International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 10 | October - 2022 Impact Factor: 7.185 ISSN: 2582-3930

Ethereum Application for Product Anti-Counterfeiting

Ruya Sakharkar[1], Dr. Bharti Singh[2]

[1]Department of Electronics and Telecommunications, K.J Somaiya College of Engineering, Mumbai
[2]Department of Electronics and Telecommunications, K.J Somaiya College of Engineering, Mumbai

---

Abstract - The project's goal is to create an application that assures consumers don't have to rely solely on merchants to determine whether or not products are authentic. It also assures that producer may use this system to supply genuine products without having to run direct operated outlets, lowering product quality assurance costs dramatically. The trade in counterfeit goods is expanding, hurting the sales and earnings of businesses who are affected. This project proposes a fully-functional blockchain system to prevent product counterfeiting to ensure the identification and traceability of real products throughout the supply chain. Establishments only need to pay minimal transaction fees, and they no longer need to be concerned about the probability of obtaining counterfeit products. This project utilizes Ethereum blockchain and Remix to develop the application code in the form of a Smart Contract. For testing and implementation, Ganache and MetaMask are used, which simulates the blockchain environment and user account respectively. The Client-Side application leverages with Reactjs and Web3js to interact with smart contract and provide its functionalities to the users, maintaining the integrity of the system.

Key Words: Ethereum, Anti-Counterfeit, Solidity, Web3js, Reactjs, Ganache

1.INTRODUCTION

Blockchain is a decentralized, distributed, and oftentimes public, digital ledger consisting of records called blocks that are used to record transactions across many computers so that any involved block cannot be altered retroactively, without the alteration of all subsequent blocks. The use of a blockchain removes the characteristic of infinite reproducibility from a digital asset. It confirms that each unit of value was transferred only once, solving the long-standing problem of double-spending. Security, Privacy, Decentralization, Intrackability, Transparency and Flexibility are the features of the blockchain.

Ethereum is a Blockchain platform that uses a turning-completeness programming language to create smart contracts. On Ethereum, anyone may create smart contracts and other decentralised apps. Users can create any rules they want, including access permissions, transaction types, state conversion equations, and so on. Users of Ethereum will first develop a smart contract in the Solidity programming language, then convert the code to Ethereum bytecode, insert the bytecode into a transaction, and deploy the transaction to the network.

This project is aimed at identifying and tracking real products throughout the supply chain where the trade in counterfeit goods is an ever-growing threat that directly affects the sales and profits of companies affected by this phenomenon. We aim to leverage the advantages of Ethereum to propose a fully-functional blockchain system to prevent product counterfeiting. Establishments only have to pay standard transaction fees, and they no longer need to worry about the possibility of obtaining counterfeit products.

2. Flow of Operation

In this project, manufacturers can use this system to store relevant information on product sales in Blockchain which is accessible to anyone. The total amount of products that can be sold by the seller and the number of products currently left by the seller are openly accessible by customers. The user can use the functions provided by our system to immediately perform verification, at the retailer-end.

The project focuses on three entities namely, Manufacturer, Sellers and Customers. Different functions are tailor-made to be used by either one of these entities during a specific stage of verification. The manufacturer is tasked with pushing seller information to the contract, including the number of products of an item the seller can sell and the seller address. After the seller acquires the manufacturer's authorization, he can then acquire a certain amount of recording rights for the number of products that he can sell.

Consumers can use the system to directly search for whether the seller is the one who is approved by the manufacturer and whether there are remainder products available for trading. On satisfactory verification, consumer can then proceed to pay the seller. All the above entities will be present in the form of distinct Ethereum account each having

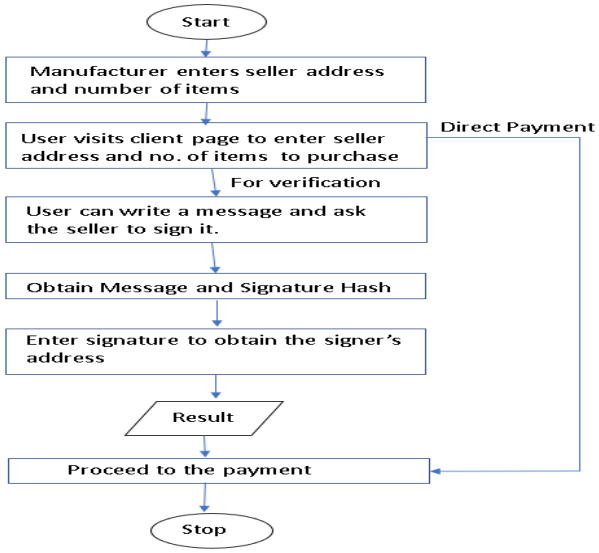


Fig 1: Flowchart of operations

its own distinct address. The interactions would be performed over the application deployed.

