DOI: 10.15849/IJASCA.250330.02

Int. J. Advance Soft Compu. Appl, Vol. 16, No. 3, March 2025
Print ISSN: 2710-1274, Online ISSN: 2074-8523
Copyright © Al-Zaytoonah University of Jordan (ZUJ)

Hybridization of Blockchain and Cryptography to Secure a Transaction Using Smart Contacts

Debbab Mohamed, ALI PACHA. A.B, BOUKLI HACENE.S

Department of Science and Technology, Tiaret University, Algeria
EEDIS Laboratory, Computer science department, Djilali Liabes University.
Sidi bel abbes, Algeria
E-mail : mohamed.debbab@univ-tiaret.dz
Department of Electronic, ORAN University, Algeria
E-mail : adda.alipacha@univ-usto.dz
Department of Computer Science, Sidi Bel Abbes University, Algeria
E-mail : boukli@univ-sba.dz

Abstract

Although often equated to a large database owing to the nature of its use, Blockchain technology is fully distinct due to its fully uncontrolled structure. These traits make it beneficial for changing how finances are exchanged as it is safe and trustable. Moreover, integrating modern cryptography such as AES-256 for encryption and ECDSA for signatures into a blockchain system can elevate its level of security to greater heights. This research examines the prospect of developing a hybrid model which incorporates the use of blockchain in accordance with cryptography to reinforce security for financial transactions. It implements a two-tier architecture comprising of a public and private layer to enable visibility and secrecy. Apart from the aforementioned cryptographic measures for security, an AI-based concept of fraud detection is integrated into the model which internally checks for suspicious activity in real-time with a 91% average success rate. The proposed model tries to solve issues like scalability and efficiency and does this through deploying smart contracts on a private Hardhat blockchain. Testing out in the real world averaged the transaction time to be around 15 seconds while the system performed well when multiple processes were executed at the same time. Not only does this project serve to emphasize the chances of hybrid encryption blockchains in the reduction of risks around online transactions but also defines the role in establishing a more secure, efficient, and scalable financial ecosystem.

Keywords: *Blockchain, Transaction, Security, Cryptography, Smart Contracts*

1 Introduction

All of the sudden leaps that occurred with digital technology have totally shattered every aspect of our world, money transactions included, and in this digital age, the way in which transactions and personal information are protected is what worries all of us. Even though this system efficiently handles most transactions, its trust-based mode remains tainted by inherent corruption. and that transports us to the blockchain universe. If electronic cash were strictly peer-to-peer, online payments could be sent directly between parties without

Received 7 November 2024; Accepted 28 December 2024

the need to go through any banking institution. Its beginning is a little vague, even to this day, and is generally traced back to 2008 and the creation of the electronic currency Bitcoin by an anonymous person or group of people by the name of Satoshi Nakamoto. Here's when cryptography comes into play. All the sensitive information can be encrypted with cryptography so that it will have authenticity, integrity, and confidentiality. We can also enhance the security of transactions by encrypting the data before it is put on the blockchain and using digital signatures to ensure that the people conducting the transactions are who they say they are. And in this project, we aim to make a hybrid approach using both techniques to create a full system for securing digital transactions [1].

2 Background

2.1 Objective

This project will involve the design of such a digital transaction system with the immutability and decentralization properties coming from the blockchain and the credit and integrity aspects coming from the field of cryptography. The idea being to hone in on the encryption of any personal transaction data into a privately owned chain of blockchains. Without compromising the accuracy of transactions or transparency of the process of that there is no central authority controlling the transactions, this hybrid approach eliminates reliance on centralized authorities, enabling secure peer-to-peer financial transactions [2].

2.2 Blockchain

Blockchain is frequently utilized within the money sector, where both peers, the sender and receiver, encounter multiple challenges, so building a decentralized digital ledger that securely records transactions across a network of computers [6]. Each transaction is grouped into a block, which is then linked to the previous block, forming a chronological chain. This structure provides several key benefits [3].

- **Immutability:** It is a peculiarity of blockchains that any features or events that have been recorded cannot be whittled down or manipulated in any way.
- **Transparency:** All the members of the blockchain network have the same data about a given transaction, promoting honesty.
- **Resilience to Tampering:** The decentralized nature of the blockchain makes it very difficult to make illegal changes. It becomes much harder for malicious users to tamper with transaction data, because there is no real control to one central authority, and so the integrity of the ledger is enhanced.
- **Consensus Mechanism:** Validator nodes accept transactions on the blockchain according to propositions such as Proof of Work (PoW) and Proof of Stake (Pos). These approaches are employed so that no scheme rehearsed in the network is ever attempted more than once. These mechanisms increase the security of the systems in such a way that no transaction can be recorded on the block chain without the cooperation of many nodes [4].

Despite these advantages, blockchain systems face challenges in scalability as transaction volume increases. This project explores scalability solutions, such as optimized consensus mechanisms and private blockchain networks, to handle large-scale deployments.

2.3 Cryptography

Cryptography is the foundation of securing digital transactions because it converts readable transaction data into an unreadable form to prevent unauthorized access. If we take a close look, we can simplify the concept of blockchain into 2 main concepts: cryptography and hashing, let's start off by breaking these 2 concepts down at their core. Such protocols are classified into two broad types of encryption technologies; symmetric encryption and asymmetric encryption. Symmetric key encryption encrypts and decrypts the information using the same single key making it effective, but it is dependent on the safe sharing and storage of its key. As for asymmetric encryption, it uses both keys: the public key for encryption and the private key for decryption, which brings an extra level of security, which is very appropriate for decentralized systems such as blockchain.

We will be using AES-256, a well-known symmetric encryption algorithm, for securing data in transit in this project. That means that, even if intercepted, data can't be read without a key to decrypt. This is a standard that stands for advanced encryption standard, which is a symmetric cipher or an encryption algorithm which is very strong as only one key is employed in the encryption or the deciphering process. For effective encryption and decryption of the data within the AES standard, it’s required in this context to utilize at least 256 bits key length. Considering the systems of blockchain, it is AES-256 that network members are considered to encrypt database transactions (hence allowing easy access only to those who know the key) Fig [1].

