Blockchain for FinTech Applications : Cross-Border Transactions

Deepanshu Agrawal (B19EE025), Anuman (B19CSE013)

December 25, 2022

Introduction

In the FinTech industry, blockchain can be used for making more secure and efficient payments systems. Cross-border payments has been a long lasting issue due to its inconveniences such as cost and time inefficiencies which is caused due to involvement of too many intermediary parties. Our goal is to create a Blockchain application that can be used to remove the intermediaries using peer-to-peer (P2P) Blockchain concept and create a time and cost efficient system. The proposed blockchain will also solve the problem of government regulations that are faced by current blockchain systems.

Existing Technologies and Their Flaws

- **Stellar**: Stellar is a decentralized protocol on open-source code to transfer digital currency to fiat money domestically and across borders. The Stellar blockchain's cryptocurrency is called the lumen, a token that trades under the symbol XLM. XLM is similar to XRP in the use case, token origination, similar points of centralization, and the fact that a central entity (Stellar Foundation) arguably controls the entire network. The network contains only 38 full validators, which are heavily centralized around SDF nodes creating network-wide vulnerabilities. XLM's regulatory future is also of concern. In the Stellar blockchain, the SHA-256 hashing algorithm is used to generate hashes for transactions and other data that is stored on the blockchain.
- **Ripple**: Ripple is a blockchain-based digital payment network and protocol with its own cryptocurrency, XRP. Rather than using blockchain mining, Ripple uses a consensus mechanism, via a group of bank-owned servers, to confirm transactions. The openness of Ripple has vulnerabilities. Although the core of the network remains highly liquid, the structure also allows for attacks on certain nodes within the network to cripple some users' access to funds. In fact, some 50,000 wallets may be immediately at risk if such an attack were to occur. In the Ripple blockchain, the SHA-256 hashing algorithm is used to generate hashes for transactions and other data that is stored on the blockchain. The hashes are used to verify the integrity of the data and ensure that it has not been tampered with. If any changes are made to the data, the resulting hash will be different, which will allow the Ripple network to detect the tampering and reject the transaction.

Proposed Blockchain and it's advantages

System Framework : We propose an asymmetric Blockchain system using Ethereum smart contracts to build a fairer and more intelligent cross-border transaction system. Smart Contracts will be used to create a Payment system where users can deposit/withdraw money from the bank and also do cross-border transactions among themselves. We refer these users as light nodes in our system. To solve the problem of government regulations, Smart Contracts will be used to create a separate system to be used by regulators to check any discrepancy in the system. We refer these regulators as full nodes in our system. Through regulators, real-time government regulation is considered to control suspicious transactions. These regulators will have power to freeze an account based on suspicious transactions. To execute foreign exchange among users, we need a currency exchange market where currency sent by the sender will be converted to the recipient's currency. The currency exchange market is regarded as the exchange rate information provider. When the light nodes and full nodes need the rate information, they can turn to the market for data acquisition. We will be using chainlink for this purpose. Chainlink is a decentralized blockchain oracle network built on Ethereum. The network is intended to be used to facilitate the transfer of tamper-proof data from off-chain sources to on-chain smart contracts. Thus, an asymmetric system of light nodes and full nodes is created using smart contracts on Ethereum network.

Advantages over existing technologies : The proposed Blockchain payment system aims at solving major issues of existing technologies like cost-efficiency, time-efficiency, third-party interventions and government regulatory issues as well as maintain the security and usability of the cross-border payment system.

- Cost-efficiency : Transaction cost in Ethereum Network is in the form of gas. Gas prices are denoted in gwei, which itself is a denomination of ETH - each gwei is equal to 0.000000001 ETH. Testing of our algorithm is done ahead on Goerli Network.
- Time-efficiency : Normally, cross-border transactions take about two to three days without use of Blockchain technology. With the use of Blockchain, this has been reduced to mere seconds. Our Blockchain achieves the same time efficiency which has been proved by results. The time-efficiency though largely vary depending on the traffic on the network. A faster payment can be executed by paying little more gas than usual in such cases.
- Security : We will be using Keccak256 hashing algorithm which is widely accepted in Ethereum Blockchain due to it's high security. Explaination of the same is given in Algorithms section.
- Third-party interventions : Blockchain establishes direct contact between the sender and the receiver. The receiver has direct access to the payment. There are no delays, unnecessary fees, or remittances involved.
- Government Regulation issues : Ripple, Stellar and other existing technologies are facing severe government regulatory issues. Ripple is currently being sued by SEC over XRP. XLM's regulatory future is also of concern. The proposed blockchain solves this issue by letting government have some involvement with the payment system to maintain regulations as discussed.