3. Software Requirements

The proposed project uses Ethereum as the back end Blockchain operating system and uses Ethereum's proprietary programming language Solidity as the high-level programming language for writing and deploying smart contracts.

Ganache is a high-end development tool used to run custom local blockchain for both Ethereum dApp (Decentralized Application) development. Ganache is helpful in all parts of the development process. The local chain allows one to develop, deploy and test the projects and smart contracts in a deterministic and safe environment.

MetaMask is a software cryptocurrency wallet used to interact with the Ethereum blockchain. It allows users to access their Ethereum wallet through a browser extension or mobile app, which can then be used to interact with decentralized applications.

The user interface seen by the user will be a web page. The server end of the web page is forged using the http-server suite, which was provided by node.js and web3.js is used as the link between the smart contract and the user interface. After deployment, the smart contract interacts with the Web3 provider running in the local server. In this case, Metamask is the Web3 provider used.

4. Implementation

Steps for the implementation and setup:

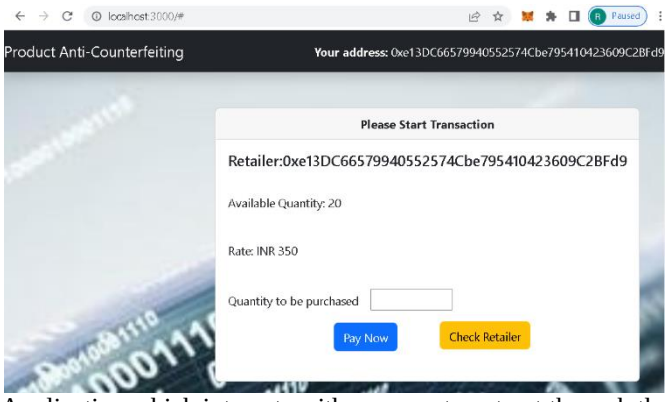
4.1 Login Process: Before establishing a connection to the system, the user has to choose which account to log in through a web3 wallet.

4.2 Deployment of the smart contract: All the functions pertaining to the logic of the project are coded in a smart contract in the language of Solidity. The smart contract was deployed over Remix IDE (Remix is a browser-based compiler and IDE that enables users to build Ethereum contracts with Solidity language and to debug transactions) and was incorporated into the app using Nodejs and Reactjs.

The smart had many functions which performed necessary tasks. First function allowed the manufacturer to add a seller along with a specific number of item approved to be sold by the seller. Another function allowed a consumer to check whether a given retailer is the one approved by the manufacturer.

Another set of functions implemented the tasks of verifying the retailer's address by signing a data using the consumer's private key and asking the seller to match it. Successful match states correct seller and consumer identity.

4.3. Client Side Application: In order to make this system accessible by customers and to protect the smart contracts from malicious modifications, we have developed a Client-Side



Application which interacts with my smart contract through the use of Reactjs and Web3.js

Fig 2: Client Side Application

The application detects the account from which user has logged in and displays the same on the top-right corner.

The first card shows the address of the currently approved seller by the manufacturer along with the quantity available at that moment. The user can enter the number of quantity that he/she wants to purchase and just proceed to payment by clicking on 'Pay Now' button. This will open a web3 wallet prompt (Metamask in this case) asking the user to confirm the details of the transaction.

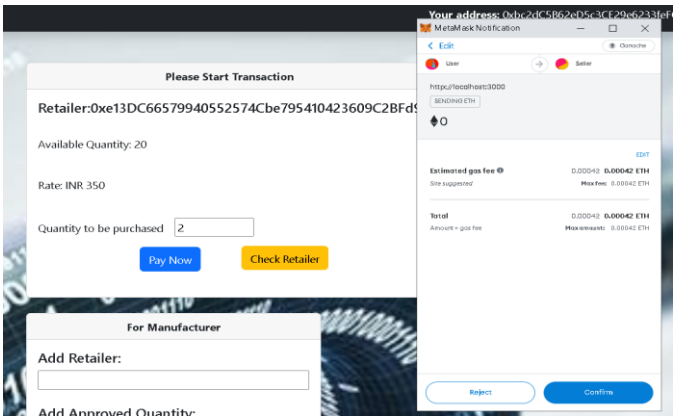


Fig 3: Payment Prompt by Metamask

It should be noted that the amount is payable only in terms of Eth (Ethers). Although due to the volatile nature of ethers we have kept the price of items constant in terms of Indian Rupee which is 350. According to the quantity to be purchased this price will then be converted to Gwei (a denomination of Ethers). This converted Gwei amount is exactly what the user will pay along with some transactional gas fee.

