PANDIT DEENDAYAL ENERGY UNIVERSITY

Raisan Village, Gandhinagar, Gujarat 382007

ACADEMIC YEAR

(2021 – 2022)

PROJECT REPORT ON

(Implementation of an Ethereum Based Blockchain Chat Application by using a Smart Contract)

M. TECH SEMESTER: 2

(CYBER SECURITY)

SUBJECT: CAPSTONE PROJECT 2

SUBJECT CODE:

SUBMITTED BY:

SR NO.	NAME	ENROLLMENT NUMBER
1	CHAUHAN DIVYANG J.	21MCS002

GUIDED BY

Dr. Kaushal Shah

Appendix 1

PANDIT DEENDAYAL ENERGY UNIVERSITY

Raisan Village, Gandhinagar, Gujarat 382007

CERTIFICATE

This is to certify that the student namely CHAUHAN DIVYANG J OF 2nd SEMESTER (M.TECH CYBER SECURITY) have successfully completed the course work and related tasks for the course of CAPSTONE PROJECT " ***Implementation of an Ethereum Based Blockchain Chat Application using Smart Contract*** *" during the academic term ending in the month of April 2022.*

Date:

Place:

GUIDED BY

Dr. Kaushal Shah

Appendix 2

- ## ACKNOWLEDGEMENT:

I am Divyang J Chauhan of M. Tech Cyber Security at Pandit Deendayal Energy University. I am preparing a project on the **Implementation of an Ethereum Based Blockchain Chat Application using Smart Contract**. I wholeheartedly express our sincere gratitude to Dr. Kaushal Shah who guided us on the project. I am also thankful to all our teachers for explaining a critical aspect of topics related to the project.

Appendix 3

LIST OF FIGURES

Appendix 5

TABLE OF CONTENTS

ABSTRACT

As we all know in this generation of social media and chatting everyone is connected to each other with different communication techniques like a chat application, video conferencing, email, etc. now these things work on a centralized system where a server can control every communication to ensure user's privacy these applications implement some privacy security setting by which only sender and receiver can see their messages and in most of the cases it will be impossible to decrypt user's messages. In this implementation, we will see how a decentralized system can be helpful to secure communication. Here we are going to use Ethereum Blockchain and smart contract to make a web-based chat application.

Chapter 1: Introduction

➢ 1.1 Introduction:

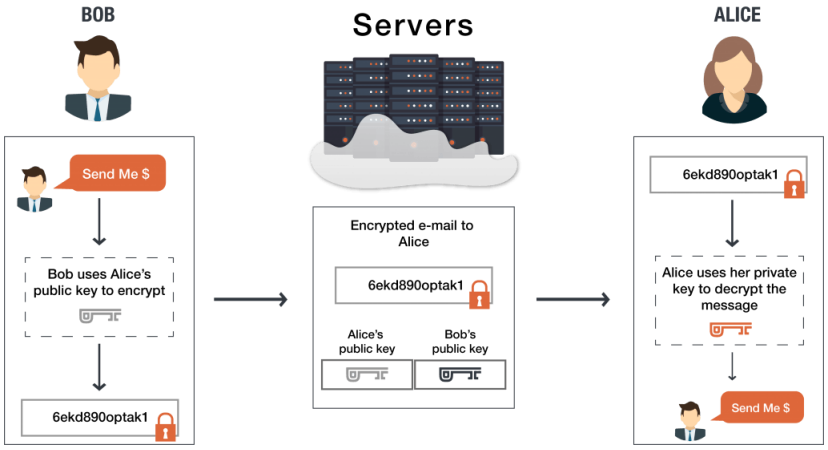
- Blockchain is a technology where we can store data inside a block data structure and make a sequence of that blocks. the blocks mainly contain detail about the transaction and in which time spam the transaction occurred, and to connect it to the previous block we can use the previous block's hash value.
- Hash is used to protecting data and in hashes, we encrypt the original message in hexadecimal format. There are different hashes available like MD5, SHA256, etc. blockchain system is widely used in cryptocurrencies where it is used to track the activity of the currencies.
- Blockchain is a solution where we need trust and authentication from two different parties. Nowadays where every communication is handled by a centralized system it may happen that our data will not be privately secure to overcome this situation, we can use blockchain to secure the communication.

