International Conference on Advances in Materials, Computing and Communication Technologies
In Association withInternational Journal of Scientific Research in Science and Technology
Volume 9 | Issue 1 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)

Implementation of ERC20 Token Using Smart Contract with Blockchain Technology

Dr. J. Preetha*[1], K. Vijay[2], P. Meyanandhan[2], K. Praveen Kumar[2]
[1]Professor, Department of Computer Science and Engineering, Muthayammal Engineering College, Tamilnadu, India
[2]Department of Computer Science and Engineering, Muthayammal Engineering College, Tamilnadu, India

ABSTRACT

The blockchain technology has been an essential part due to its decentralization and security, some of its applications are decentralized voting system and transactions. The most important feature of blockchain is smart contract. The smart contract are the lines of code similar to agreement that runs on the top of blockchain to execute a process. Solidity is a common language used to design the smart contract and smart contract are stored in public database and execute automatically and cannot be changed once executed. Smart contract are not controlled by the user and they are deployed to the blockchain network and execute as programmed. Ethereum is a decentralized smart contract which runs on its own native platform.

Index Terms: Blockchain, Smart contract, Solidity, Ethereum

I. INTRODUCTION

Cryptocurrency is a type of digital currency used as a form of currency that can be transferred without the use of bank. Transaction between the two persons are usually done by the centralized platforms like bank and without the involvement of bank the transaction is not possible and to avoid such problem, cryptocurrency can be used as a mode to transfer our assets. Sometimes the bank may have some security issues and limited amount of asset can only be transferred and in cryptocurrency there is no such problems. Large amount of asset can be transferred in the form of cryptocurrency and no network issues and security issues. While making a transaction with the bank we may face high transaction fee and to avoid large transaction fee, cryptocurrency can be used and very low transaction fee is used to transfer our assets.

Own cryptocurrency and tokens can be created in the Ethereum platform. ERC 20 is a token that runs on the Ethereum blockchain and many ERC 20 tokens run on the Ethereum and Ethereum is used as the main asset to transfer the asset. Ethereum is used as gas fee to transfer the cryptocurrency and very low gas is used to transfer the digital asset and the transaction will not fail due to network issues and to speed up the transaction more gas needed to be used and very fast transaction can be possible as compared to the bank.

Smart contract is the set of code that runs on the top of the blockchain and it cannot be changed after the executed and the smart contract execute automatically as programmed and so it is very secure to transfer asset between the two parties. The smart contract must be deployed to the network to make the transaction and the remaining process will be automated by the smart contract with safe and secure.

II. BLOCKCHAIN TECHNOLOGY

Blockchain is a decentralized digital ledger that records the transaction from many computers as blocks. The block contains list of hashes of previous blocks and transaction data. Once the data is recorded, the data cannot be altered or changed without changing the entire block.

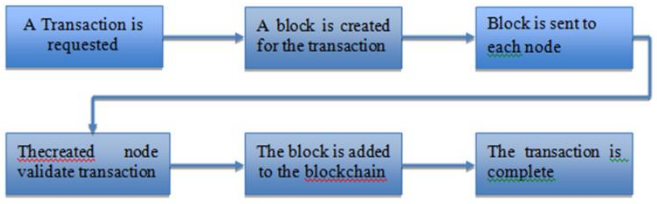

A Transaction is requested
A block is created for the transaction
Block is sent to each node
Thecreated node validate transaction
The block is added to the blockchain
The transaction is complete

Fig 1: Working Process in Blockchain

III. SMART CONTRACT

Smart contract is a set of program which implements set of operations as programmed. Smart contract runs in blockchain which is immutable once executed. Solidity is a high level language which is used to for smart contract creation. The smart contracts are executed by paying cost and the cost depends on the gas of the cryptocurrency. Smart contract execute automatically when the conditions are met and Smart contract audit is the process of testing the smart contract for design issues and rectify the bugs and issues in the smart contract. Smart contract is successfully deployed in the Ethereum network.

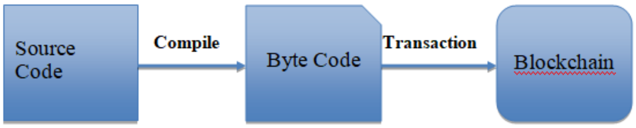


Fig.2: Smart Contract Deployment

IV. REQUIREMENTS

1. Text editor:

Text editor is used to edit the smart contract before Deployment.