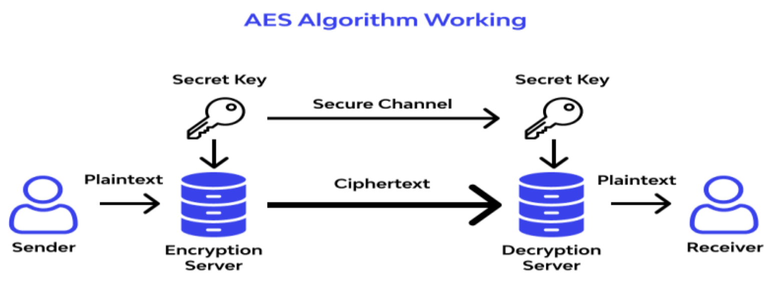


Fig 1: AES encryptions

We also incorporated digital signature assurance for the authenticity of a transaction using the ECDSA, or the Elliptic Curve Digital Signature Algorithm [12]. The Elliptic Curve Digital Signature Algorithm is also another asymmetric encryption that also fosters the authenticity and integrity of certain transactions. As all other cryptography methods such a system could use two keys, in this case, an elliptic curve ECDSA is used. There is a private key that signs the transaction and a public key that verifies it. This means that it is only the owner of the private key who would be able to approve any transaction made, thus eliminating the chances of fraudulent activities by unauthorized persons. As relates to blockchain systems, the relevance of ECDSA chains as a skill or ability to provide trust and identity verification for safe and trustless transaction processing is for that reason the very base. With regard to the decentralized transaction validation, AES-256 and ECDSA are cooperating in providing the solution to the two upset problems of the blockchain networks confidentiality or nonrevealing format and authenticity. Data is secure and verifiable, and the integrity is guaranteed Fig [2]. Digital signatures ensure that no tampering of information has occurred and that indeed it was sent by the right sender [5]. Every transaction gets signed by the sender's private key and must be checked using the receiver's public key for validity [7]. Blockchain makes use of hash functions like Secure Hash Algorithm SHA-256 to uniquely identify a transaction. Each block contains hashes of the previous blocks so that in case of tampering, the integrity of the chain would be broken [3].

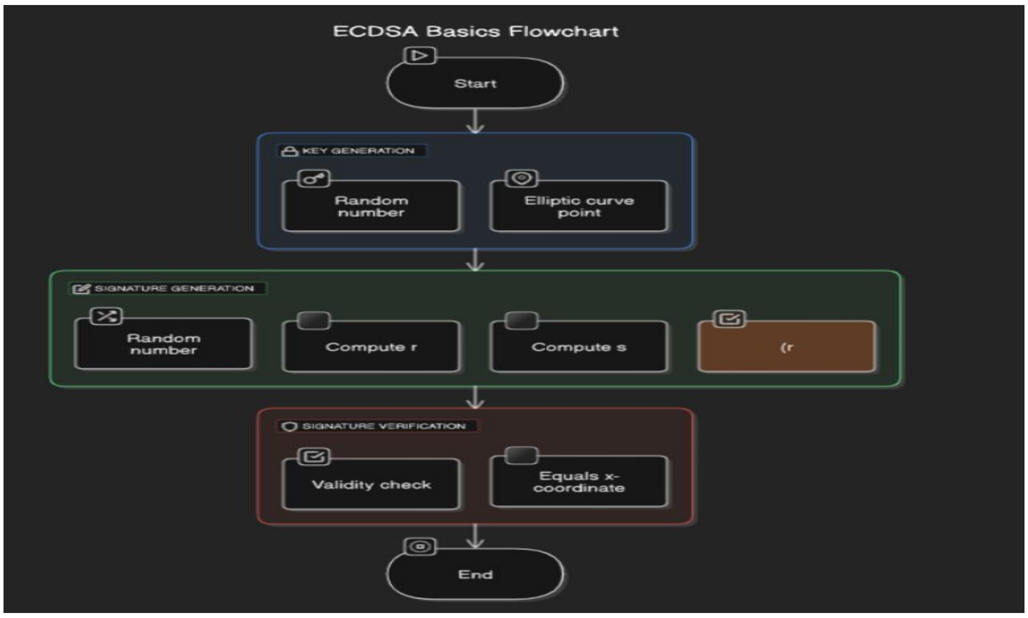


Fig 2: Elliptic Curve Digital Signature Algorithm Diagram

2.4 SHA-256 Hashing and Cryptographic Alternatives

The utilization of SHA-256 hashing within the project is critical to protecting the integrity and the immutability of the blockchain. It does this by deriving a unique hash of a specified length per transaction which allows the validation of transactions to take place without the need to disclose sensitive information concerning the transaction, the property of speed and collision free made it very qualified for security purposes for the Blockchain. Further, the transaction data is protected in transit by AES-256 encryption while authenticity as well as non-repudiation is achieved through digital signatures by ECDSA.

Potential solutions from quantum resistant ciphers and RSA were, indeed, also considered but there was a clear indication that SHA-256 was going to be used of the project owing to the availability of certain factors. Although RSA is one of the very boldest guard algorithms, very resource-intensive, particularly when it comes to multiple transactions in a blockchain based setting. The longer the keys, the stronger is the security, thus requires quite heavy computation, adversely impacting performance as well as scalability in real time applications. On the other hand, SHA-256 is still one of the secure algorithms and it consumes less power which allows it to be used in places where many transactions are done within seconds such as the blockchain.

3 Smart contract and solidity

3.1 Smart contract

In what is called smart contracts, the conditions of the contract are directly written into lines of code. These exclude middlemen and automatically enforce and verify whether a transaction has taken place. In this project, we write and implement transaction logic using a smart contract programming language on the Ethereum platform, namely Solidity, in a transparent and secure way. Smart contracts are highly important when it comes to ensuring security for our hybrid blockchain system. Because they allow transactions to run in a secure, tamper-proof automation mode (discarding any third-party control). This allows smart contracts; once coded, it cannot be changed, and everything is transparent on the blockchain. This enhances the tamper-proof nature and the reliability of the system.

Automation of execution of transactions are automatically handled, which protects against fraudulent activities based on predetermined requirements; an example is payment verification [7].

3.2 Solidity

We coded our smart contracts in Solidity for transaction verification and security in this project. Solidity allows us to do the following because it is designed especially for Ethereum based platforms:

- Logically define how to execute the transactions based on business rules.
- Digitally create, verify, and enforce a contract without any centralized authority's intervention.
- Ensure that the smart contract is deployed on a private Hardhat blockchain to enhance privacy and control over the transactions.

Following are a few other key features of Solidity in this project:

- Events and Functions: The features of Solidity will enable us to define functions that encapsulate transaction logic, and events will be emitted on particular conditions such as the successful execution of a transaction
- Best Practices for Security: Input validation, error handling, and reentrancy guards were just but a few best practices applied to make the smart contracts resistant to most of the known attacks [8].

3.3 Private Hardhat Blockchain

The smart contracts in this project were set up on a private Hardhat blockchain a general purpose Ethereum development environment. Hardhat allows running a local Ethereum network for testing and deploying so that one can deploy and debug your contracts seamlessly, without exposure on the mainnet (main network) and without any real currency usage. Below, we will explain its features, setup, and advantages used for this project.

What is Hardhat?

