Ethereum Letter of Credit

Dakshit Babbar (2020EE30163) Tushita Pandey (2020MT60307)

Introduction

A Letter of Credit (LOC) is a financial instrument mostly used in international trade to facilitate transactions between a buyer and a seller who may not have an established relationship or trust. It serves as a guarantee from a buyer's bank to a seller that payment will be received on time and for the correct amount, provided that the seller meets all the terms and conditions specified in the LOC.

Traditionally, LOCs have been paper-based financial instruments facilitated through traditional banking systems. However, with the rise of blockchain technology, there has been exploration and experimentation with deploying LOCs on blockchain networks.

Blockchain-based LOCs offer several potential benefits, including increased transparency, efficiency, and security. Smart contracts, programmable agreements executed on blockchain networks, can automate the execution of LOC terms and conditions, streamlining the process and reducing the potential for disputes. Additionally, blockchain-based LOCs can provide greater visibility into the transaction process, enabling parties to track the status of transactions in real-time.

Structure

We implement a smart contract here which will simulate the execution of an LOC. We have written the code for our smart contract in solidity and deployed it on Sepolia, a test network of Ethereum. We as well create a frontend for our application using HTML, JS and CSS.

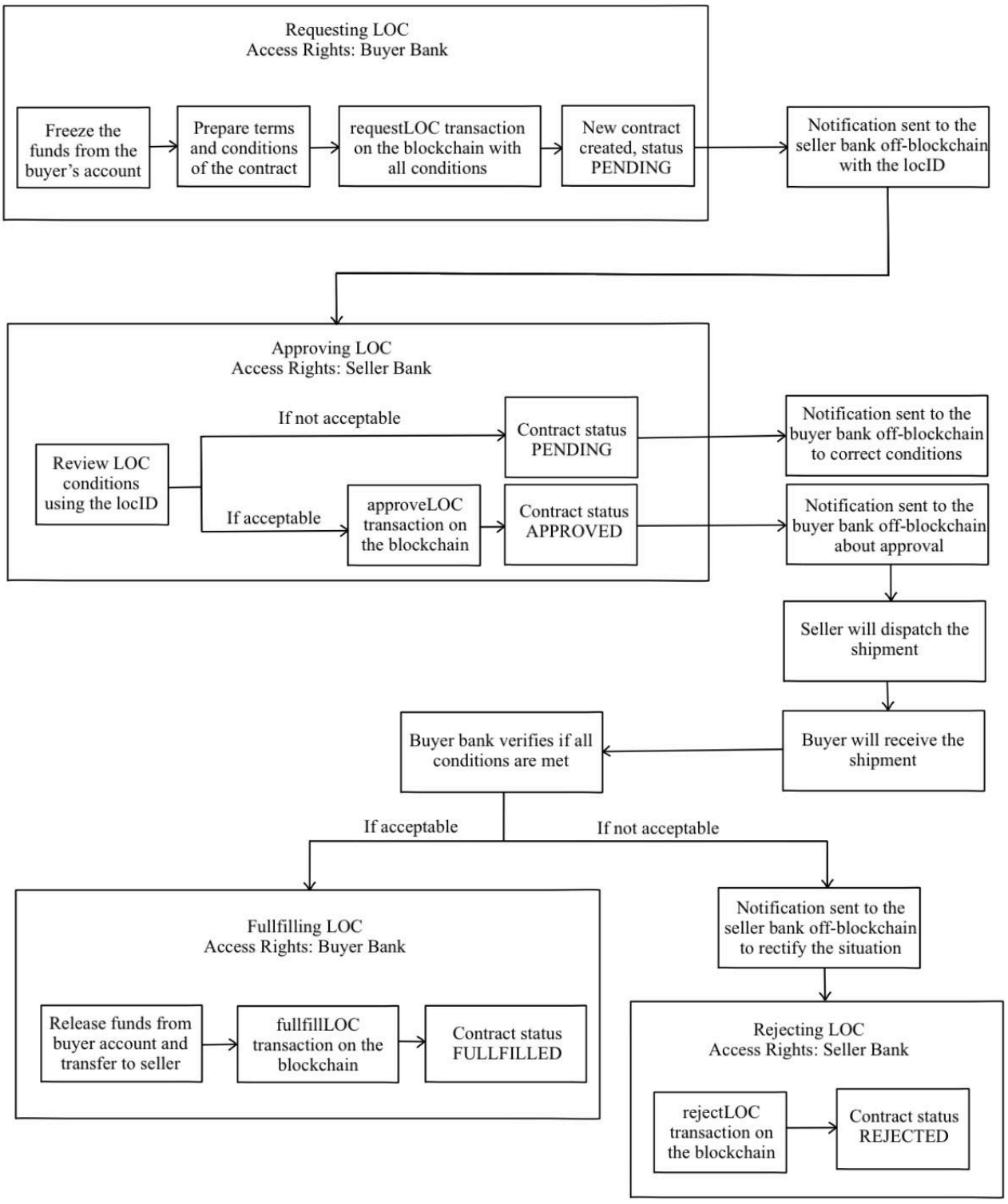
Parties Involved

Buyer: Wants to purchase goods and requires shipping terms fullfillment guarantee
Seller: Supplies the goods and requires a payment guarantee
Buyer’s Bank: Issues the LOC on behalf of the buyer
Seller’s Bank: Receives the LOC that guarantees the payment once terms are met