2. Extension:

Metamask extension is used to make transaction for deployment of the Smart contract.

3. Compiler:

Remix Ethereum is an online compiler and used for creating, compiling and deploying the smart contract.

4. Wallet:

Wallet is required to deploy the smart contract and the wallet address is recorded in the deployment of smart contract.

5. Byte Code:

Byte code is generated after the smart contract and the byte code is an assembly language made up of many opcodes. Byte code is a low level language that is compiled from high level language like solidity and smart contract run on any machine and so the byte code is not human readable and machine readable.

```
Pragma solidity ^0.4.18;
Contract sample
{
……………
}
```

6. Metamask Extensison:

Metamask is a google chrome extension and this extension is used to connect local Ethereum network to the account and interact with smart contracts. Metamask extension is available in the chrome web store and install the plugin and enable the plugin to use the metamask.

7. Verifying smart contract:

Smart contract will be public after the deployment and the verifying the smart contract is the best practice after deployment of smart contract. Any one can view the smart contract, once it is deployed and nothing will be hidden. Once contract is deployed, it will be in the Ethereum blockchain and the etherscan is the site for Ethereum blockchain and enter the contract address generated at the deployment.

Steps to verify smart contract:

1. Click the contract tab to verify the smart contract
2. Enter verify and publish smart contract.
3. Enter the contract address and compiler version.
4. Choose the solidity files that need to be verified
5. Upload the solidity files and click to upload selected files
6. Click verify and publish.
7. Smart contract is verified successfully.

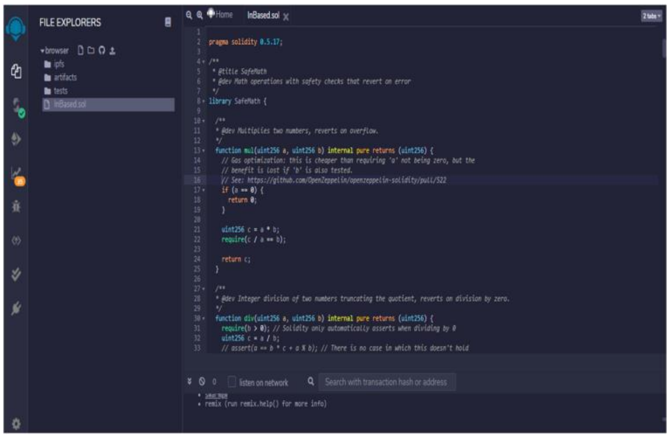
8. Functionalities in smart contract:

Many functions are used to create a smart contract and the smart contract can be created in a text editor or in the remix IDE. Each function performs different and unique operation in the solidity. Function is a set of code which can be use anytime by calling the function name. Writing same code again and again will be avoided by the functions. Solidity support modular code to write own functions in solidity. Function keyword is used to define the function in solidity and the example for defining the function is shown below.

```
function function-name (parameter-list) scope
returns()
 {
//statements
}
```

9. Implementation of Smart Contract:

Smart contract is written in the solidity and the smart contract must be saved in the extension of .sol. The remix Ethereum is an online platform for designing, compiling and deploying the smart contract. It has updated versions of compilers and many folders can be created as based on the requirements. Click the file option to create a new file and name it as filename.sol and there is git integration to store the smart contract in the github.



```

pragma solidity 0.5.17;

/**
 * @title SafeMath
 * @dev Math operations with safety checks that revert on error
 */
library SafeMath {

  /**
  * @dev Multiplies two numbers, reverts on overflow.
  */
  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
    // benefit is lost if 'b' is also tested.
    // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
    if (a == 0) {
      return 0;
    }

    uint256 c = a * b;
    require(c / a == b);

    return c;
  }

  /**
  * @dev Integer division of two numbers truncating the quotient, reverts on division by zero.
  */
  function div(uint256 a, uint256 b) internal pure returns (uint256) {
    require(b > 0); // Solidity only automatically asserts when dividing by 0
    uint256 c = a / b;
    // assert(a == b * c + a % b); // There is no case in which this doesn't hold
```

```
• remix (run remix.help() for more info)
```