- Hardhat is a development framework for Ethereum used to compile, deploy, test, and debug decentralized applications (dApps). It is one of the most utilized tools across the blockchain ecosystem due to its flexibility and multiple in-built tools that can help simulate an instance of the blockchain [9].
- Ethereum compatible Network: Hardhat lets one create an Ethereum compatible blockchain the kind that can emulate real network deployment of contracts locally, thereby emulating gas and executing transactions.
- Local Blockchain: Hardhat's local blockchain is fast, resettable, and forks any live blockchain state for fast testing and iteration while developing.

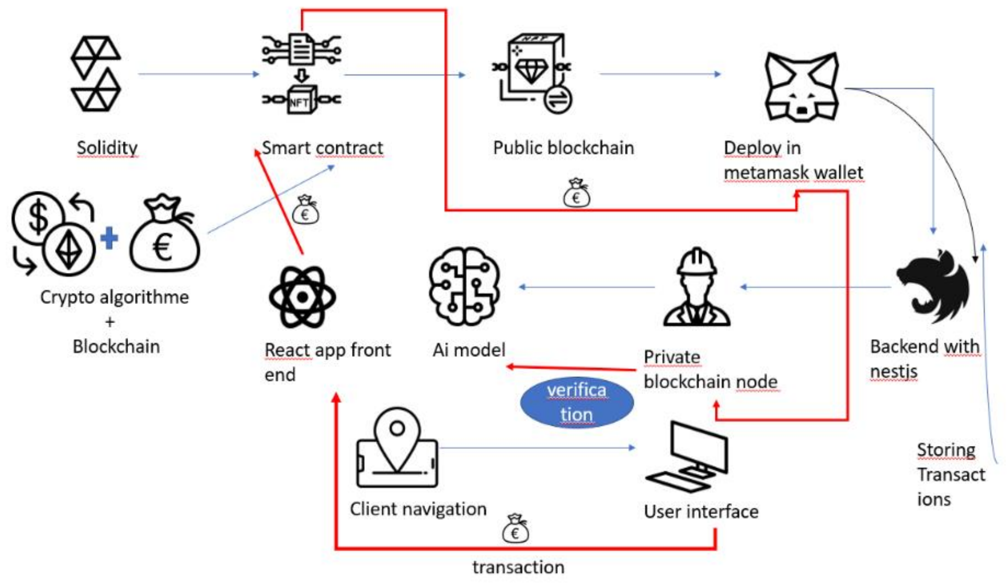


Fig 3: Legend for System Architecture Diagram

- **User:** Represents the end-user interacting with the system through MetaMask (software cryptocurrency wallet).
- **MetaMask:** A browser extension that allows users to interact with the Ethereum blockchain and manage their digital assets.
- **React Frontend:** The user interface built with React, providing an interactive environment for users to initiate transactions and view results.
- **Private Hardhat Blockchain:** A local blockchain environment where smart contracts are deployed and tested without real currency, facilitating rapid development.
- **Smart Contracts:** Self-executing contracts with the terms of the agreement directly written into code, running on the private blockchain.
- **Nest.js Backend:** The server-side application that handles communication between the frontend and the blockchain, processing requests and managing data.
- **AI Model for Fraud Detection:** A component that analyzes transactions in real time to detect and flag potentially fraudulent activities.

4 Technologies Used

For this project, we used a variety of technologies combined to implement an unbreakable yet robust system of transactions. At its core, MetaMask serves as a browser extension that connects users to the Ethereum blockchain through a graphic user interface, paving the way for interactions with smart contracts and digital assets. Fig [3] MetaMask securely manages wallet functions like signing transactions. and our role is to develop these smart contracts that contain a set of rules to secure transactions in a safe and easy manner [10].
For developing and testing our smart contracts, we used Hardhat, an Ethereum development framework. Hardhat provided a local blockchain network, which allows us to manipulate the smart contract from deployment to testing and debug without the need to use the main Ethereum network. The hardhat local setup was essential during the development stage, enabling us to iterate on contract logic without worrying about real-world currency or network fees. Another benefit of Hardhat was to simulate and monitor transactions. This can be beneficial for us because it helps us catch any bug and fix it before the deployment on a live network.

Coding is all about logic, and it's handled in the backend, and for that, our go-to was Nest.js, a JavaScript framework for backend development thanks to its modular and scalable architecture. Nestjs allowed us to handle logic, test the rules of the smart contract, and implement debugging statements to make sure that everything works smoothly. We filled the gap between the frontend and the blockchain through its positioning as an intermediary in transaction processing. This includes verification of signatures and storage of key data. Its flexible architecture also made it easier to integrate our AI model, allowing for efficient data processing and analysis within the backend and adding another layer of security on top of that.
To understand how our smart contract operates, it's essential to dig deeper into the rules governing the transactions. This section will cover the detailed processes involved in writing these rules and executing transactions within the project.

4.1 Smart contract and Transaction Process

Building on previous concepts Our smart contract, Secure Transaction, is specifically designed to facilitate secure Ethereum transactions between two parties, using a combination of cryptographic algorithms and blockchain technology to ensure strict security measures. The process begins with the store Transaction function, which initializes the transaction by capturing essential details, including the recipient's address, the amount of Ether, a cryptographic hash, a digital signature, and any encrypted data. This function is responsible for verifying the transaction's validity, including checks on the exact Ether amount and a valid digital signature. Next, the verify Transaction function confirms that the Ether amount matches the transaction details provided by the user and aligns with the sender's identity, using the Elliptic Curve Digital Signature Algorithm (ECDSA) to produce a cryptographic hash that corresponds with all transaction details. After the verification was successful, he securely logs transaction details into a mapping function that stores them on the blockchain and transfers the specified Ether amount to the recipient's account. To provide transparency, the contract emits a Transaction Stored event, publicly logging transaction information on the blockchain [11].

