THE DEVELOPMENT OF BLOCKCHAIN BASED PAYMENT GATEWAY FOR SECURING TRANSACTION

Manju Biswas
Assistant Professor
Department of Computer Science & Engineering
Kalyani Govt Engineering College, Kalyani, India

Abstract : A gateway that accepts and processes payments using blockchain (crypto, tokenized assets or stablecoind), combining on-chain smart contract with off-chain services. In centralized system these gateways has led to a lack of transparency, long processing time, high fess, central point of failure and fraud and significant risk of fraud. This work is about a blockchain based payment gateway. The objective of this dissertation is to address the problem of implementation of a fast, secure, trust less payment gateway system which can handle the problems due to single third-party dependency in traditional systems. The work includes a thorough study of cutting-edge technologies like Blockchain, web3 applications, decentralised applications, smart contract, multi-currency supported traditional payment gateway systems. The payment gateway system is implemented using the Sophelia network, Ethereum virtual machine, remix software. The outcomes demonstrate a web3 application that can solve the given problem. The report concludes with possible future extensions and improvements of the proposed work.

IndexTerms **- Blockchain, Smart Contract, Ethereum, Payment gateway**

1. INTRODUCTION

Online transactions have gained a huge market for payment and hence it becomes important to look into its cons as well as flaws. Most of the current payment gateway systems don't include the transactions where multiple currencies are involved. Besides, the current payment gateway systems include various third party systems, which is time consuming because it happens that the transaction has to go through multiple third parties which also creates the risk of the transaction getting failed. The very next important factor is security, where the current system is not fulfilling the expectations of the customer. This is because there are various cases breaking the security of the transaction where the attacker tampers into the network and leading to money loss and also the faith of the customers is lost. Then there comes additional transactional charges which is point to be looked upon from the customer's point of view which can be reduced using blockchain [4]. Again, improving financial management is a need which can be accomplished using blockchain where we look upon to create a decentralized application. Blockchain makes it easy to maintain the transactions on a whole, and also fastens the transactions where the current system fails i.e. it is much slower compared to blockchain. So we plan on introducing blockchain for online payment. Each node i.e. customer side transaction would be recorded in the blockchain. The current public blockchains available are Ethereum[13][14], Bitcoin [1][5][7] etc. Though Ethereum[15] provides a platform to create our own smart contracts, since it is a public network the transaction details get visible to all rather than just showing it to a sub-network where only respective people get to view their transactions.

The concept of payment gateway has evolved over the years, with various techniques being developed to tackle challenges in this area. Multi-currency payment gateway [2][3] enables businesses to accept international payments from global customers. It acts as a bridge between a business's website and various international payment methods. This system automatically converts currencies, often using real-time exchange rates, streamlining the payment process and minimizing conversion fees. By supporting transactions in multiple currencies, businesses can expand their market reach, improve customer satisfaction, and efficiently manage international sales. But in most of the cases, such gateway systems work on a trusted centralized third party. This problem can be removed using Blockchain technology. The blockchain is a distributed database[6] of records of all transactions or digital events that have been executed and shared among participating parties. Each transaction is verified by the majority of participants of the system. It contains every single record of each transaction. Bitcoin is the most popular cryptocurrency, an example of the blockchain. Blockchain Technology first came to light when a person or group of individuals named 'Satoshi Nakamoto' published a white paper on "Bit-Coin: A peer-to-peer electronic cash system" in 2008. Blockchain Technology Records Transaction in Digital Ledger which is distributed over the Network thus making it incorruptible. Anything of value like Land Assets, Cars, etc. can be recorded on Blockchain as a Transaction. A "smart contract" is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. Smart contracts are a type of Ethereum account. This means they have a balance and can be the target of transactions. However, they're not controlled by a user, instead they are deployed to the network and run as programmed. User accounts can then interact with a smart contract by submitting transactions that execute a function defined on the smart contract. Smart contracts can define rules, like a regular contract, and automatically enforce them via the code. Smart contracts cannot be deleted by default, and interactions with them are irreversible.

The rest of the thesis is organized as follows: • Chapter 2 discusses a short summary of literature reviews. Chapter 3 presents an elaborated description of the proposed work. • Chapter 4 concludes the report and suggests future directions.

The rest of this paper is summarized as follows. Literature review on Blockchain based Payment Gateway is provided in Section 2. The existing relevant work is discussed in Section 3. The suggested framework is fully detailed in section 4. The BI2V framework's performance is analyzed in Section 5 using a variety of performance measures. The conclusion and future work are presented in section6.