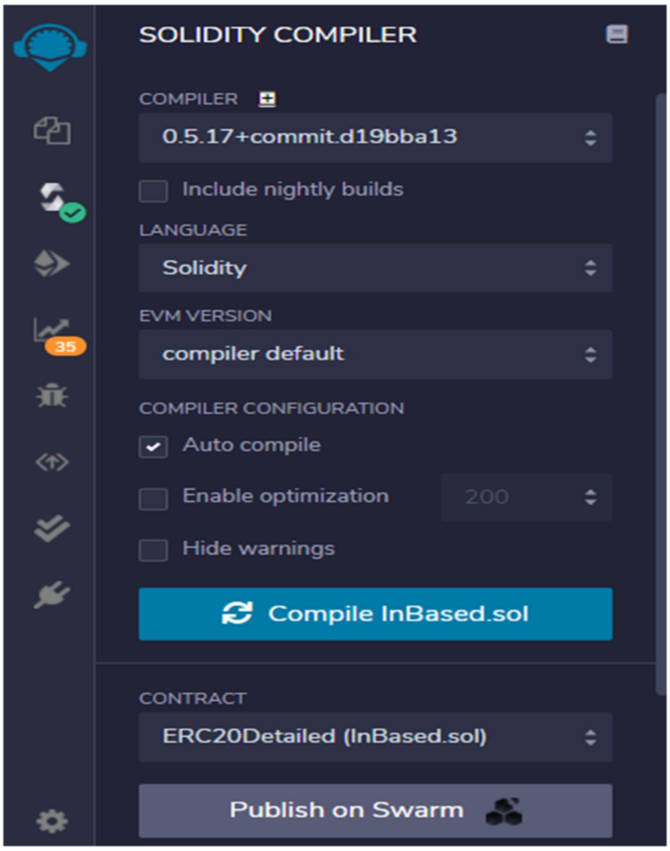
Fig. 3: Smart contract in Online platform

The cryptocurrency have some properties like total supply, token name and token symbol and owner account and these are the details that needs to be defined in the smart contract. Token name name is the identifier of the token and it represent the info of the cryptocurrency. Total supply is the total number of cryptocurrencies needs to be created in the smart contract and it cannot be changed after smart contract deployment.

The Owner account needs to be defined in the smart contract to use burn functions to burn the token if required. Remix Ethereum is the online integrated development environment to describe the smart contract and compile the smart contract. Code the smart contract in the IDE and change the variables of the token in the smart contract. The total supply of the token must be related with the decimal of the currency to mention the number of digits of the currency. The smart contract must be deployed in the testnet to avoid the errors in the mainnet and once the smart contract successfully deployed in the testnet and it will deploy in the mainnet without any errors. The deployed smart contract can be viewed in the etherscan to view all the records and transaction of the smart contract.

V. COMPILING AND DEPLOYING SMART CONTRACT

Smart Contract is compiled in the online compiler platform Remix Ethereum and it has all compiler versions. The solidity files can be compiled in the online compiler and support two programming languages as solidity and Yul. Solidity files can be compiled in many versions and no need for any application to compile the smart contract and the Bytecode will be displayed after the compilation of the smart contract.