➢ 1.2 Aim and Objective:

- In this project, we are going to make a decentralized Ethereum blockchain-based chat application.
- By using blockchain technology can we secure our communication with two different entities?
- Is blockchain reliable for daily communication?

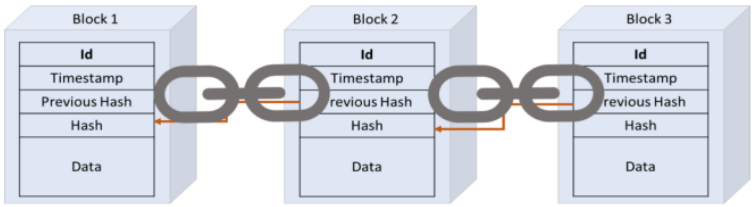
➢ 1.3 End-to-End Encryption and Blockchain:

- **E2EE** is a technique that is used to secure communication between two people. And no third party can directly access the data from any system. In E2EE message is encrypted at the sender side and only the receiver can decrypt it. Popular messaging applications like WhatsApp and Facebook are using E2EE.
- E2EE works on an asymmetric cryptographic technique where the message is encrypted using the sender's public key and that is decrypted using the recipient's private key.
- To authenticate the generated key, we have to use the certificate that is digitally signed by a recognized certificate authority.



1 END-to-END Encryption

- **Blockchain** is a decentralized distributed storage mechanism that is shared with different peers of a computer network. Blockchain stores information digitally in blocks. Blockchain is mainly used for cryptocurrencies to maintain security and to record decentralized transactions. Blockchain provides the security of data without the need for third-party authentication [2].
- The main difference between traditional centralized storage mechanisms and blockchain is how data is structured. In blockchain number of data stored in one block most likely contains the Hash value of that block, original data, timestamp, and a hash of the previous block. Once a block is full, we linked it to the previous block. And forming a chain of blocks.



2 Blockchain

- A block may contain various elements like
- **Magic Number:** a number that is used to identify that block for a particular network.
- **Block Size:** sets the size of the block so that only a limited amount of data can be stored in the block.
- **Block header:** information about the block.
- **Transaction Counter:** a number that represents how many transactions are stored in the block.
- **Transactions:** list of all transactions. The size of the transaction element is the largest because it stores most of the information.
- In blocks of blockchain also contains
  - **Hash of that block**
  - **Timestamp**
  - **Data**
  - **Previous Block Hash**

➢ Types of Blockchains:

- Blockchain is classified into three categories.
- **1) Permissionless Public Blockchain**
- A blockchain that every user in the world is allowed to use without any prior permission. Users can read, and make transactions also. That means you can be a part of any network. There is no way to censor transactions on the network [3]. Ex. Bitcoin, Ethereum.
- **2) Permissioned Private Blockchain**
- A blockchain where you need permission to access, read, and modify the blockchain. Only know peers are allowed to participate in the network [3].
- **3) Hybrid Blockchain**

Chapter 2: Software Requirements and Specification

➢ 2.1 Introduction

• 2.1.1 Purpose:

- Make a decentralized chat application based on Ethereum Blockchain.

• 2.1.2 Project Scope:

1) Project Goals:

- Introduction to End-to-End Encryption and Blockchain.
- Introduction to Smart contract.
- Deploy and Run of Smart contract to a test on an Ethereum blockchain

2) Software Requirements:

- Remix IDE.
- Meta musk Browser Extension.
- Solidity.

3) Tasks Of a Keylogger:

- Send and Receive a message using this smart contract on an Ethereum Blockchain.