The buyer and seller agree on terms of the trade, including price and delivery details. They decide to use an LOC as the method of transaction to guarantee the payment.

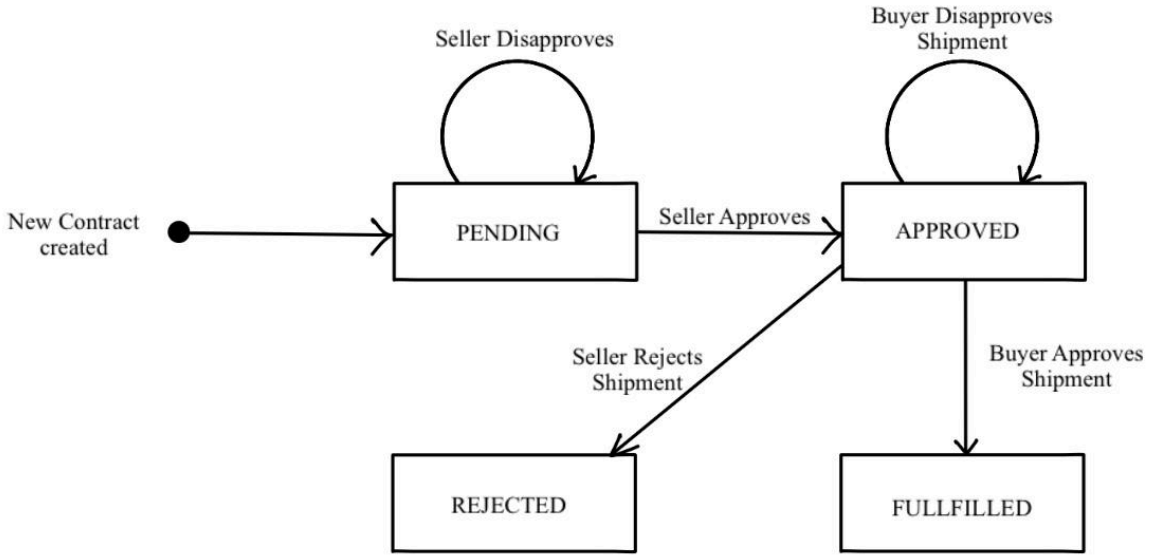
Working

The following shows the working of the letter of credit from the moment when the buyer and seller have agreed on the terms and conditions and have involved their respective banks in the transaction.



Contract States

The LOC seems to work similar to a Finite State Machine (FSM) with the following states and transitions:



Note that for every change in state of the contract we have a transaction being recorded on the block chain. That transaction is either initiated by the seller bank or the buyer bank and is immutably recorded on the block chain.

Each of the transactions cost some amount of gas fees which depends on the network congestion and complexity of the transaction.

Implementation

Following are the implementation details of the smart contract created with the pseudocode of the transaction functions as well as the public view functions.

```
struct LOC {
        uint id;
        address buyerBank;
        address sellerBank;
        uint amount;
        Status status;
        //more conditions of the LOC if needed
}

//dynamic array to store the contracts as they are created
LOC[] public locs;
```

```
function requestLOC(address _sellerBank, uint _amount) external {
      //new locID=length of the locs array
      //make a new contract struct and push into array
      //emit an event with the locID
}

function approveLOC(uint _locId) external {
      //ASSERT locId<locs.length
      //ASSERT the caller is the seller
      //change status to approved
      //emit event with locID
}

function fulfillLOC(uint _locId) external {
      //ASSERT locId<locs.length
      //ASSERT the caller is the buyer
      //ASSERT the status of the contract is APPROVED
      //change status to FULLFILLED
      //emit event with locID
}

function rejectLOC(uint _locId) external {
      //ASSERT locId<locs.length
      //ASSERT the caller is the seller
      //ASSERT the status of the contract is APPROVED
      //change status to REJECTED
      //emit event with locID
   }

function  getLOC(uint  _locId)  public  view  returns  (uint,  address,
address, uint, Status) {
      //ASSERT locId<locs.length
      //return al details as a tupple
   }
```