2. LITERATURE REVIEW

The establishment of large hospitals where hundreds to thousands of patients are treated , it has created a serious problems of Satoshi Nakamoto [1] present Bitcoin on the behalf of Blockchain Technology, invented new technologies in digital payments by replacing the middle man or any third party. Here he present decentralized process for peer-to-peer transactions. This system reducing the transaction fees and it is very much transparency system. This Bitcoin technology scaling up the process of blockchain-based payment gateways through it's secure system, immutability, integrity of large transaction

Buterin [8] expand blockchain system through Ethereum, here he presenting smart contracts to make the payment system automatic and secure payment processes. This smart contract is an agreement, which facilitates real time, self-executing payments without human intervention. This system make trust among the user int the payment platforms.

Poon and Dryja [9] introduced the lightning network as a layer 2 solution to solve the Bitcoin's scalability issues. By scaling-up off-chain microtransactions, this invented technology improved the speed & cost effectiveness of blockchain-based payments. They making the technology more effective for real world applications, including payment gateways.

Hileman and Rauchs [10] investigate the adoption of cryptocurrencies in payment systems, noted the growing preference for blockchain-based solutions. Their report told that decentralize application decrease the settlement times, fraud risks, enhance the global payment accessibility, and also creating opportunities for blockchain powered payment gateways.

Das et al. [11] reviewed blockchain's integration with payment processors, highlighting enhanced user privacy, faster transaction speeds, and reduced operational costs. The study emphasized the transformative potential of blockchain in reshaping traditional payment gateways into decentralized platforms.

According to the study by Subramaniam[12], blockchain technology can provide a cheaper and efficient way of transferring funds, because it eliminates the need for expensive broker. As a result the transaction fees are reduced. The author also indicates, the blockchain technology can be used to create digital identities for the customers, so that it can help to increase financial inclusion and provide a secure way of storing personal information.

3. PROPOSED WORK

3.1: Problem Statement:

Now-a-days, humans are too much dependent on online transaction methods rather than traditional paper-based transactions because it provides 24×7 facilities and it can be accessed from anywhere and anytime. Besides, it has low maintenance. According to a survey by worldline reports, digital payments capture almost 65% of the global trades. But there are very few payment gateways like razor pay, stripe which supports multiple currencies. In most of the cases, the system involves multiple banks and the system is centralized. As a result, if the single central server fails, it is impossible to provide trust and security. So, we require a decentralised gateway for such cases. Blockchain can solve the issue.

Rather than using multiple currencies, we have worked on two types of currency. We have focused on such a system where the customer wants to pay the amount in ether but the merchant prefers to receive the amount in rupees. Besides, we have detected the possible frauds and their solutions.

3.2: Proposed Solution

In our proposed Solution, we have worked on a special case of online transactions where multiple currencies are involved. We have assumed that the customer can only pay using Cryptocurrency like Ethereum and the merchant has only a traditional monetary system. As a result, it requires an intermediate system to convert Ethereum into rupees. Rather than using a single system, there are some independent volunteers who can access both types of currencies. They pay the rupees amount to the merchant and receive the equivalent Ethereum amount. Interestingly, in our system any person can act as a customer or intermediate or both and the customer or the intermediate are totally unaware of their addresses. Thus, the system provides a trust less environment. A brief architecture is given in the figure.