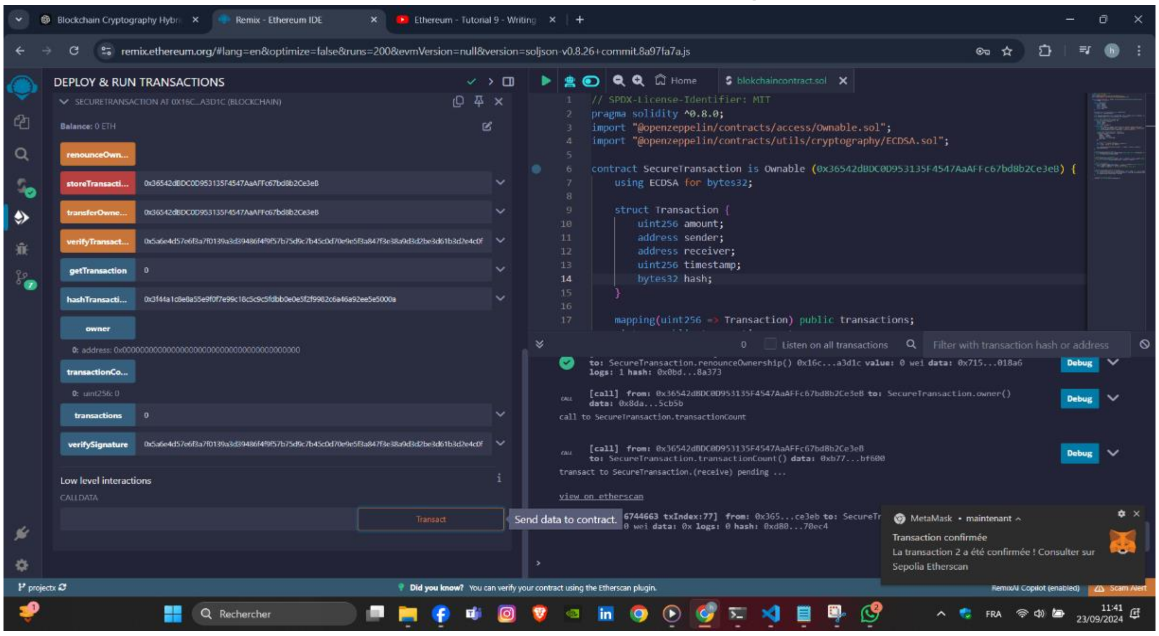
Additionally, the hash Transaction function generates a unique identifier for each transaction using the public addresses of both parties, creating a fingerprint on the blockchain. Finally, the verify signature function uses the generated hash and digital signature to confirm the sender's identity.

4.2 Challenges Encountered Using Remix IDE

First, the smart contract was tested and deployed in Remix IDE Fig [4], an online integrated development environment for Solidity. In tests, several problems were encountered regarding too little test Ether supplied while simulating transactions, which resulted in failed transactions. Although Remix is useful in testing a small-scale application, it had certain limitations with regard to deployments and execution of large numbers of transactions involving cryptographic operations.
Due to these issues, the project migrated to the Hardhat development environment. Hardhat provided a local Ethereum network that allows:

- Control of the gas: Making sure one has enough balance to cover the cost of deploying contracts and sending transactions.
- Faster development cycles, because you can deploy, test and debug more effectively locally than on Remix IDE.

- More customization: Hardhat allowed me to script complex test scenarios and thus interact with the smart contract for more extensive testing [12].


Blockchain Cryptography Hybr
Remix - Ethereum IDE
Ethereum - Tutorial 9 - Writing
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js
DEPLOY & RUN TRANSACTIONS
SECURETRANSACTION AT 0X16C...A3D1C (BLOCKCHAIN)
Balance: 0 ETH
renounceOwn...
storeTransacti...
transferOwne...
verifyTransact...
getTransaction
hashTransacti...
owner
transactionCo...
transactions
verifySignature
Low level interactions
CALLDATA
Transact
Send data to contract.
Home
blokchaincontract.sol
Listen on all transactions
Filter with transaction hash or address
Debug
projectx
Did you know? You can verify your contract using the Etherscan plugin.
RemixAI Copilot (enabled)
Scam Alert
Rechercher
FRA
11:41
23/09/2024

```
storeTransacti...   0x36542dBDC0D953135F4547AaAFFc67bd8b2Ce3eB
transferOwne...     0x36542dBDC0D953135F4547AaAFFc67bd8b2Ce3eB
verifyTransact...   0x5a6e4d57e6f3a7f0139a3d39486f4f9f57b75d9c7b45c0d70e9e5f3a847f3e38a9d3d2be3d61b3d2e4c0f
getTransaction      0
hashTransacti...    0x3f44a1c8e8a55e9f0f7e99c18c5c9c5fdbb0e0e5f2f9982c6a46a92ee5e5000a
owner
    0: address: 0x0000000000000000000000000000000000000000
transactionCo...
    0: uint256: 0
transactions        0
verifySignature     0x5a6e4d57e6f3a7f0139a3d39486f4f9f57b75d9c7b45c0d70e9e5f3a847f3e38a9d3d2be3d61b3d2e4c0f
```

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/cryptography/ECDSA.sol";

contract SecureTransaction is Ownable (0x36542dBDC0D953135F4547AaAFFc67bd8b2Ce3eB) {
    using ECDSA for bytes32;

    struct Transaction {
        uint256 amount;
        address sender;
        address receiver;
        uint256 timestamp;
        bytes32 hash;
    }

    mapping(uint256 => Transaction) public transactions;
```

```
to: SecureTransaction.renounceOwnership() 0x16c...a3d1c value: 0 wei data: 0x715...018a6
logs: 1 hash: 0x0bd...8a373

[call] from: 0x36542dBDC0D953135F4547AaAFFc67bd8b2Ce3eB to: SecureTransaction.owner()
data: 0x8da...5cb5b
call to SecureTransaction.transactionCount

[call] from: 0x36542dBDC0D953135F4547AaAFFc67bd8b2Ce3eB
to: SecureTransaction.transactionCount() data: 0xb77...bf600
transact to SecureTransaction.(receive) pending ...