The second card is reserved only for manufacturers. This enables them to approve a certain retailer along with some specific quantity. However, this transaction will only work when the account that is logged in to the application is that of the Manufacturer. In case of mismatch the transaction will not be initiated. Such mismatch will be shown by a prompt in browser.

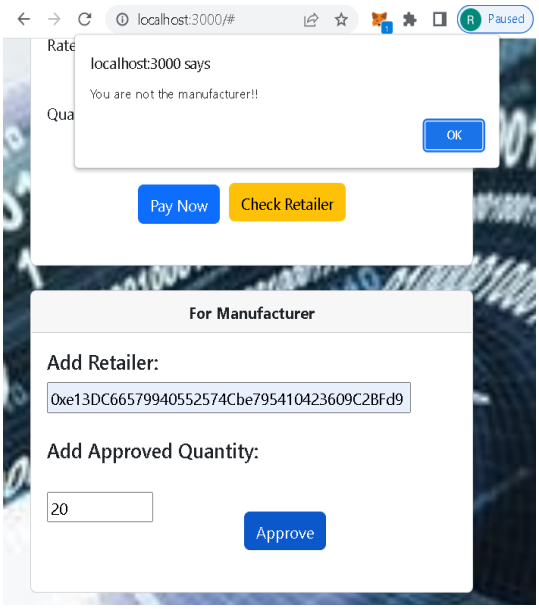


Fig 4: Prompt when approval is not done by the manufacturer

When manufacturer is approving a seller he/she will enter the desired retailer's address and approved quantity and then click on Approve button. The manufacturer will then need to confirm a smart contract transaction through Metamask to complete the approval function.

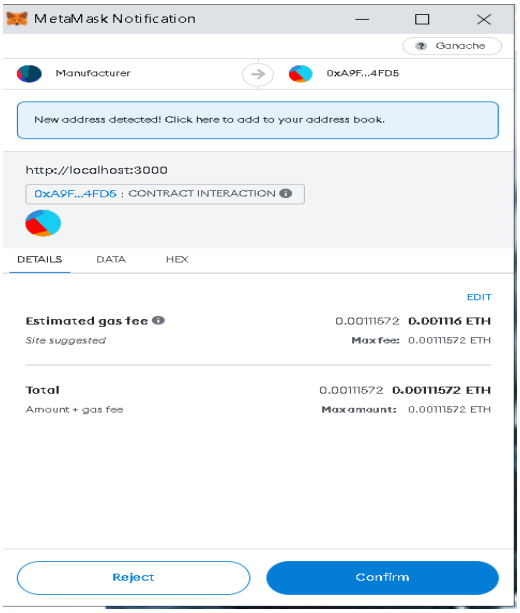


Fig 5: Prompt when the manufacturer is approving a retailer

After confirming the transaction, the change is reflected in the application after refreshing it. New seller's address and quantity available are displayed accordingly. And all payments from thereon will be made to this new added retailer only. After each payment, quantity will be reduced which will be displayed on the website.

The user has the option to verify the address of the retailer by signing a message using its private key and verifying the signed message with the arguments passed. Successful match means that the retailer address, the user's address, the message is correct. Unsuccessful match means some discrepancy.

This provision of signing and verifying a message is available in the application. User just needs to type any message and then sign it by clicking on Sign message button and then approving a transaction through Metamask.

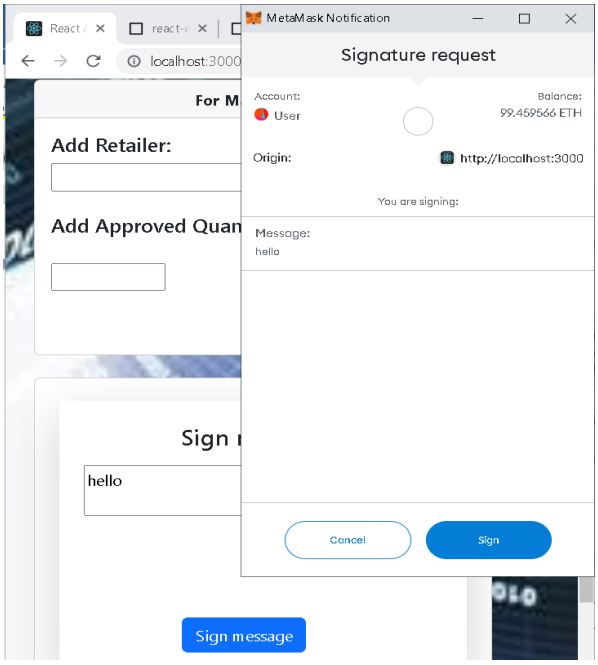


Fig 6: Prompt for signing a message

After signing a message the application will then return the message, the signer's address and the 32 bytes hash value.

