to look at this information.
21     */
22     string public name; //fancy name: eg Simon Bucks
23     uint8 public decimals; //How many decimals to show.
24     string public symbol; //An identifier: eg SBX
25
26     function Cyber(
27         uint256 _initialAmount,
28         string _tokenName,
29         uint8 _decimalUnits,
30         string _tokenSymbol
31     ) public {
32         balances[msg.sender] = _initialAmount; // Give the creator all initial tokens
33         totalSupply = _initialAmount; // Update total supply
34         name = _tokenName; // Set the name for display purposes
35         decimals = _decimalUnits; // Amount of decimals for display purposes
36         symbol = _tokenSymbol; // Set the symbol for display purposes
37     }
38
39     function transfer(address _to, uint256 _value) public returns (bool success) {
40         require(balances[msg.sender] >= _value);
41         balances[msg.sender] -= _value;
42         balances[_to] += _value;
43         emit Transfer(msg.sender, _to, _value); //solhint-disable-line indent, no-unused-vars
44     }
45 }
```

0 ☐ listen on network Search with transaction hash or address

CALL [call] from: 0x1d01D7Dc619887270b14A9C31B10B97D405f9cdb to: Cyber.balances(address) data: 0x27e...f9cdb

After that deploying around 100000 crypto.

The image shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active. It displays the following settings:

- ENVIRONMENT:** Injected Web3
- ACCOUNT:** 0x1d0...f9cdb (0.99407884)
- GAS LIMIT:** 3000000
- VALUE:** 0 wei
- CONTRACT:** Cyber - Cyber.sol
- DEPLOY:**
 - _INITIALAMOUNT: "100000"
 - _TOKENNAME: Cyber
 - _DECIMALUNITS: 0
 - _TOKENSYMBOL: CYB

At the bottom of the panel is a 'transact' button.

On the right, the 'Cyber.sol' contract code is displayed. The code is as follows:

```
5
6 pragma solidity ^0.4.21;
7
8 import "../EIP20Interface.sol";
9
10
11 contract Cyber is EIP20Interface {
12
13     uint256 constant private MAX_UINT256 = 2**256 - 1;
14     mapping (address => uint256) public balances;
15     mapping (address => mapping (address => uint256)) public allowances;
16     /*
17     NOTE:
18     The following variables are OPTIONAL vanities. One does not
19     have to implement them, but it is recommended that you do
20     to make your contract more user friendly. They allow one to
21     customise the token contract & in no way affect the main
22     logic of the contract. Some wallets/interfaces might not even
23     bother to look at this information.
24     */
25     string public name; //fancy name: eg Simba
26     uint8 public decimals; //How many decimals to show
27     string public symbol; //An identifier: eg $BTC
28
29     function Cyber(
30         uint256 _initialAmount,
31         string _tokenName,
32         uint8 _decimalUnits,
33         string _tokenSymbol
34     ) public {
35         balances[msg.sender] = _initialAmount; //Initial amount
36         totalSupply = _initialAmount; //Total supply
37         name = _tokenName; //Token name
38         decimals = _decimalUnits; //Number of decimals
39         symbol = _tokenSymbol; //Token symbol
40     }
41
42     function transfer(address _to, uint256 _value) public return
43         require(balances[msg.sender] >= _value);
44         balances[msg.sender] -= _value;
45         balances[_to] += _value;
46         emit Transfer(msg.sender, _to, _value); //solhint-disable
```

Check the transaction detail and deployment contract

DEPLOY & RUN TRANSACTIONS

Deployed Contracts

▼ CYBER AT 0xFF9...D07B6 (BLOCKCHAIN)

approve address_spender, uint256_value

transfer address_to, uint256_value

transferFrom address_from, address_to, uint256_value

allowance address_owner, address_spender, uint256_value

allowed address, address

balanceOf address_owner

balances

0: uint256: 100000

call


```

5
6  pragma solidity ^0.4.21;
7
8  import "./EIP20Interface.sol";
9
10
11  contract Cyber is EIP20Interface {
12
13      uint256 constant private MAX_UINT256 = 2**256 - 1;
14      mapping (address => uint256) public balances;
15      mapping (address => mapping (address => uint256)) public allowed;
16      /*
17      NOTE:
18      The following variables are OPTIONAL vanities. One does not have to include them.
19      They allow one to customise the token contract & in no way influences the core functionality.
20      Some wallets/interfaces might not even bother to look at this information.
21      */
22      string public name; //fancy name: eg Simon Bucks
23      uint8 public decimals; //How many decimals to show.
24      string public symbol; //An identifier: eg SBX
25
26      function Cyber(
27          uint256 _initialAmount,
28          string _tokenName,
29          uint8 _decimalUnits,
30          string _tokenSymbol
31      ) public {
32          balances[msg.sender] = _initialAmount; // Give the creator all initial tokens
33          totalSupply = _initialAmount; // Update total supply
34          name = _tokenName; // Set the name for display purposes
35          decimals = _decimalUnits; // Amount of decimals for display purposes
36          symbol = _tokenSymbol; // Set the symbol for display purposes
37      }
38
39      function transfer(address _to, uint256 _value) public returns (bool success) {
40          require(balances[msg.sender] >= _value);
41          balances[msg.sender] -= _value;
42          balances[_to] += _value;
43          emit Transfer(msg.sender, _to, _value); //solhint-disable-line indent, no-unused-vars

```

You have successfully created your Cyber test crypto.

Now to check go to ropsten.etherscan.io



All Filters
Search by Address / Txn Hash / Block / Token / E

Ropsten Testnet Network Home Blockchai

Transaction Details

Overview

State

[This is a Ropsten **Testnet** transaction only]

② Transaction Hash: 0x3971db4e7e0741d933f6e97595164e00b36d3f489066c3062459c67ddba22bd1

② Status: Success

② Block: 10404866 13 Block Confirmations

② Timestamp: ③ 3 mins ago (Jun-09-2021 05:38:20 PM +UTC)

② From: 0x1d01d7dc619887270b14a9c31b10b97d405f9cdb

② To: [Contract 0xff9a9fcc13a40731e3ebcaa7232d965fdd3d07b6 Created] Success

② Value: 0 Ether (\$0.00)

② Transaction Fee: 0.000845867 Ether (\$0.00)

② Gas Price: 0.000000001 Ether (1 Gwei)

Ⓜ This website uses cookies to improve your experience and has an updated Privacy Policy.
Got it

This proves that our transaction was successful.

FUTURE GOALS

- Create a cryptocurrency token on the Binance Smart chain.
- Give liquidity to cryptocurrency using the platform Pancake Swap
- Giving it value and increasing the market capitalization and adding liquidity to existing crypto that we have and use it for trading.
- This would help people to become cashless and try to pay using cryptocurrencies which will be the future of payment where our cryptocurrency will use very little fee for the transaction and make it more profitable after investing in it.

CONCLUSION

We have successfully tested the crypto creation and have performed this task on the remix for creating the crypto and ropsten test network which acted as the testing network and confirmed it using the ropsten.etherscan.io platform.