view on etherscan
6744663 txIndex:77] from: 0x365...ce3eb to: SecureTr
0 wei data: 0x logs: 0 hash: 0xd80...70ec4
```

MetaMask • maintenant
Transaction confirmée
La transaction 2 a été confirmée ! Consulter sur Sepolia Etherscan

Fig 4: Remix IDE with Deployed Smart Contract

5 Workflow overview

The complete workflow of the smart contract system will be described in this section, along with how it integrates with the frontend (User Interface), backend (Hardhat), and AI model for data processing.
The Secure Transaction smart contract: is written in Solidity and compiled using Hardhat as our development environment. With Hardhat we can deploy the contract to Ethereum blockchain or, more probably a local testnet.
The process of a transaction starts when the sender triggers one from frontend interface by specifying receiver address, amount of transaction to be transferred and any confidential data if required. The frontend then passes this information to the smart contract in a call to store transaction, where inside the contract we make checks verifying hash and digital signature of transaction using ECDSA. If successful, the amount of specified Ether is then on-chain transferred from the sender to receiver securely. Upon finishing, the contract emits a Transaction Stored event so on chain interfaces can monitor and acknowledge that last transaction completed successfully.
Hardhat is configured for local development to simulate a blockchain network for the deployment of contracts, interaction with them, and running automated tests.
Local Development Network: Hardhat spins up a local Ethereum blockchain one that mimics the main network but allows for fast, non-real ETH testing and deployment. Then Hardhat compiles the contract and deploys it onto the Hardhat network. A deployment script has been created to help automate the deployment process and pass all the necessary parameters, like constructor inputs.
Unit Tests: Mocha and Chai (libraries in the JavaScript ecosystem) are used to write tests in ensuring smart contract functionality behaves as expected.
Examples include the following: transaction storage and verification of signatures.

Simulating Transactions: Using the Hardhat development network, several accounts could be simulated to send transactions and have their behavior checked before deploying on a public blockchain.
After the contract has been fully tested on Hardhat, it can be deployed to a public test network say, Rinkeby or the Ethereum mainnet. This will deploy the functionality of executing real-world transactions with actual Ether.
UI Framework: The frontend has been created using modern web development frameworks such as React.js. This allows the users to have an easy and user-friendly interface through which the user will interact with the smart contract Fig [3].
Form Inputs: A user will enter all his details in this form, which are required in the transaction, including the address of the receiver, the amount in the transaction, and any files that he wants to get encrypted.
MetaMask Wallet Integration: The frontend is integrated with MetaMask to connect users Ethereum wallets to the application for secure transaction initiation.
Initiate Transaction: A user initiates a transaction by submitting his data through the UI. In the sequence, this data gets forwarded to the smart contract through either Web3.js or Ethers.js.
Real-time Feedback: During the time an event notification from a smart contract is sent to the UI, real-time feedback would show up in the status of a transaction as either pending, confirmed, or failed.
Error Handling: Errors common on the frontend are handled with insufficient gas fees for a transaction, wrong inputting, and even complete failure, sending messages to the user, Fig [3] illustrate these concepts.

5.1 AI Model Integration

The smart contract stores some encrypted information of transactions on chain. This can be used by the AI model for some predictive analytics or decision-making purposes. Preprocessed transactional information, such as volume, time of day, or encrypted data, is inputted into the AI to identify patterns or abnormalities [13].
The model of AI at which this project has implemented is a neural network formulated for binary classification between fraudulent and legitimate transactions. The constructed three hidden layers employ ReLU (Rectified Linear Unit) activation functions followed by softmax layer output classification to ensure the capability of the model to represent complex patterns in transactional data. Fig [5] Historical transaction data was collected as a training dataset during the actual development of the system. Data generated through backend logic integration with Etherscan (blockchain explorer and analytics platform for the Ethereum blockchain) was formatted into CSV and put through rigorous data preprocessing, where noise removal, balancing for relevance to the classification problem, was carried out. The preprocessing steps included normalization, outlier removal, and encoding categorical variables. The training process involved two primary steps, the first step is forward propagation, where input data passed through the network generated predictions, and backward propagation, where the model’s parameters weights and biases were iteratively adjusted by calculating gradients of the cost function using gradient descent to minimize error. The multiplication of this input by weights, summation, and bias was done by neurons before passing the output through the ReLU activation functions to accommodate non-linearity.

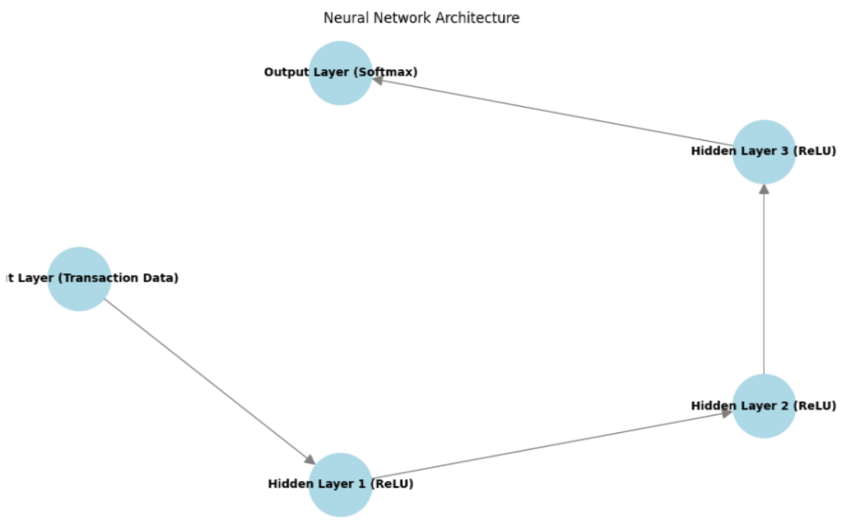


Fig 5: AI Architecture