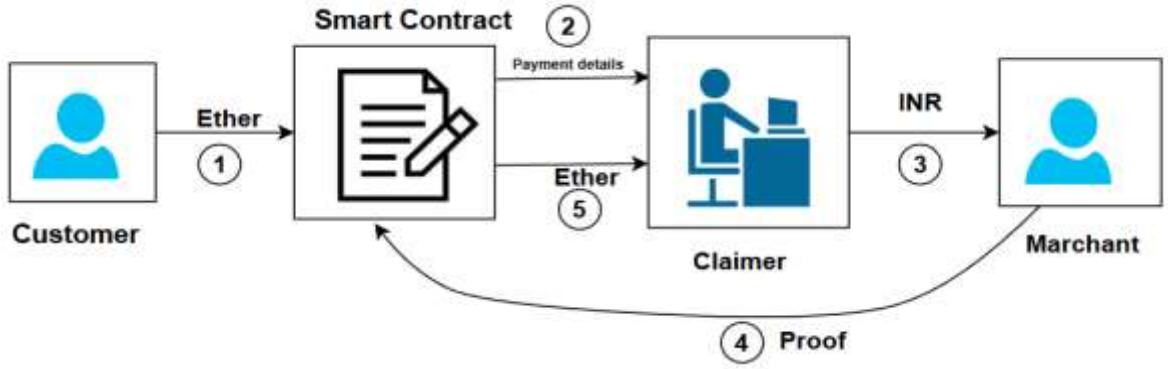

Fig. 1: Basic Architecture

3.2.1 Functionality of Customer

The customer connects at first his/her wallet using the user interface. Then he/she provides the merchant UPI id and the amount (In INR) to be transferred. Then, he pays the equivalent Ethereum in the smart contract. For a specific amount of time, a timer runs in the user interface. If the timer expires and no one shows interest to pay on behalf of him/her, the

Ethereum amount is refunded and the payment is a failure. Details breakdown are provided in the flowchart and user interfaces section.

3.2.2 Functionality of Smart Contract

Smart contract plays a pivotal role in the system. It stores the data about the merchant and equivalent Ethereum from the customer. It also controls the flow of the Ethereum amount in a proper way. It also checks any fraudulent activities and takes steps against such activities. Detailed breakdown is given in the smart contract section.

3.2.3 Functionality of intermediate

Intermediate acts as a connection between the customer and the merchant. They must have both types of currencies used in this case. He checks the database for unclaimed UPI id, amount of money to send to the receiver, and the equivalent Ethereum he will gain in return. As he claims one of the unclaimed transactions, he has to pay the amount in rupees to the merchant. He has to submit a specific proof to the smart contract in a certain amount of time . Smart contract checks the correctness of the proof and according to it, smart contract decides whether the Ethereum should be refunded to the customer or should be paid to the intermediate as Detailed breakdown is described in the flowchart and user interfaces section.

3.2.4 Functionality of the Marchant

Here, the Marchant has very low significance. He just receives the money from the intermediate.

3.3: Flowchart and Sequence diagrams

The algorithm of this complete process from transaction from customer to smart contract and from the intermediate to the merchant is represented in the form of a flowchart given as follows:

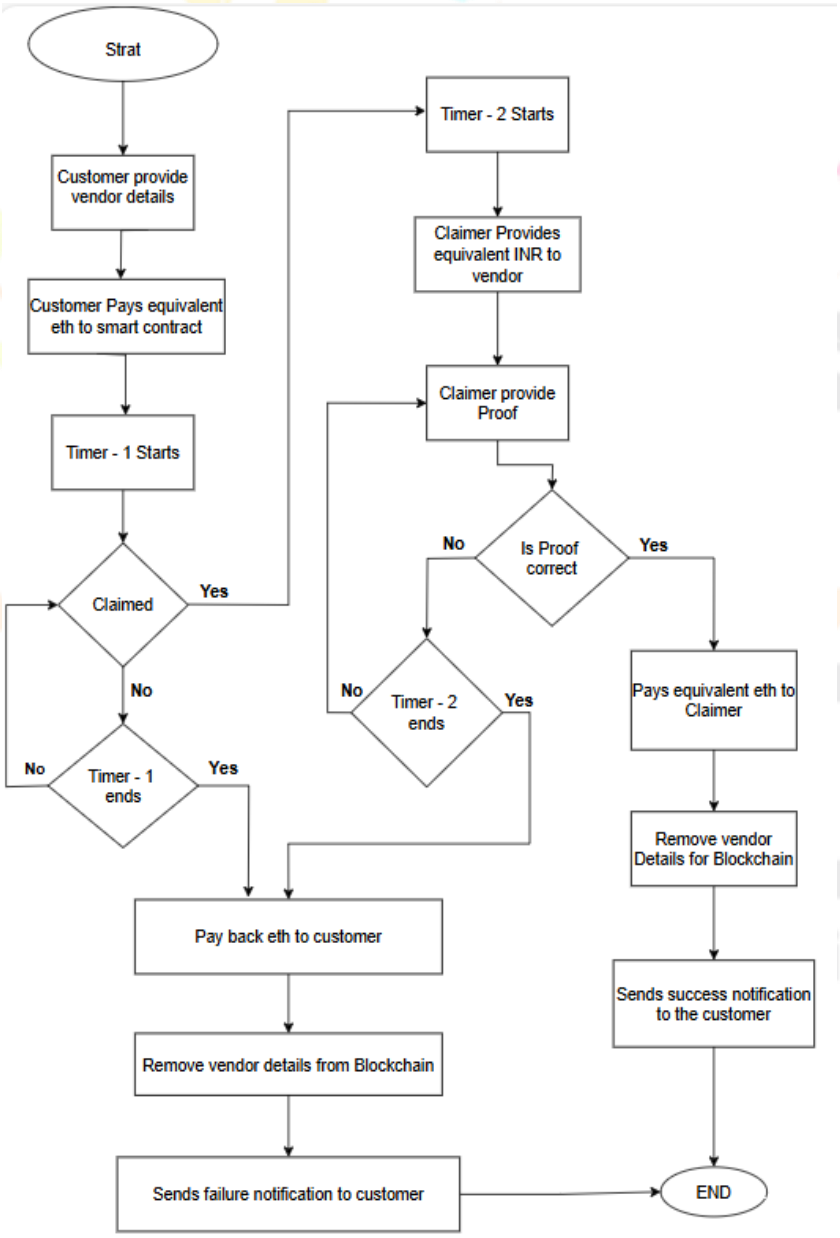

Fig. 2: Flowchart of the proposed model

Sequence diagram for a successful payment is given in Fig. 3 and unsuccessful payment is given in Fig. 4

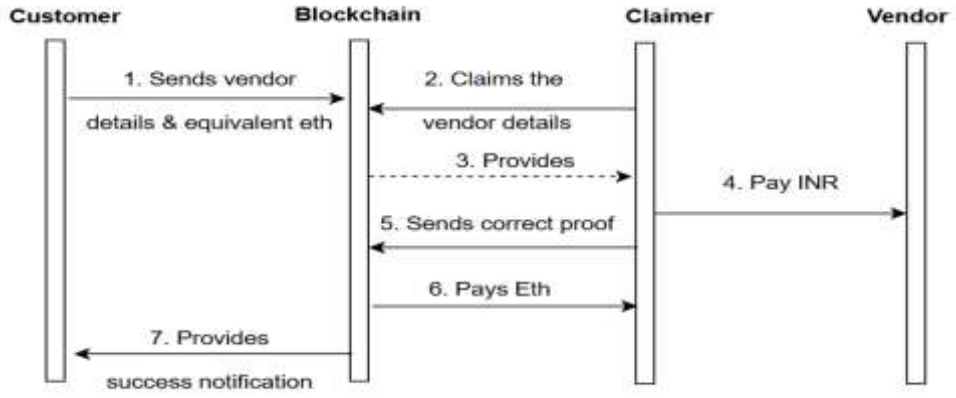

Fig.3: Sequence UML diagram for a successful payment

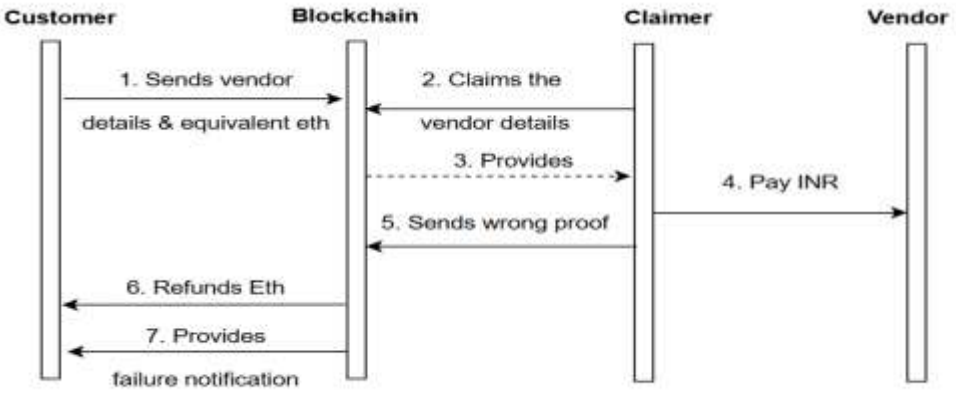

Fig.4: Sequence UML diagram for a successful payment

4 IMPLEMENTATION OF SMART CONTRACT

4.1 Data structures

stored_payment_request:
We have used a user-defined private data structure named as " stored_payment_request" which is used to store the details of a payment request made by the customer, whose components are:

```
struct stored_payment_request {
      address customer_address;      //provided address
      string upi_id;                        //provides the entered upi_id
      uint amount_in_rupee;                 //provides amount to pay in INR
      address claimer;                      //address of the claimer, 0 if not claimed
      uint submitted_eth;            //eth provided by the customer
      uint request_time;             //timestamp when it is requested
   }
```

transactionMapping:
We have used a hashmap for providing each address an unique transaction_id:

```
mapping(uint => stored_payment_request) private transactionMapping;
```

viewer_side:

We have used another structure which hides some information and provides only those information to the claimer.

```
struct viewer_side {
      uint transaction_id;
      uint amount_in_rupee;
      string upi_id;
      uint submitted_eth;
   }
```

4.2 Functions

Request_payment:
Using this function, the customer can request for a payment. This function also stores information to the mapping, with proper error management.

```
function request_payment(string memory _upi_id, uint _amount_in_rupee) public payable  {
```

```
    require(msg.value > 0, "Payment must be greater than 0");
    transactionMapping[transaction_id] = stored_payment_request({
      customer_address: msg.sender,
      upi_id: _upi_id,
      amount_in_rupee: _amount_in_rupee,
      claimer: address(0),
      submitted_eth: msg.value,
      request_time : block.timestamp
    });
    transaction_id++;
  }
```

show_pendings:
This function is used by viewer to view the unclaimed recent payment requests. Since, after 1 minute, the payment request will be refunded, we have limited the time limit as 1 minute.

```
function show_pendings() public view returns (viewer_side[] memory result) {
    uint count = 0;
    for (uint i = transaction_id-1; i>0 && count < 20; i--) {
      stored_payment_request storage node = transactionMapping[i];
      if(node.request_time < block.timestamp-1 minutes)
          break;
      if (node.claimer == address(0)) {
        result[count] = viewer_side({
          transaction_id: i,
          amount_in_rupee: node.amount_in_rupee,
          upi_id: node.upi_id,
          submitted_eth: node.submitted_eth
        });
        count++;
      }
    }

    return result;
  }
```

claim_the_transaction:
This function is used by the claimer to claim a request and it returns the success status.

```
function claim_the_transaction(uint _transaction_id) public returns (bool) {
    require(_transaction_id != 0 && _transaction_id < transaction_id,"Invalid transaction id");
    stored_payment_request storage node = transactionMapping[_transaction_id];
    if (node.claimer != address(0)) {
      return false;
    }
    node.claimer = msg.sender;
    return true;
  }
```

refund_the_customer:
The function is used to refund the ether amount for failed transaction. It can be called only after timer-1 (1 minute) for unclaimed transaction and for claimed, it only be called after timer-2 expires (3 minutes). It also delete details of the transaction after refunding.

```
function refund_the_customer(uint _transaction_id) public returns (bool) {
    require(_transaction_id != 0 && _transaction_id < transaction_id,"Invalid transaction id");
    stored_payment_request storage node = transactionMapping[_transaction_id];
    require(node.request_time<block.timestamp-1 minutes,
" Transaction cannot be refunded ");
    require(node.claimer==address(0) || node.request_time < block.timestamp-4 minutes,
                    "Transaction cannot be refunded");
    address payable customer = payable(node.customer_address);
    uint refundAmount = node.submitted_eth;
    if (refundAmount == 0 || customer == address(0)) {
      return false;
    }
    (bool sent, ) = customer.call{value: refundAmount}("");
    if (!sent) {
      return false;
    }
```

```
    delete transactionMapping[_transaction_id];
    return true;
  }
```

provide_proof:
This function is used to check the provided proof of UPI transaction (performed by claimer to the merchant). It checks the proof and if proof is correct, then the amount is transferred to the claimer using another one function.

```
function provide_proof(uint _transaction_id, bytes32  proof) public returns (bool){
    require(_transaction_id != 0 && _transaction_id < transaction_id,"Invalid transaction id");
    require(proof.length > 0,"msg cannot be empty");
    stored_payment_request storage node = transactionMapping[_transaction_id];
    require(msg.sender == node.claimer,"Unauthorized access ");
    if(sha256(abi.encodePacked(node.amount_in_rupee,node.upi_id))!=proof)
      return false;
    return pay_to_claimer(_transaction_id);
  }
```

pay_to_claimer:
This function is called automatically for successful payment. This is used to provide the ether amount to the claimer from the contract. This function also cleans the record.

```
 function pay_to_claimer(uint _transaction_id) private returns (bool) {
    stored_payment_request storage node = transactionMapping[_transaction_id];
    address payable claimer = payable(node.claimer);
    uint amount = node.submitted_eth;
    if (claimer == address(0) || amount == 0) {
      return false;
    }
    (bool success, ) = claimer.call{value: amount}("");
    if (!success) {
      return false;
    }
    delete transactionMapping[_transaction_id];
    return true;
  }
```

5 EXPERIMENT RESULT

The contract is successfully deployed in Sepolia test network.