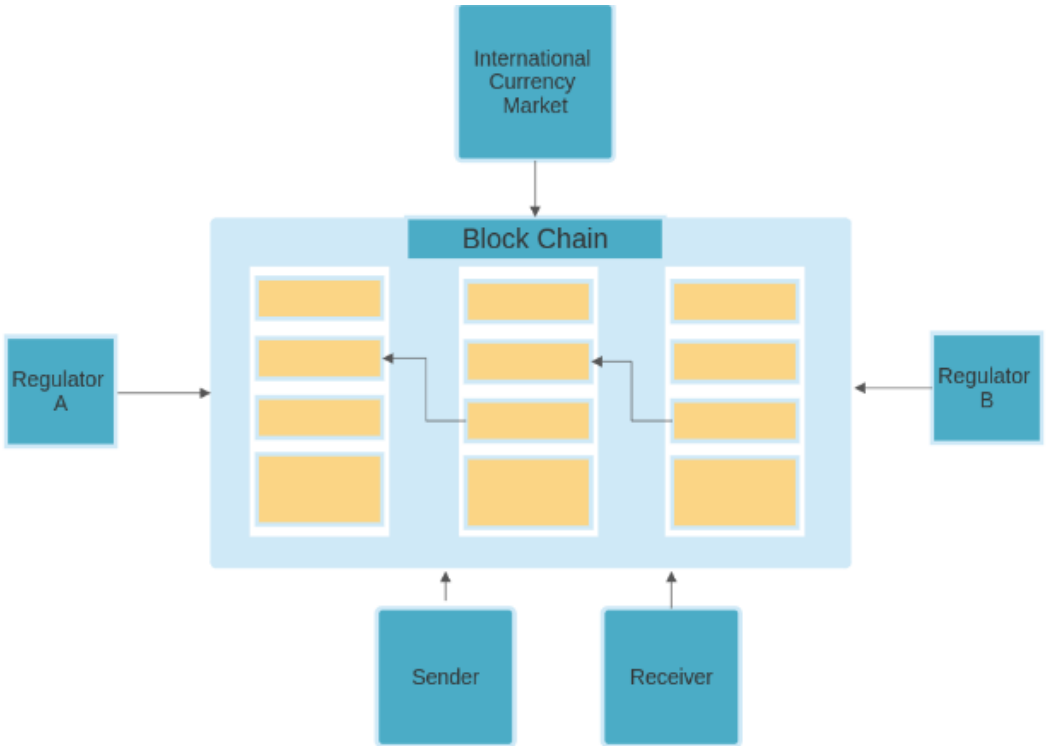

Figure 1: System Framework

Algorithms

Hashing Algorithm : Keccak256 and SHA-256 are both cryptographic hash functions that are used to secure data and protect against unauthorized access. Both algorithms are widely used and are considered to be very secure, but there are some key differences between them that may make Keccak256 a better choice for certain applications, particularly in the blockchain space.

Here are some potential advantages of using Keccak256 over SHA-256 in the blockchain:

- Resistance to attacks: Keccak256 has been designed to resist certain types of attacks that are effective against SHA-256 and other SHA-2 hash functions. This makes it a more secure choice for applications that require high levels of security, such as the blockchain.
- Flexibility: Keccak256 can produce hashes of various sizes, including 224, 256, 384, and 512 bits, while SHA-256 produces hashes of fixed size, with a length of 256 bits. This flexibility may make Keccak256 a better choice for applications that require hashes of specific sizes.