model's trustworthy performance could be seen as it achieved 91% accuracy with an 85% precision, recall of 88%, and ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) of 0.92, which fairly confirms strong classification ability. The model was converted into an API **(**Application Programming Interface**)** for real-time fraud detection, with responses effectively delivered within approximately 300 milliseconds' time, which results in negligible lag. The application also integrates perfectly with smart contracts, whereby suspicious transactions flagged by the model trigger additional verification or halt execution, taking advantage of Solidity's automated enforcement capabilities. Thus, a project that exploits the penetration of blockchain and cryptographic security, along with artificial intelligence, has been established to provide solid groundwork for secure transactions, suggesting the possibilities of using such advanced technologies to address future challenges in completely digital and financially evolving worlds Fig [6].

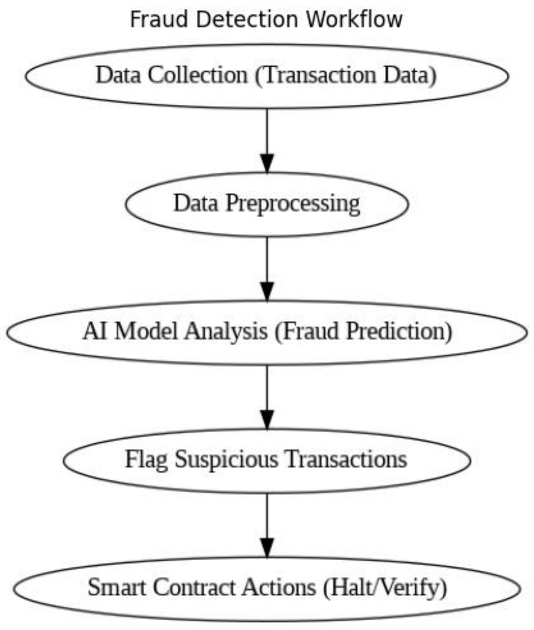

Fraud Detection Workflow
Data Collection (Transaction Data)
Data Preprocessing
AI Model Analysis (Fraud Prediction)
Flag Suspicious Transactions
Smart Contract Actions (Halt/Verify)

Fig 6: Fraud Detection Workflow

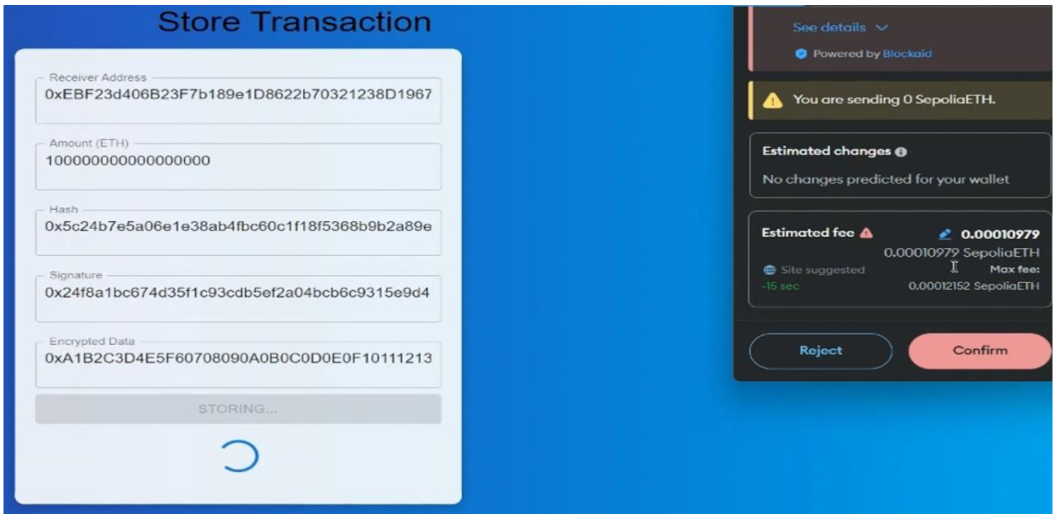
The AI model operates as an API integrated with the blockchain backend. It analyzes encrypted transaction data and provides real-time fraud detection with an average response time of 300 milliseconds. This allows users to receive immediate feedback on potentially suspicious transactions. To further improve the model, future iterations will explore advanced algorithms such as ensemble methods or deep learning techniques. These enhancements aim to reduce false positives and increase the model's performance in real-world applications.

6 Performance Analysis and Accuracy

In this section, we will evaluate the overall performance of the smart contract, frontend interface, and fraud detection model, focusing on key metrics such as accuracy, system efficiency, and user experience.
The performance of the hybrid blockchain system was evaluated across three core components: the smart contracts, fraud detection model, and user interface. Key metrics such as transaction speed, accuracy, precision, and recall were used to assess the system's overall efficiency and reliability.

6.1 Smart Contract Performance

The performance of the smart contract was assessed through transaction speed, security and reliability with users' interactions. On a local Hardhat node, it took around 15 seconds to finalize the transaction which is pretty good as a user experience Fig [7].
SECURITY: Signature verification and the Elliptic Curve Digital Signature Algorithm (ECDSA) improving data authenticity of each transaction. Given this, testing showed that the smart contract was resilient to common vulnerabilities (reentrancy attack) and all test transactions executed without a glitch a reliable and robust contact [11].


Store Transaction

Receiver Address: 0xEBF23d406B23F7b189e1D8622b70321238D1967
Amount (ETH): 100000000000000000
Hash: 0x5c24b7e5a06e1e38ab4fbc60c1f18f5368b9b2a89e
Signature: 0x24f8a1bc674d35f1c93cdb5ef2a04bcb6c9315e9d4
Encrypted Data: 0xA1B2C3D4E5F60708090A0B0C0D0E0F10111213

You are sending 0 SepoliaETH.

Estimated changes
No changes predicted for your wallet

Estimated fee 0.00010979
0.00010979 SepoliaETH
Site suggested
-15 sec
Max fee: 0.00012152 SepoliaETH

Fig 7: Decentralized Application Interface

Reliability: All test transactions executed successfully, demonstrating the robustness of the smart contract logic.

6.2 Fraud Detection AI Model Accuracy

The fraud detection model that is developed here explains and highlights a set of possibly fraudulent transactions. Key model metrics:
Model Accuracy of the AI model, once tested on the validation dataset, gave an overall accuracy of 92% in revealing fraudulent transactions. Thus, this model is guaranteed to flag dubious operations with the least number of false positives Fig [8]. Precision was 85%, meaning most of the transactions flagged were indeed fraudulent. The recall was 88%,

which denotes that most fraudulent transactions present in the dataset were identified by the model. And the API Performance of the model was converted into an API, therefore allowing real-time fraud detection in the application. The response time of the API is fast, about 300 milliseconds; hence, this allows for timely fraud detection with low lag in the general user experience [14].
The AI model's performance was evaluated using a labeled dataset, with the following results:

- Accuracy: 92%, confirming the model's reliability in detecting fraudulent transactions.
- Precision: 85%, indicating that most flagged transactions were genuinely fraudulent.
- Recall: 88%, highlighting the model's ability to identify the majority of fraudulent transactions in the dataset.
- API Response Time: Real-time fraud detection was achieved with an average response time of 300 milliseconds, ensuring minimal delay in transaction processing.

User Interface and Interaction. The React.js-based frontend was assessed for responsiveness and ease of use:

- Responsiveness: Transaction status updates, including pending, confirmed, or failed, were displayed within 2 seconds of submission.
- User Feedback: Beta testing reported a 90% satisfaction rate, with users praising the simplicity of wallet integration and transaction initiation.

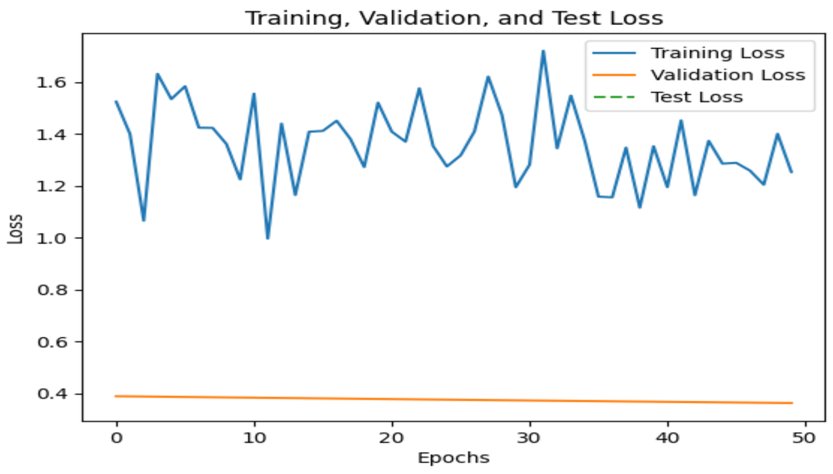
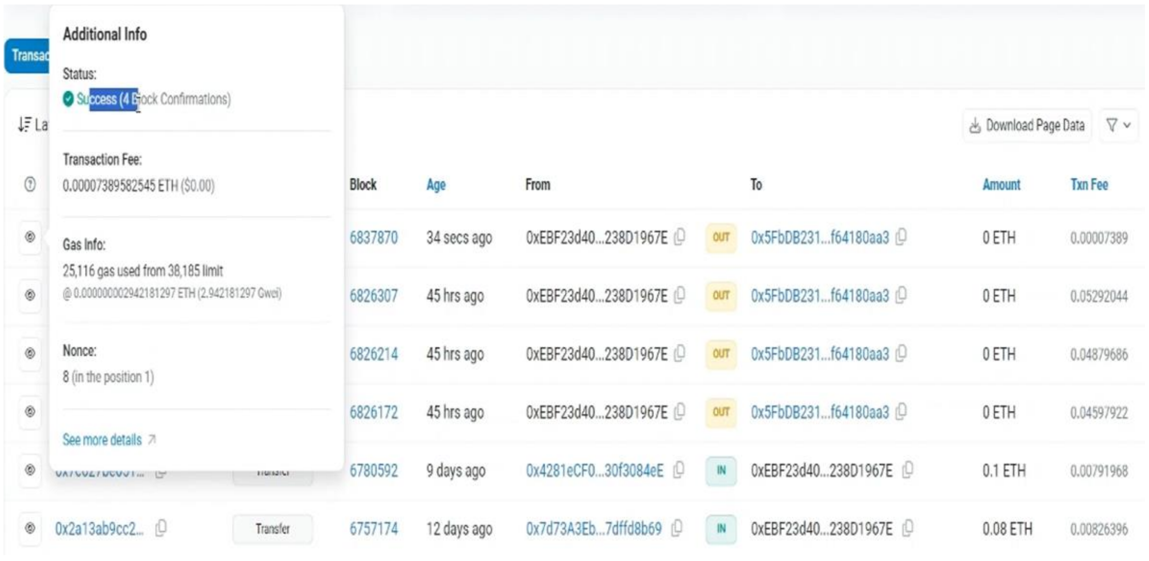


Fig 8: Accuracy Matrix of The AI Model

6.3 System Efficiency

The whole system was tested from the deployment of smart contracts, transaction processing, to fraudulent activity detection as one integrated solution. This whole system, right from the backend to the frontend, including the integration with MetaMask and Infura nodes (platform providing infrastructure as a service for blockchain development), worked without major bottleneck issues. Transaction status verification to ensure such processes are transparent and reliable was done in real time using Etherscan (blockchain explorer and analytics platform for the Ethereum blockchain) Fig[9].