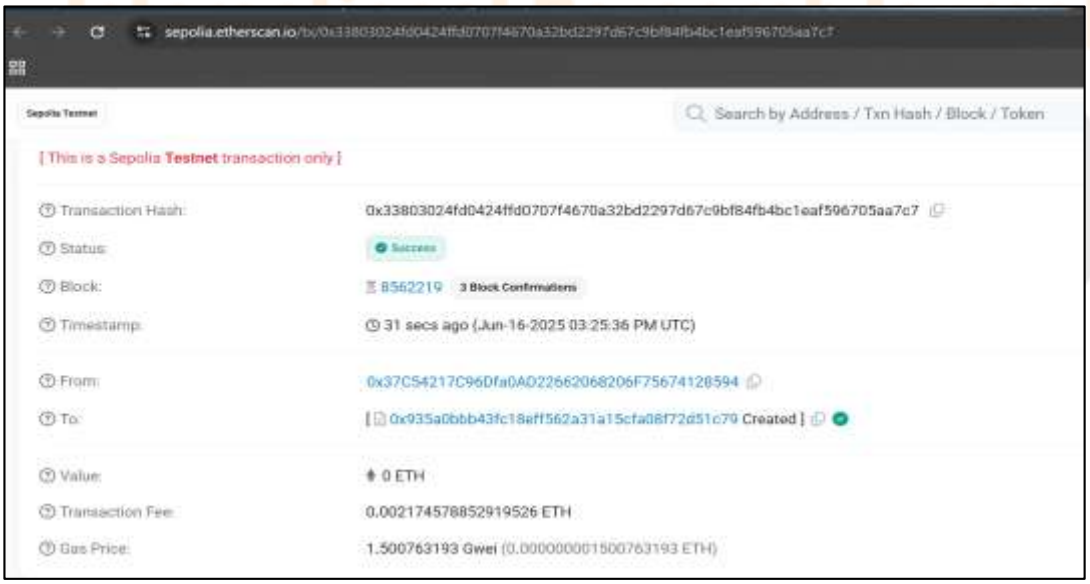

Fig.5: Successful Deployment of the Contract

Creating a payment Request:

Here, an account with address "0x17F6AD8Ef982297579C203069C1DbfFE4348c372" is creating a payment request and providing 10 Eth for the transaction. After successful transaction, the balance of the contract is 10 Eth (a little bit less for gas fees).

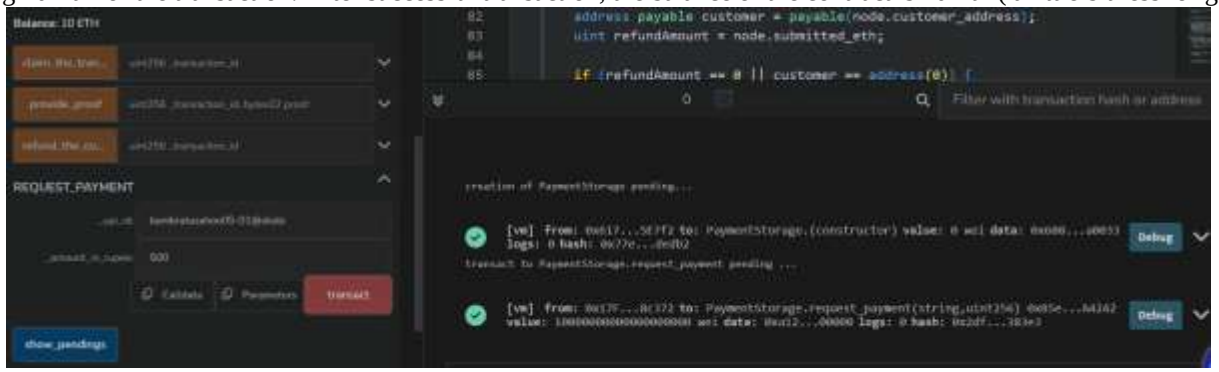

Fig. 6: Successful payment request

Showing the pending request and claiming:

An account "0x5c6B0f7Bf3E7ce046039Bd8FABdfD3f9F5021678" views the pendings and claim transaction_id – 1