- Efficiency: Keccak256 is generally more efficient than SHA-256 in terms of the computational resources it requires, which may make it a more practical choice for applications that need to process large amounts of data, such as the blockchain. Keccak256 is based on a sponge construction, which involves using a variable-sized input to absorb and squeeze data through a series of internal permutations. This allows Keccak256 to process data more efficiently than SHA-256, which is based on a fixed-block message-digest algorithm that processes data in blocks of 512 bits.

Opening Bank Account : Anyone can open an account in this system. User verification is not the part of the project. Therefore, currently we are using Metamask wallets to interact with the Ethereum Blockchain. To open an account, Ethereum wallet address is necessary. Each wallet address can create only one account. Here is the applied algorithm. The algorithm ensures that there is only one account associated with an address and sets the password and customer name for that account.

Algorithm 1 Opening new bank account

Require: Address, BankName, CustomerName, Password
  if address.UserExists = false **then**
    address.UserExists ← true
    address.Bank_Name ← BankName
    address.Customer_Name ← CustomerName
    address.Pwd ← Password
    Print Congratulations message as account created successfully.
  else
    Print Error as account already exist
  end if

Currency Exchange Market : We need live, accurate and reliable updates on conversion rates between different currencies. For this purpose, an currency exchange smart contract is created which uses chainlink to extract conversion rates between different currencies. Chainlink is a decentralized blockchain oracle network built on Ethereum. The network is intended to be used to facilitate the transfer of tamper-proof data from off-chain sources to on-chain smart contracts. Chainlike provides integrated pre-built, time-tested oracle solutions that already secure tens of billions in smart contract value for market-leading decentralized applications. To find a conversion rate using chainlink, we need to know the price feed contract address for the particular currencies which are to be exchanged. Smart contracts do not support decimal values, therefore, values given by chain link are multiplied by 10^18 to take decimal values into consideration.. To accomodate these zeros, we divide our answer by same value.

Algorithm 2 Currency Exchange Contract

Require: SenderCurrency, ReceiverCurrency, Amount
  address ← SenderCurrency to ReceiverCurrency Price Feed address
  PriceFeed ← AggregatorV3Interface(address)
  ConversionFactor ← priceFeed.latestRoundData()
  ConvertedAmount ← (ConversationFactor * Amount)/1000000000000000000
  Return ConvertedAmount

Foreign Exchange : The most important part of the project is the international transaction or the foreign exchange system. The users can send and receive money after applying the foreign exchange tax on the amount being sent. The currency of the sender is converted to the receiver's currency using algorithm 2. After checking the base conditions like sender and receiver's account existence and correct value of amount to transfer, forex tax is calculated on the amount to be sent. After calculating the forex tax and transfer tax, transfer amount and transfer tax is debited from the sender's balance and converted amount is added to the receiver's balance and the transaction is successful.

Algorithm 3 Foreign Exchange

Require: SenderBankName, ReceiverBankName, ReceiverAddress, TransferAmount, Password

```
if SenderBankName = true and Password = true then

    if SenderBalance > TransferAmount then
        Continue
    else
        Insufficient Balance
    end if
    if SenderExists = true then
        Continue
    else
        Account is not created
    end if
    if RecieverExists = true then
        Continue
    else
        Receiver's Account does not exist
    end if
    if TransferAmount > 0 then
        Continue
    else
        Enter non-zero value for sending
    end if
    forexrate ← 5
    convertedrate ← converter(SenderBankName, ReceiverBankName, TransferAmount)  ▷ /*Using Algorithm 2*/
    transfertax ← (TransferAmount*forexrate)/100
    forextax ← forextax + converter(SenderBankName, ReceiverBankName, transfertax)
    SenderBalance ← SenderBalance - transfertax - TransferAmount
    ReceiverBalance ← ReceiverBalance + convertedrate
    Print successful transaction
else
    Invalid Credentials
end if
```