Fig 7: Result after signing a message

Fig 7 shows the output obtained after signing a message. This 32 byte hash is the signature produced by the user's private key which is used to verify retailer's address

This signature along with its input parameter is then cross verified and result is displayed. Valid signature denotes no discrepancy whereas invalid reflects some discrepancy in the signature.

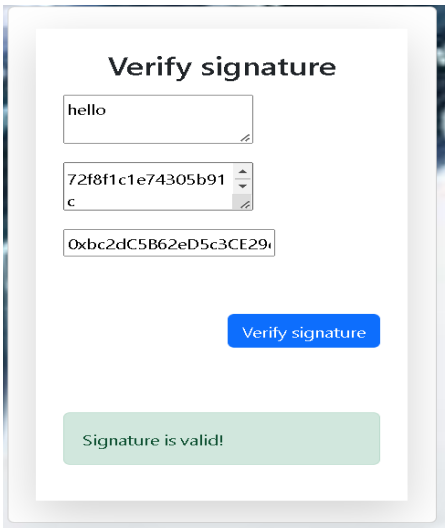


Fig 8: Valid Signature

After verification the user can choose to proceed with the payment as desired.

5. CONCLUSIONS

Manufacturers can use the system to store relevant information regarding their product sales in Blockchain which can be accessed by anyone. The total amount of sales that can be sold by the seller and the number of products currently left by the seller are transparent and can be viewed by the customer. The user can use the functions provided by the system to immediately perform vendor-side verification. The system provides identity verification by using digital signatures. There are no other means to decrypt the private key of the key owner and therefore the user must note that he doesn't leak his private key anywhere even accidentally. We have successfully implemented on a test network, a functional low-cost system for product anticounterfeiting that has the advantages of transparency, authenticity, security, and privacy. We have also developed a client-side application that checks whether the seller address matches the one set by the manufacturer through web3js and also displays the quantity of the product available with them.

6. FUTURE SCOPE

A provision of sending the details of all transactions to the consumer's email could be added. There is also scope to fetch the live price of Ethereum and use it to convert from Indian Rupee.

ACKNOWLEDGEMENT

We would like to convey our heartfelt gratitude to Dr Shubha Pandit, Principal, K.J Somaiya College of Engineering, Mumbai for her tremendous support and assistance in the completion of our project. We would also like to thank our vice-principal, Dr Ameya Naik, for providing us with this wonderful opportunity to work on a project. The completion of the project would not have been possible without their help and insights. We would always be grateful to them.

REFERENCES

1. K. Toyoda, P. T. Mathiopoulos, I. Sasase and T. Ohtsuki, "A Novel Blockchain-Based Product Ownership Management System (POMS) for Anti-Counterfeits in the Post Supply Chain," in IEEE Access, vol. 5, pp. 17465-17477, 2017, doi: 10.1109/ACCESS.2017.2720760.
2. V. P. Ranganthan, R. Dantu, A. Paul, P. Mears and K. Morozov, "A Decentralized Marketplace Application on the Ethereum Blockchain," 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), 2018, pp. 90-97, doi: 10.1109/CIC.2018.00023.
3. R. Taş and Ö. Ö. Tanrıöver, "Building A Decentralized Application on the Ethereum Blockchain," 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2019, pp. 1-4, doi: 10.1109/ISMSIT.2019.8932806.
4. T. T. Huynh, T. Tru Huynh, D. K. Pham and A. Khoa Ngo, "Issuing and Verifying Digital Certificates with Blockchain," 2018 International Conference on Advanced Technologies for Communications (ATC), 2018, pp. 332-336, doi: 10.1109/ATC.2018.8587428.

BIOGRAPHY

Ruya Sakharkar graduated in Electronics and Telecommunication from Dr. Babasaheb Ambedkar Technological University, Lonere in 2019.

She is pursuing Masters in Electronics and Telecommunication from K.J Somaiya College of Engineering, Mumbai. She is currently working as a Quantitative Analyst (Python).

Her research interests interest include Ethereum, Data Mining, Machine Learning and Deep Learning.