4) Working:

- First we write a basic Smart Contract that will enable people to chat on the blockchain.
- Smart Contract will have two functions:
- 1) Allow a user to post a message that will be fully stored on the blockchain.
- 2) Second function will allow us to retrieve the messages
- 3) After compiling now we have to deploy it on a test blockchain here we are using Metamask chrome extension and Rinkeby test network.

- **Rinkeby Test Network:**
  - The blockchain is immutable. A deployed smart contract can't be modified anymore. While we've covered our BlockchainChat smart contract with a few tests, it might not be wise to deploy it to Ethereum Mainnet yet.
  - Deploying a smart contract cost a lot of gas, which means a lot of real money. We might want to test it in "real condition" for free first!
  - There are what we call Testnets (Test network). Rinkeby is one of them. A testnet is an environment that is pretty much like Mainnet, despite the fact that it's here to act as a sandbox.
  - It's a testing environment, that can be used to run your smart contract in almost real condition, without involving real assets (we will see in a bit that we will still use ETH, but fake ETH!).
- **How we can get ETH to deploy our smart contract:**
  - https://rinkebyfaucet.com/ by this link we can get o.1 ETH in our account for testing of our smart contract.
  - Put Metamask account number to get ETH in your account.
- **Ethereum:**
  - Ethereum is a public or permissionless Blockchain platform. Ethereum is one of the blockchain platforms to build decentralized applications. Ethereum Platform provides the flexibility to not only store transaction details on the block but also code snippets, which are termed as smart contracts[4]. The use of Smart Contracts makes the ethereum platform programmable. Ethereum follows WEB 3.0 Architecture.

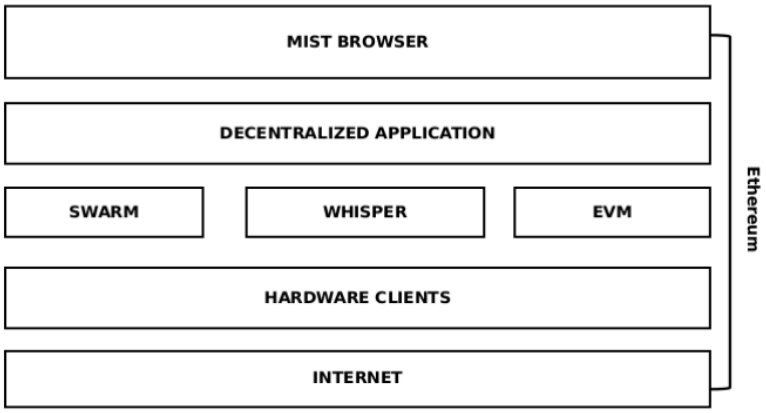


Figure 4. Web 3.0 Architecture - Representing Ethereum

- **Smart Contract:**
  - Smart contracts are actually a business logic that is being developed as per requirements. It is used for automation and executes a specific task based on some events, i.e. Got automatically triggered when some events are executed. In the Ethereum platform Smart Contracts are written in "Solidity", the Ethereum State Machine or (Ethereum Virtual Machine) converts the solidity code into low-level machinery language called Opcodes. Opcodes, often known as operational codes, is a set of instructions to execute a specific task. Initially, there were 140 Unique

Opcodes. These opcodes together make the Ethereum machine a Turing complete.

- **Solidity:**
  - Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behaviour of accounts within the Ethereum state.
  - Solidity is a curly-bracket language designed to target the Ethereum Virtual Machine (EVM). It is influenced by C++, Python and JavaScript.
  - Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.

3) Code:

1) blockchainchat.sol:

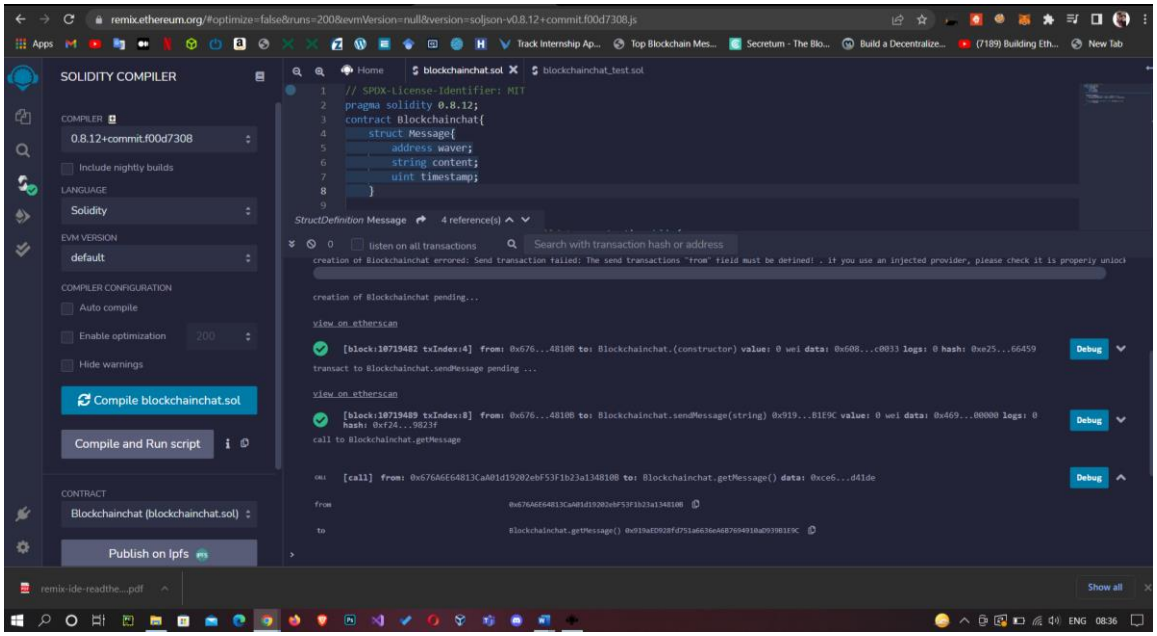
- So we said we want to store messages in our contract. More than the message, we need to also store the address of the user that has sent it and also a timestamp of when the message was added.

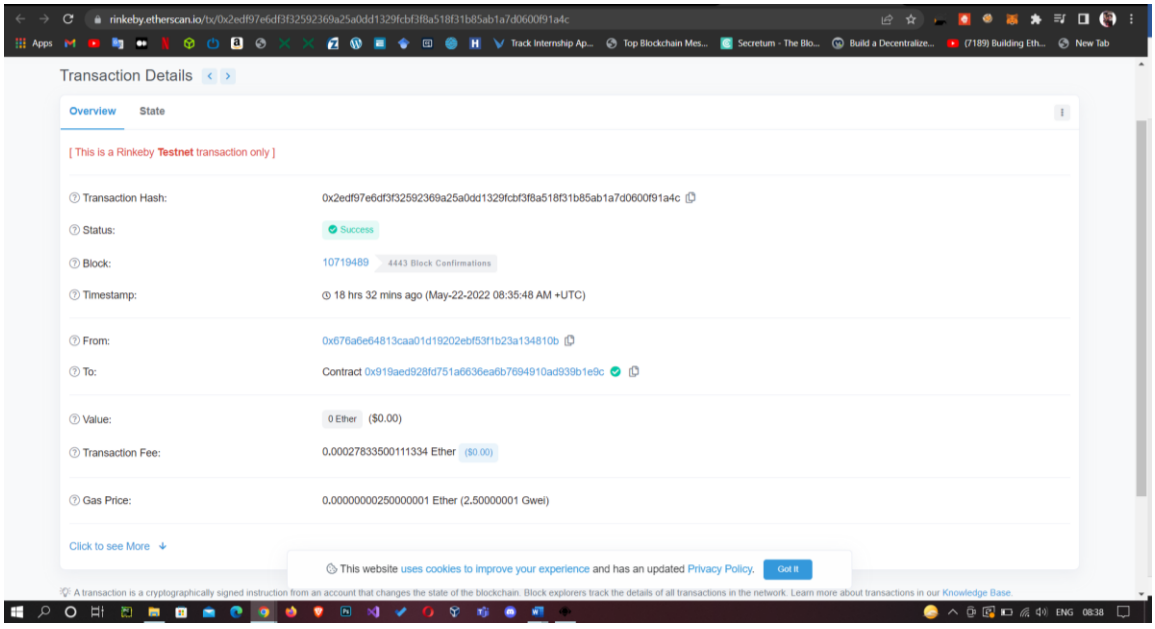
```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.12;
contract Blockchainchat{
    struct Message{
        address waver;
        string content;
        uint timestamp;
    }

    Message[] messages;
    function sendMessage(string calldata _content) public{
        messages.push(Message(msg.sender,_content,block.timestamp));
    }
    function getMessage() view public returns (Message[] memory){
        return messages;
    }
}
```