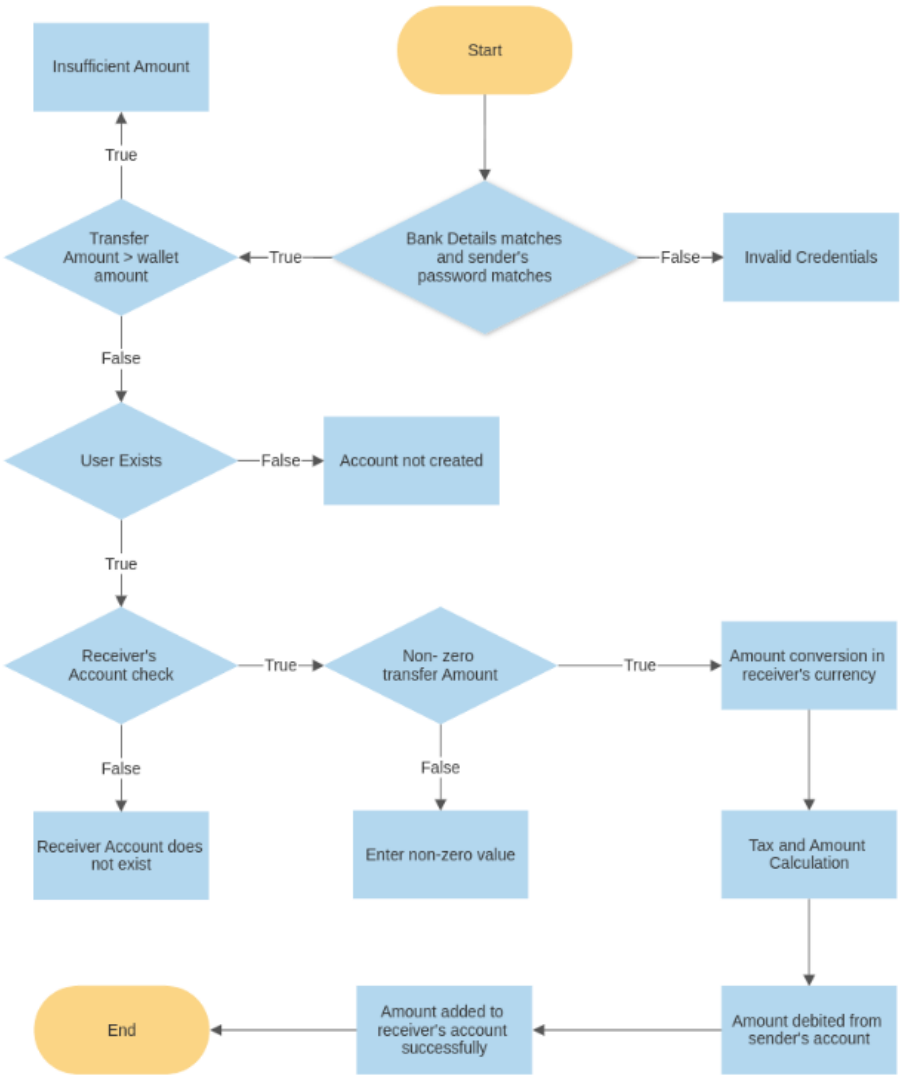

Figure 2: Code Flow

Result and Analysis

Smart contracts for currency exchange market and foreign exchange have been written in solidity language. To test the working of our codes, we are using Metamask wallets and using Goerli Test Network. Remix IDE, is being used as a GUI for developing smart contracts. First, the smart contract must be deployed using Injected Provider - Metamask as environment. In the Metamask extension, user must log-in to his Metamask account. First a user needs to open account by giving bank name, customer name and newly created password for this account. After the account creation, money can be deposited and withdrawn to/from this account from/to Metamask wallet. The deposited money will be entered in Wei which is the Ethereum currency. The money will be automatically converted to the bank's currency. Similarly, while withdrawing, currency stored in bank will be converted to Wei and transferred to Metamask wallet. After depositing some money, a user can send this deposited money to any other user by knowing his account address and receiver's bank name. The sender's currency will be converted to receiver's currency and will be sent after applying the forex tax. Currently, there are three banks added in the system - "Bank of India", "Bank of Japan" and "Bank of America" for testing purposes. The time and cost efficiency of the system is calculated and compared to existing technologies ahead with the results also showing the accurate working of the system.