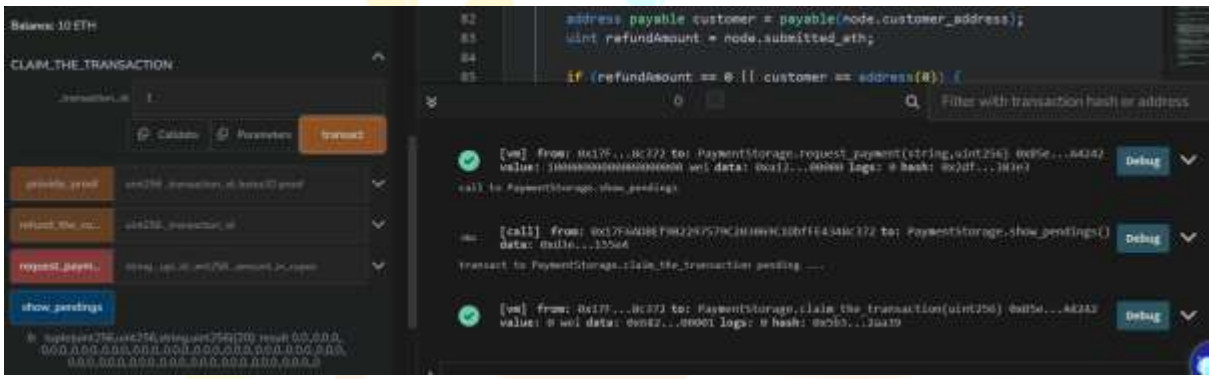

Fig 7: Successful claiming

Claimer reaches its time-limit and refund to customer:

Here, the the function refund_to_customer is called twice. For first time it is called below time limit and reverted. Then, after time limit it is called and amount is paid to the customer.

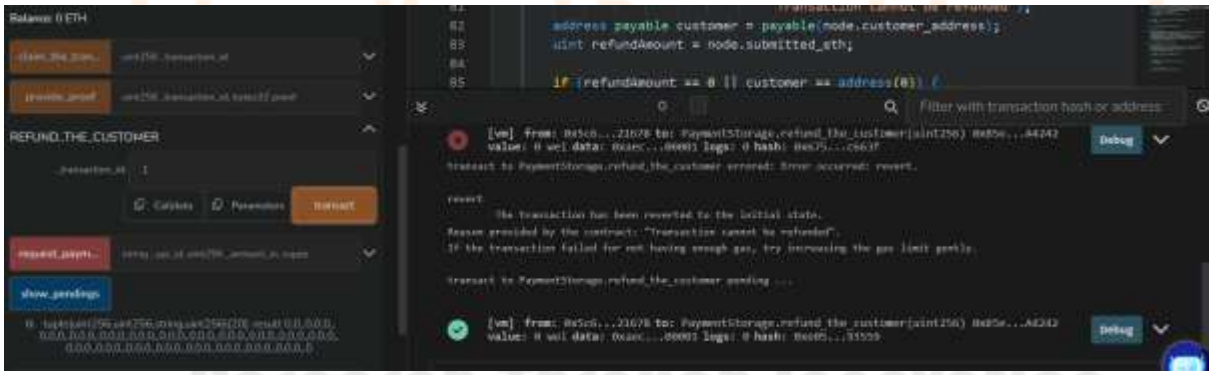

Fig. 8: failed and then successful refund of ether to the customer after failed transaction

Fig. 9A: Account Details before transaction

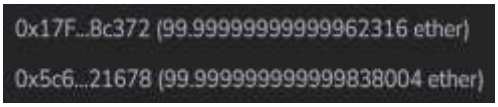

Fig. 9B: Account Details after transaction

Providing correct proof and transferring ether equivalent:

Let, here the customer is "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c" and the claimer is "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C". Other data are same as the previous one. After successful payment, claimer provides a unique value for the transaction (here "0xb522bfa3b63bb9a87c0005f25c6ffdbc67642cce07a2cfb73995431d55dee679 ") and collects the ether amount in its wallet. The little bit change from original amount is due to gas cost and others.