SOLIDITY COMPILER

COMPILER
0.5.17+commit.d19bba13
□ Include nightly builds

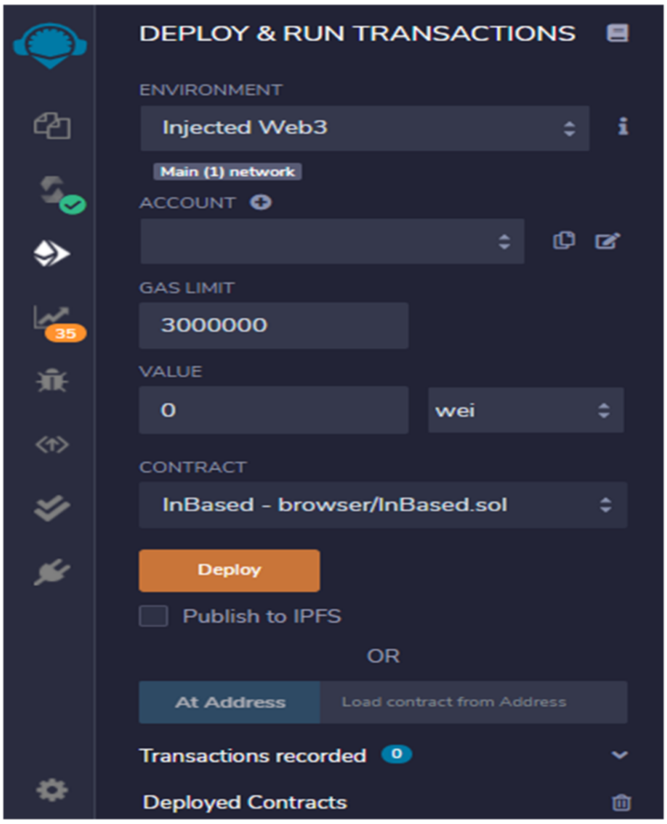
LANGUAGE
Solidity

EVM VERSION
compiler default

COMPILER CONFIGURATION
☑ Auto compile
□ Enable optimization 200
□ Hide warnings

CONTRACT
ERC20Detailed (InBased.sol)

Fig. 4: Compiling of Smart contract



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT
Injected Web3
Main (1) network

ACCOUNT

GAS LIMIT
3000000

VALUE
0 wei

CONTRACT
InBased - browser/InBased.sol

□ Publish to IPFS

OR

Transactions recorded 0

Deployed Contracts

Fig.5 Deploying of smart contract

Smart contract is deployed in the same platform and three environments are available to deploy the smart contract and the injected web 3 directly connect with the wallet and make the transaction and select the contract that needs to be deployed and select deploy and the smart contract is successfully deployed and it is recorded in the blockchain. The deployed contracts will be displayed below.

VI. CONCLUSION

The Cryptocurrency is created from the proposed system and two solidity files are created to perform various functions to create it. The smart contract cannot be changed once it is deployed and different tasks are performed by the smart contract automatically as programmed and no need for any manual operation.Smart contract needs to be audited to ensure that there is no bug or issues in the smart contract and smart contract audit helps to remove the design issues in the contract. The cryptocurrency is launched in the blockchain and they needs to be implemented in the exchange to buy and sell and many other operations can be performed like staking and farming as they needs to be predefined in the smart contract.

VII. REFERENCES

[1]. S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System”, 2008. Available: https://bitcoin.org/bitcoin.pdf

[2]. Massimo Bartoletti, Tiziana Cimoli, and Roberto Zunino. Fun with bitcoin smart contracts.

[3]. Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151:1–32, 2014.

[4]. V. Buterin, “A next-generation smart contract and decentralized application platform.," Available online at: https://github.com/ethereum/wiki/wiki/White-Paper/ Accessed 19/02/2017.

[5]. Gianni Fenu, Lodovica Marchesi, Michele Marchesi, and Roberto Tonelli. The ico phenomenon and its relationships with ethereum smart contract environment. arXiv preprint arXiv:1803.01394, 2018

[6]. K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, “Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab," in International Conference on Financial Cryptography and Data Security, pp. 79-94, Springer, 2016.

[7]. Saman Adhami, Giancarlo Giudici, Stefano Martinazzi,” Why do businesses go crypto? An empirical analysis of Initial Coin Offerings”, Journal of Economics and Business

[8]. W. Egbertsen, G. Hardeman, M. van den Hoven, G. van der Kolk, and A. van Rijsewijk, “Replacing paper contracts with ethereum smart contracts," 2016.

[9]. https://github.com/ethereum/EIPs/blob/master/EIPS/eip20.md

[10]. Melanie Swan. Blockchain: Blueprint for a new economy. " O’Reilly Media, Inc.", 2015.

[11]. https://tokenmint.io/blog/erc-20-tokens-and-how-to-createyours.html

[12]. G. Bigi, A. Bracciali, G. Meacci, and E. Tuosto, “Validation of decentralised smart contracts through game theory and formal methods," in Programming Languages with Applications to Biology and Security, pp. 142-161, Springer, 2015.

[13]. https://solidity.readthedocs.io/en/v0.5.3/introduction-tosmart-Scontracts.html

[14]. V. Buterin, “On public and private blockchains," Available online at: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains

[15]. J. Stark, “Making sense of blockchain smart contracts," Available online at: http://www.coindesk.com/making-sense-smart-contracts