Currency Exchange Market : We deployed the price converter contract and called for the price conversion factor for Japanese Yen to US Dollar. Gas had to be paid to deploy the contract but no gas had to be paid to see the price conversion rate as anyone can access the rates freely in this system. The time taken by call to be executed was less than 1 second so it is time-efficient as well. Since solidity doesn't support decimal values, therefore, the conversion factor is multiplied by 10^18 to accommodate decimal values. The result of the call for Yen to Dollar conversion was 7532105000000000 which is equal to 0.007532105 which was the correct value for 1 Japanese Yen in Dollars when

checked manually at the time of execution of call. Hence, our currency exchange market is working accurately.

Figure 3: Deploying the Currency Exchange Market Smart Contract

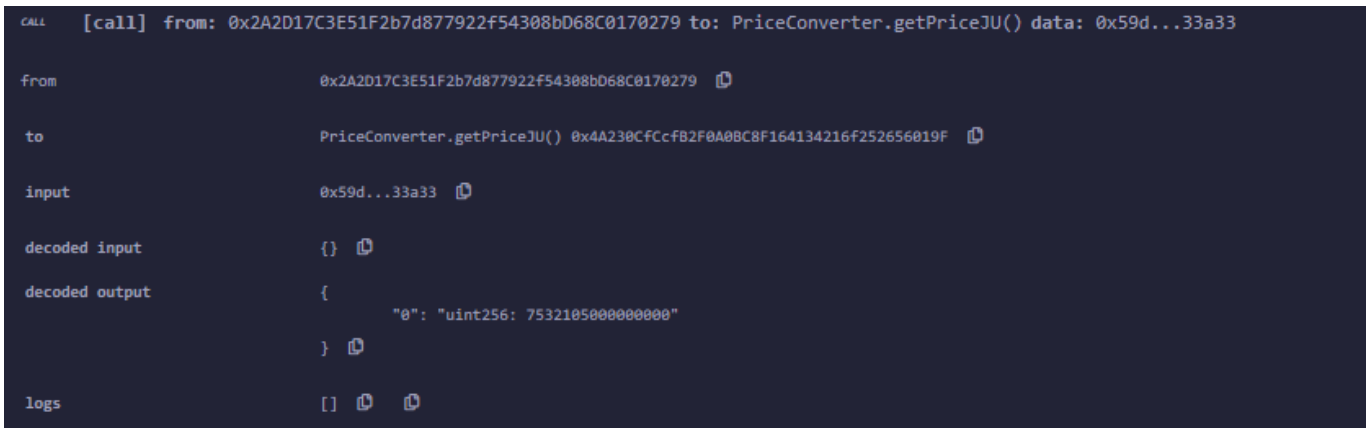

Figure 4: Price Conversion for Japanese Yen to US Dollars

Foreign Exchange : To check the efficiency of foreign exchange system, we opened two bank accounts in Bank of Japan and Bank of America with customer names A and B. Customer A deposited 9957 Yen in his account using GoerliEth from Metamask and sent 5000 Yen to customer B. A 5 percent forex tax was applied here to check the working of the code. Here are the results of deploying the contract, opening bank account, depositing money, transferring money, checking balance after transfer, and withdrawing the money back to Metamask wallet.