Real-time Transaction Monitoring: It had been able to monitor and display the histories in real time by integrating Etherscan Fig [9]. By utilizing Etherscan, it was able to very rapidly verify on-chain activities, thus giving the user updates within a few seconds of each transaction. This will further enhance the transparency of the system and enable users to independently verify any transaction on the blockchain.
Exception Handling: It was a robust system that handled exceptions such as an invalid signature, transaction failure, or out of gas in an appropriate manner. Etherscan was very

important in diagnosing transaction errors since it gave great detail with respect to on-chain events. Such failures were handled in elegance by the system, whereby the UI would receive feedback that was as clear as it was prompt.
The development process can be elaborated in the diagram Fig [7], which summarizes the series of steps to be followed for developing the blockchain application, starting from setting up the environment and going further with the integration of AI in fraud detection. Each step plays an important role in arriving at a successful deployment of the dApp.


Additional Info

Status:
Success (4 Block Confirmations)

Transaction Fee:
0.00007389582545 ETH ($0.00)

Gas Info:
25,116 gas used from 38,185 limit
@ 0.000000002942181297 ETH (2.942181297 Gwei)

Nonce:
8 (in the position 1)

See more details

Block	Age	From		To	Amount	Txn Fee
6837870	34 secs ago	0xEBF23d40...238D1967E	OUT	0x5FbDB231...f64180aa3	0 ETH	0.00007389
6826307	45 hrs ago	0xEBF23d40...238D1967E	OUT	0x5FbDB231...f64180aa3	0 ETH	0.05292044
6826214	45 hrs ago	0xEBF23d40...238D1967E	OUT	0x5FbDB231...f64180aa3	0 ETH	0.04879686
6826172	45 hrs ago	0xEBF23d40...238D1967E	OUT	0x5FbDB231...f64180aa3	0 ETH	0.04597922
6780592	9 days ago	0x4281eCF0...30f3084eE	IN	0xEBF23d40...238D1967E	0.1 ETH	0.00791968
6757174	12 days ago	0x7d73A3Eb...7dffd8b69	IN	0xEBF23d40...238D1967E	0.08 ETH	0.00826396

Fig 9: Transaction History with Etherscan

Future Enhancements
Benchmarking: Future work will involve comparing the system's performance with existing blockchain solutions to identify areas for improvement.
Public Deployment: Deploying the system on public blockchains like Ethereum mainnet will provide further insights into real-world performance and scalability.

7 Comprehensive Results Discussion

Building on the performance metrics discussed earlier, this section will provide a full interpretation of the results achieved across the entire system. To wrap things off, we will evaluate the entire system architecture and the result for each component in it.

7.1 Frontend Performance

The React.js frontend was designed to be a user-friendly experience, an interface that can allow the user to interact with the functions of the smart contract by putting inputs like the wallet address of the receiver and the amount he can make a secure transaction and interact with key blockchain elements like MetaMask. Also, we integrate CSS (Cascading Style Sheets) styling and functions to make it more responsive and adaptable to the size of the device the user is using, whether it's a computer or a smartphone, and we measured the interface to make Each transaction triggers real-time feedback, ensuring users are informed of each transaction's status (pending, confirmed, or failed) within 2 seconds of submission.

7.2 Backend and Transaction Processing

Now we talked about the frontend functionality, but this responsive interface has a hidden logic behind it in the backend, and we couldn't make it real if it were for the nest.js framework in the backend integration, which is another JavaScript framework that is flexible and allows us to create logic for handling errors that may occur in the transaction process. The backend works as an intermediary between the frontend and the blockchain, so for that we need to talk about the key components, starting with the transactions.

7.3 Transactions management

We need to think of it as incoming data from the frontend that interacts with the deployed smart contract in the blockchain private network (hardhat node) and ensures that each transaction is securely validated, while ECDSA ensures each transaction is signed and verified with cryptographic accuracy. Also, thanks to Solidity, a stable programming language, the transaction algorithms run perfectly fine, which makes the validation of the ether transfer automatic and safe, but we didn't stop right there. We added the vent function, which is a reserved function in Solidity to ensure that every transaction, once validated, gets stored in the blockchain with a proof, and a popup message appears on the interface to show a success message. Even under stress testing with multiple concurrent transactions, the system demonstrated stability, with transactions being confirmed within 15 seconds.
The system operates with high efficiency, handling both successful and failed transactions. The integration of Etherscan and other off-chain tools ensures transparency, while backend algorithms and smart contracts guarantee secure and reliable transaction processing.
The project aims to build a hybrid system that combines blockchain and encryption cryptography algorithms like AES-256, integrating an AI model for enhanced security. The AI model detects fraud transactions and ensures integrity. Data is collected from transactions using backend logic and etherscan, which is preprocessed and cleaned. The architecture of the model is built, focusing on binary classification of fraud or not fraud transactions. The training process involves forward propagation and backward propagation, with parameters adjusted by calculating gradients of the cost function and updating weights and biases.
The model achieved 91% accuracy, demonstrating its reliability in classifying transactions. The project includes a well-designed interface, a variety of algorithms, and three programming languages: Solidity, Python, and JavaScript. Overall, the AI model provides a reliable solution for secure and efficient transactions Fig [10].