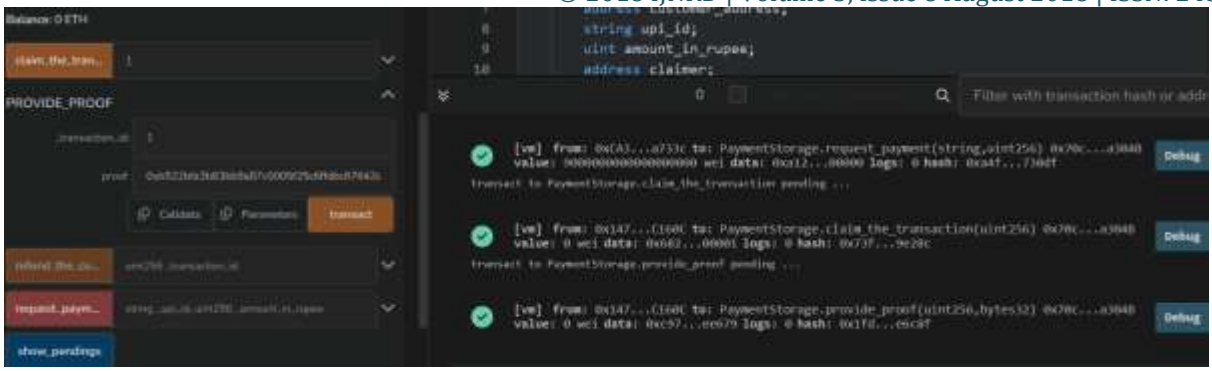

Fig. 10: Successful transaction of ether to the claimer after successful payment

6. CONCLUSION AND FUTURE WORK

In this paper, we have addressed the problem of brief restatement of the problem tackled. A solution was proposed using methodology/approach used, which was implemented and evaluated. The results indicate that the proposed approach insert outcome, e.g., improves performance, reduces latency, increases accuracy, etc.. Comparative analysis with existing solutions shows that highlight any advantage or insight. Throughout this report, the design, implementation, and evaluation stages have been discussed in detail. The work demonstrates the feasibility of state main achievement and offers a strong basis for further exploration.

The possible future directions of this work are exploring advanced algorithms to improve efficiency and scalability, integration with real-world systems and platforms, expanding the dataset or scope of application for generalization and performing longitudinal studies for performance evaluation.

REFERENCES

[1] Nakamoto S., "Bitcoin: A peer-to-peer electronic cash system," Bitcoin, 2008.
[2] Lowry, P.B., Wells, T.M., Moody, G.D., Humphreys, S., Kettles, D.: Online payment gateways used to facilitate e-commerce transactions and improve risk management. Communications of the Association for Information Systems (CAIS) 17(6), 1–48 (2006)
[3] Krebs, B.: Target: 40 million credit cards may be involved in breach. KrebsOnSecurity (2014). https://krebsonsecurity.com/ 2014/12/target-40-million-credit-cards-may-be-involved-inbreach/
[4] Yli-Huumo, J., Ko, D., Choi, S., Park, S., Smolander, K.: Where is current research on blockchain technology?-a systematic review. PLoS ONE 11(10), 0163477 (2016)
[5] Narayanan, A., Miller, A., Clark, J., Kroll, J.A., Felten, E.W.: Bitcoin and cryptocurrency technologies: A comprehensive introduction. Princeton University Press (2016). Chap. Chapter 12
[6] Liu, J.K., Liang, K., Susilo, W., Liu, J., Xiang, Y.: Two-factor data security protection mechanism for cloud storage system. IEEE Trans. Comput. 65(6), 1992–2004 (2015)
[7] Narayanan, A., Bonneau, J., Felten, E., Miller, A., Goldfeder, S.: Bitcoin and cryptocurrency technologies: A comprehensive introduction (2016)
[8] Buterin V., "A next-generation smart contract and decentralized application platform," Ethereum Whitepaper, 2013.
[9] Poon J. and Dryja T., "The bitcoin lightning network: Scalable off-chain instant payments," https://lightning.network/, 2016.
[10]Hileman G. and Rauchs M., "Global cryptocurrency benchmarking study," Cambridge Centre for Alternative Finance, 2017
[11] A. Das et al., "Review of blockchain integration with payment processors: Opportunities and challenges," International Journal of Advanced Computer Science and Applications, vol. 9, no. 10, pp. 234–240, 2018.
[12] Subramanian, H.: Decentralized blockchain-based electronic marketplaces. Commun. ACM61(1), 78–84 (2017)
[13] Antonopoulos, A.M., Wood, G.: Mastering ethereum: building smart contracts and dapps (2018)
[14] Dannen, C.: Introducing ethereum and solidity1(2017)
[15] Khoury, D., Kfoury, E.F., Kassem, A., Harb, H.: Decentralized voting platform based on ethereum blockchain. In: 2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET), pp. 1–6 (2018). IEEE

IJNRD Research Through Innovation