Figure 5: Deploying the Foreign Exchange Smart Contract

Figure 6: Opening a new Bank Account

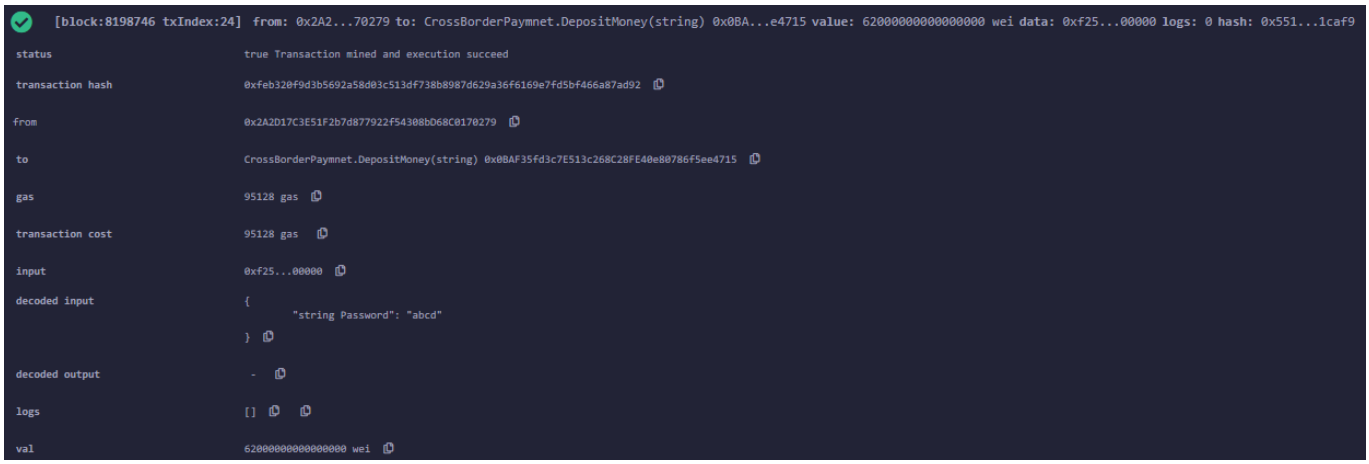

Figure 7: Depositing money into the account from Metamask wallet

Figure 8: Transfer of money from customer A to B

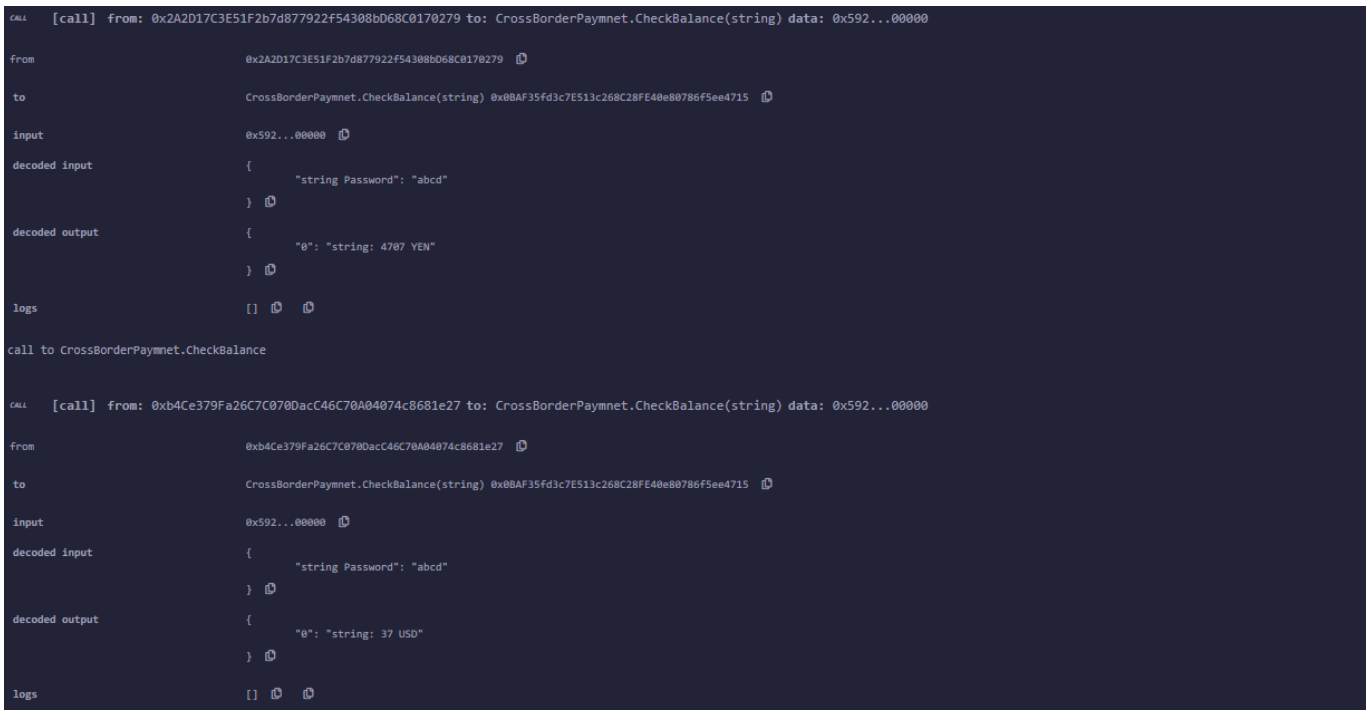

Figure 9: Checking the balance of accounts after the transfer of money

Figure 10: Withdrawing money from customer A's account back to Metamask wallet

Accuracy : From the results, it can be seen that all transfers are accurate when checked with manual calculations. On transferring 5000 Yen, forex tax of 5 percent which is equivalent to 250 Yen is applied. Hence a total of 5250 Yen is reduced from customer A account. On the receiving end, 37.66 US dollars are received, which is equivalent of 5000 Yen in dollars. Therefore, our system is working accurately.

Cost-efficiency : Ethereum network is bigger than Ripple and Stellar and is busy so Ethereum's value is much greater than Ripple and Stellar. The implemented algorithm still provides nearly the same cost efficiency as the existing technologies. According to Ripple official documentation, the minimum transaction cost on the Ripple network is currently set to 0.00001 XRP, which is equivalent to about 0.000004 USD at current exchange rates. These values are not calculated for foreign transactions in the documentary but the estimated cost for cross-border transactions is 0.001 dollars on a busy day. In case of Stellar, according to its official documentation, the transaction cost is 100 stroops (0.00001 XLM), which is equivalent to about 0.0000007 USD at current exchange rates. However, the actual transaction cost that a user pays may be higher depending on the demand for transaction processing on the network and the size of the transaction similar to the case of Ripple. In case of payments made through SWIFT network, the transaction fee varies greatly and are subject to both a fixed fee and a variable fee. The fixed fee for a SWIFT cross-border payment can range from a few dollars to several hundred dollars, depending on the specific circumstances of the payment. The variable fee for a SWIFT cross-border payment is typically based on the exchange rate that is being used to convert the currencies being exchanged. The variable fee for a SWIFT cross-border payment can range from a few basis points (a basis point is equal to 1/100th of 1%) to several percentage points, depending on the