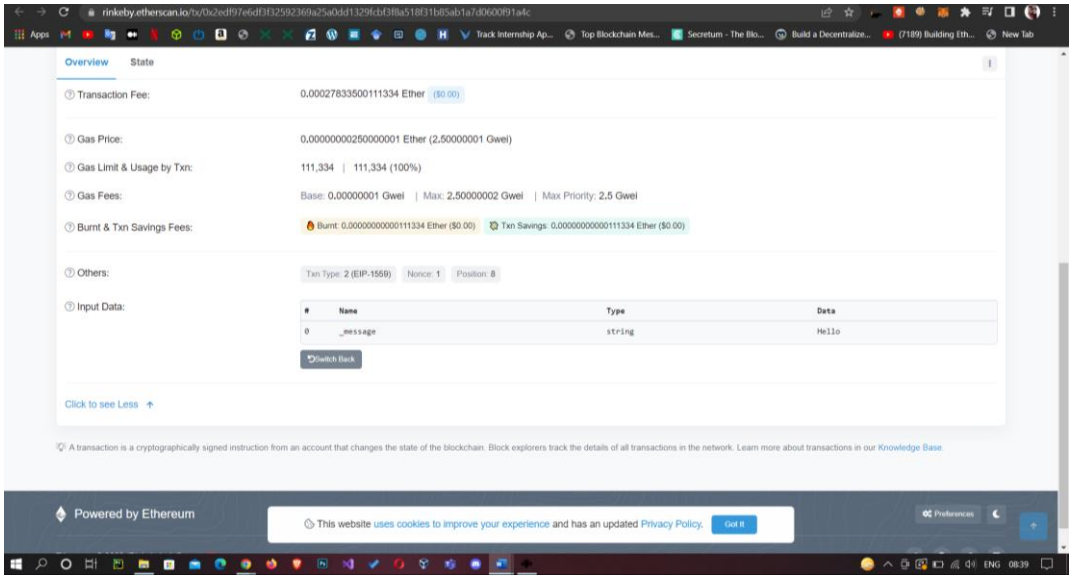
2) blockchainchat_test.sol:

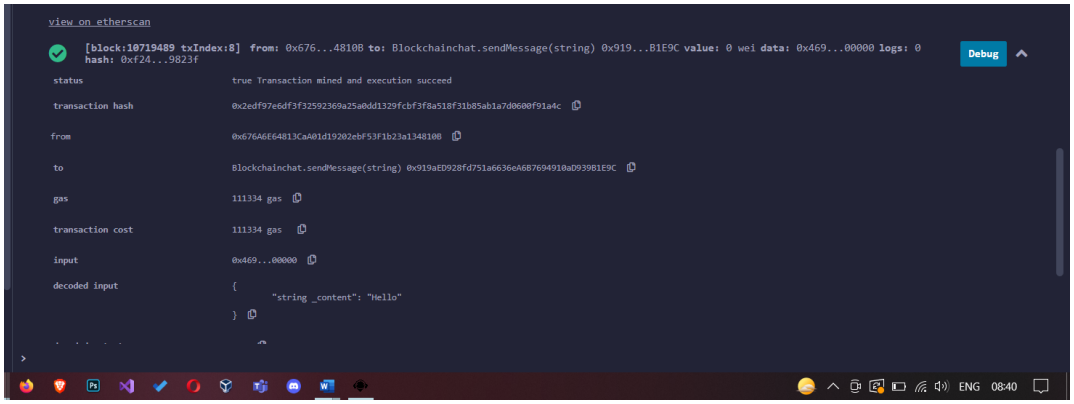
```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.12;
import "remix_tests.sol";
import "../contracts/blockchainchat.sol";
contract BlockchainChatTest {
  Blockchainchat blockchainChatToTest;
  /// 'beforeAll' runs before all other tests
  function beforeAll () public {
    blockchainChatToTest = new Blockchainchat();
  }
function checkSendMessage () public {
    // Send a first message
    blockchainChatToTest.sendMessage("Hello World!");
    // Ensure the messages variable contains 1 message
    Assert.equal(blockchainChatToTest.getMessage().length, uint(1), "messages
state variable should contain 1 message");
    // Ensure that our first message's content is "Hello World!"
    Assert.equal(blockchainChatToTest.getMessage()[0].content, string("Hello
World!"), "The first Message in message should be \"Hello World!\"");
    // Send a second message
    blockchainChatToTest.sendMessage("This chat is super fun.");
    // Ensure the messages variable contains 2messages
    Assert.equal(blockchainChatToTest.getMessage().length, uint(2), "messages
state variable should contain 2 messages");
  }
}
```

4) Output:

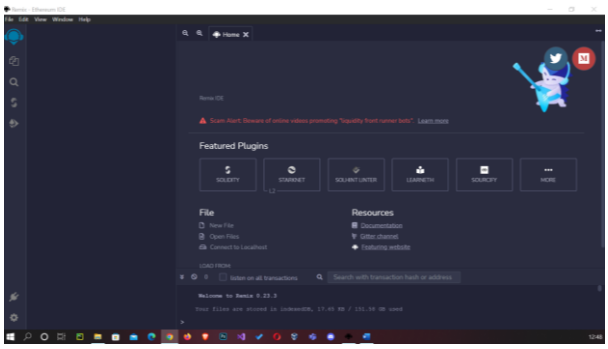

SOLIDITY COMPILER
COMPILER
0.8.12+commit.f00d7308
Include nightly builds
LANGUAGE
Solidity
EVM VERSION
default
COMPILER CONFIGURATION
Auto compile
Enable optimization
Hide warnings
Compile blockchainchat.sol
Compile and Run script
CONTRACT
Blockchainchat (blockchainchat.sol)
Publish on Ipfs
// SPDX-License-Identifier: MIT
pragma solidity 0.8.12;
contract Blockchainchat{
struct Message{
address waver;
string content;
uint timestamp;
}
creation of Blockchainchat pending...
view on etherscan
transact to Blockchainchat.sendMessage pending ...
call to Blockchainchat.getMessage