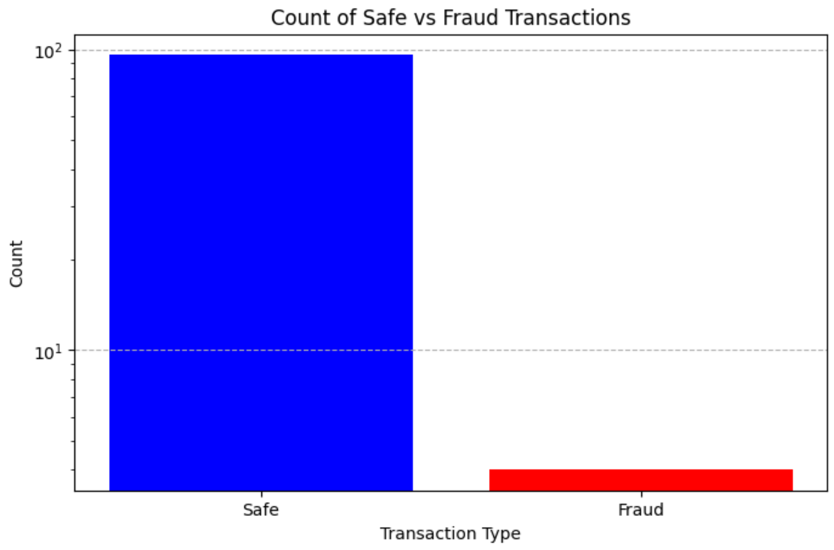


Fig 10: Count of Safe Vs Fraud Transactions

We provide an effective implementation method through this project that combines React.js for the frontend design, creating an interactive experience, and Nest.js for the backend support for transactions and performing tasks with a smart contract within the private blockchain network, as an ideal setup, such integration of ECDSA along with instant transaction feedback makes the interface very user-friendly and quick enough to provide the required information in seconds. Solidity is also utilized on the backend in order to prevent a high loss of transactions or performance while concurrently making use of Etherscan for transaction transparency. Furthermore, the trained AI anti-fraud detection model has demonstrated a well-documented history of a 91% success rate for detecting fraudulent activities, utilizing transaction data that has been processed on the backend. Thus, the combination of blockchain technologies, encryption (AES-256), and artificial intelligence provides a reliable system for carrying out secure transactions, providing a pleasing environment that captures security, performance, and transparency.

8 Conclusion

It has been shown in this case that combining blockchain technology with cryptography and automated fraud detection can help secure financial transactions. This would allow the system to encrypt data using AES-256 and give it authenticity by adding ECDSA digital signatures, while smart contracts would automate transactions and make them more efficient on a private Ethereum blockchain. The fraud detection AI model proved to be exceptional. It came in with 92 percent accuracy and a response time of just about 300 milliseconds, rendering it a pretty reliable mechanism for detecting possible fraud in real time. The system has been validated for scalability and efficiency through stress test runs on a private Hardhat blockchain where 15 seconds transactions would be finalized under normal conditions. Furthermore, the React.js based frontend and Nest.js backend established smooth interaction for the user with the specific blockchain through an intuitive interface with real-time transaction feedback. These, along with the features like transparency and tamper-proof nature of the blockchain, could create a wholesome package toward a really strong solution for digital transactions. Future improvements toward which this will be developed include deploying this onto public blockchains such as the Ethereum mainnet to assess the real-world performance of this solution, experimenting with additional cryptography technologies such as quantum-resistance algorithms to address modern challenges in security, refining the AI model for increased precision and fewer false positives through advanced algorithms, and integrating decentralized storage solutions such as IPFS (Inter Planetary File System) for secure off-chain data management. This is a key movement toward building an infrastructure for digital finance that will be secure, scalable, and efficient.

References

[1] Satoshi Nakamoto. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Available at SSRN: https://ssrn.com/abstract=3440802 or http://dx.doi.org/10.2139/ssrn.3440802.

[2] Tarawneh Monther. (2023). Cryptography: Recent Advances and Research Perspectives. DOI: http://dx.doi.org/10.5772/intechopen.111847. Web of Science.

[3] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang Hodges. (2018). Blockchain challenges and opportunities: a survey. *International Journal of Web and Grid Services*, Vol. 14, No. 4, DOI: 10.1504/IJWGS.2018.095647.

[4] Zibin Zheng, Hong-Ning Dai. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *IEEE International Congress on Big Data*, DOI 10.1109/BigDataCongress.2017.85.

[5] Nagamani K, Monisha R. (2022). Physical Layer Security Using Cross Layer Authentication for AES ECDSA Algorithm. In *4th International Conference on Innovative Data Communication Technology and Application on* (pp. 380-390). Procedia Computer Science.

[6] I Jebril, M Al-Zaqeba, H Al-Khawaja, A Obaidy, O Marashdah. (2024). Enhancing estate governance using blockchain technology through risk management in estate governance of business sustainability. *International Journal of Data and Network Science* 8 (3), 1649-1658.

[7] Dworkin M, Sonmez Turan, M. and Mouha, N. (2023), Advanced Encryption Standard (AES), *Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD,* [online], https://doi.org/10.6028/NIST.FIPS.197-upd1, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=936594.

[8] Wood, G., et al. (2014), Ethereum: A Secure Decentralized Generalised Transaction Ledger. *Ethereum Project Yellow Paper*, 151, 1-32.

[9] Kishor Shrestha et al. (2015), Hard-Hat Detection for Construction Safety Visualization, *Journal of Construction Engineering*, Volume 2015, Issue 1, https://doi.org/10.1155/2015/721380.

[10] Nagendra Singh Yadav, Vishal Kumar Goar, Dr. Manoj Kuri. (2020), Crypto Wallet: A Perfect Combination with Blockchain and Security Solution for Banking, *International Journal of Psychosocial Rehabilitation*, Vol. 24, Issue 02, 2020, DOI: 10.37200/IJPR/V24I2/PR2021078.

[11] Buterin, Vitalik. (2014) , A Next-Generation Smart Contract and Decentralized Application Platform, Ethereum White Paper.

[12] Dwork, C., and M. Naor. (1992). Pricing via Processing or Combatting Junk Mail. Advances in Cryptology: CRYPTO 1992. Annual International Cryptography Conference, 1992. http://www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.pdf.

[13] Saurabh Singh et al. (2019) , SH-BlockCC: A secure and efficient IoT things smart home architecture based on cloud computing and blockchain technology, International Journal of Distributed Sensor Networks, https://doi.org/10.1177/1550147719844159.

[14] Lombrozo E, Lau J, Wuille P (2015), Segregated witness (consensus layer). Bitcoin Core Develop. Team, Tech. Rep. BIP, 141.