specific circumstances of the payment. Overall, the fees for cross-border payments made through the SWIFT network can vary significantly depending on a number of factors, including the specific banks or financial institutions involved in the payment, the currencies being exchanged, and the location of the sender and recipient. The transaction cost of the implemented smart contract is 121285 gas which is 0.000121285 Ether on a busy network. This is the cost in a worse-case scenario. This cost is efficient given the current market-scenarios. Hence, our system has proven to be cost-efficient as well.

Time-efficiency : Each of the above calls took less than 5 seconds to execute on a stable Network. Moreover, checking the account balance took less than 1 second to execute. The same can be verified by checking the Transaction Hash of the calls on Etherscan. Hence, our system is time efficient. Comparing it with the traditional payment methods such as SWIFT and ACH, these networks are typically slow and expensive, and can take days or even weeks to complete a cross-border payment. According to SWIFT's documentation, the average transaction time for a payment made through the SWIFT network is currently around 2-3 business days while the transaction time in this contract is less than 5 seconds. The currently existing decentralized technologies such as Stellar and Ripple also takes around $3-5$ seconds according to their official documentation. However, it's important to keep in mind that the actual transaction time that a user experiences may vary depending on the specific circumstances of the payment and the demand for transaction processing on the network.

References

1. https://docs.chain.link/data-feeds/price-feeds/addresses/?network=ethereum
2. https://xrpl.org/transaction-cost.html
3. https://developers.stellar.org/docs/encyclopedia/fees-surge-pricing-fee-strategies
4. https://www.researchgate.net/publication/337486692
5. https://rbi.org.in/scripts/PublicationsView.aspx?Id=20315#CH12
6. https://www.google.com/finance/quote