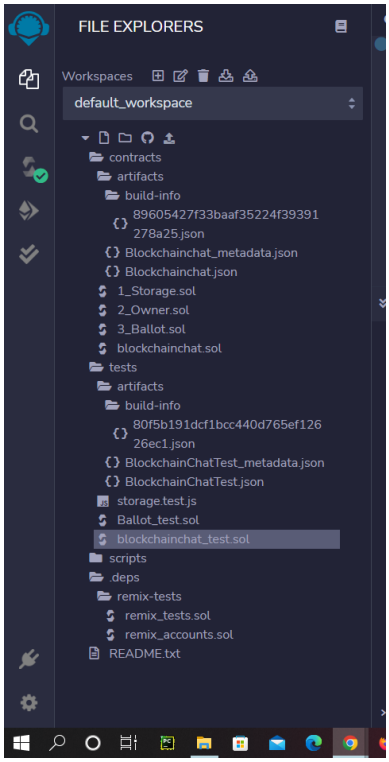

Transaction Details
Overview
State
[This is a Rinkeby Testnet transaction only]
Transaction Hash:
0x2edf97e6df3f32592369a25a0dd1329fcbf3f8a518f31b85ab1a7d0600f91a4c
Status:
Success
Block:
10719489
4443 Block Confirmations
Timestamp:
18 hrs 32 mins ago (May-22-2022 08:35:48 AM +UTC)
From:
0x676a6e64813caa01d19202ebf53f1b23a134810b
To:
Contract 0x919aed928fd751a6636ea6b7694910ad939b1e9c
Value:
0 Ether ($0.00)
Transaction Fee:
0.00027833500111334 Ether ($0.00)
Gas Price:
0.0000000250000001 Ether (2.50000001 Gwei)
Click to see More
This website uses cookies to improve your experience and has an updated Privacy Policy.
Got it

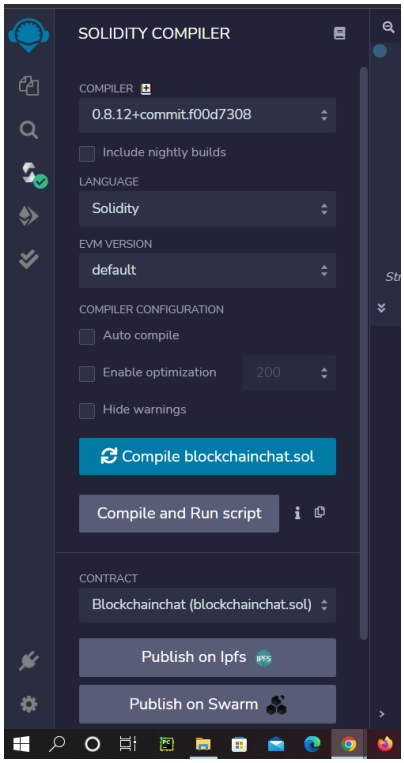

rinkeby.etherscan.io/tx/0x2edf97e6df3f32592369a25a0dd1329fcbf3f8a518f31b85ab1a7d0600f91a4c
Overview
State
Transaction Fee:
0.000278335001113 34 Ether
Gas Price:
0.000000002500000001 Ether (2.50000001 Gwei)
Gas Limit & Usage by Txn:
111,334 | 111,334 (100%)
Gas Fees:
Base: 0.00000001 Gwei | Max: 2.50000002 Gwei | Max Priority: 2.5 Gwei
Burnt & Txn Savings Fees:
Others:
Input Data:
Name Type Data
0 _message string Hello
Switch Back
Click to see Less
A transaction is a cryptographically signed instruction from an account that changes the state of the blockchain. Block explorers track the details of all transactions in the network. Learn more about transactions in our Knowledge Base.
Powered by Ethereum
This website uses cookies to improve your experience and has an updated Privacy Policy.
Got it
Preferences

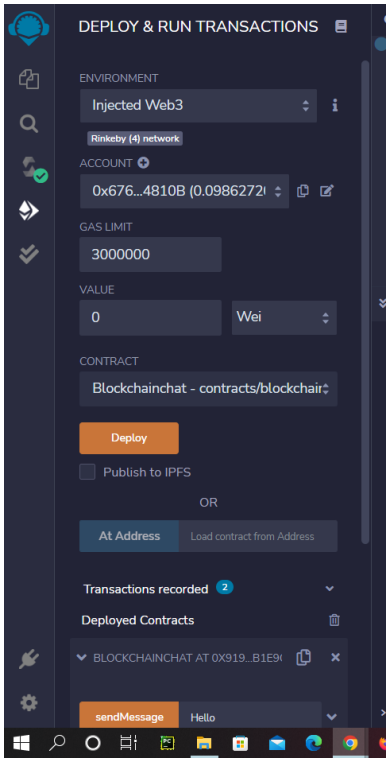

view on etherscan
[block:10719489 txIndex:8] from: 0x676...4810B to: Blockchainchat.sendMessage(string) 0x919...B1E9C value: 0 wei data: 0x469...00000 logs: 0
hash: 0xf24...9823f
Debug
status
true Transaction mined and execution succeed
transaction hash
0x2edf97e6df3f32592369a25a0dd1329fcbf3f8a518f31b85ab1a7d0600f91a4c
from
0x676A6E64813CaA01d19202ebF53F1b23a1348108
to
Blockchainchat.sendMessage(string) 0x919aED928fd751a6636eA6B7694910aD939B1E9C
gas
111334 gas
transaction cost
111334 gas
input
0x469...00000
decoded input
{
"string _content": "Hello"
}

Chapter 3: Graphical User Interface





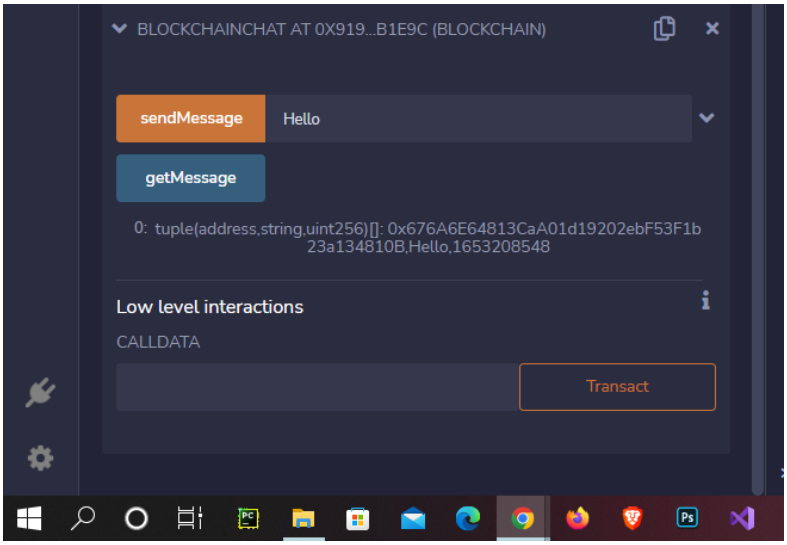


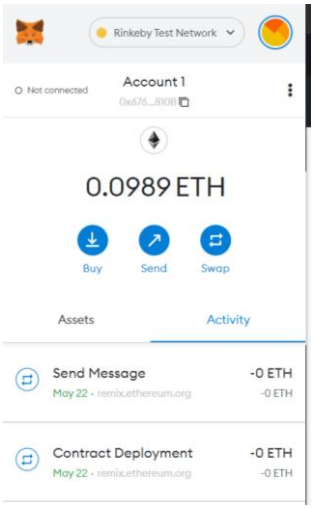


1 File Explorer **2 Compiler** **3 Deploy And Run**

- The file explorer, as seen in the picture above. This is where we're going to write our smart contract and its test file. It should be pre-filled with some files already. Feel free to go through them!
- The Solidity compiler. This is where we will compile our smart contract, once it is ready.

- The Deploy and Run Transactions, which is, you guessed it, where we will send our smart contract to the Blockchain.





(METAMUSK EXTENSION)

Chapter 4: Conclusion and Future Scope

4.1 Conclusion:

- Smart contracts can be used to make an automated system where users can chat with each other by using Ethereum Blockchain.

4.2 Future Scope:

- In future scope we will integrate this smart contract with a web3 platform and make a frontend for web page-based chat application.
- Where user can see with which blockchain they are connected and they can interact with each other.
- To make a web-based application we will use react native For the development of web-based application.

4.3 References:

- [1] A blockchain-based secure communication framework for community interaction by Rahul Sharma, Mohammad Wazid, Prosanta Gope.
- [2] https://www.investopedia.com/terms/b/blockchain.asp
- [3] Secure peer-to-peer Communication based on blockchain by Kahina Khacef, Guy Pujolle.
- [4] Secure Communications Using Blockchain Technology By Peter Menegay, Jason Salyers, Griffin College.
- [5] Securing Messaging platform Using Blockchain Technology by Suman Kumar Das