SECURING DATA
A COMPREHENSIVE APPROACH FOR ENHANCING DATA SECURITY IN DECENTRALIZED ENVIRONMENTS WITH AES-256-GCM, IPFS BLOCKCHAIN, AND MACHINE LEARNING

A Main Project submitted
in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Submitted by

1. K. Priyanka (20PA1A0567) **2. M. Sai Venkat Ganesh (20PA1A0592)**

3. P.Aharon Poul (20PA1A05C2) **4. M.Lakshmi Narayana (20PA1A05A7)**

Under the esteemed guidance of
CH. Lakshmi Veenadhari
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VISHNU INSTITUTE OF TECHNOLOGY
(Autonomous)
(Approved by AICTE, Accredited by NBA & NAAC and permanently affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202
2023 – 2024

VISHNU INSTITUTE OF TECHNOLOGY

(Autonomous)

(Approved by AICTE, accredited by NBA & NAAC, and permanently affiliated to JNTU Kakinada)

BHIMAVARAM-534202

2023-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "EVENT MANAGEMENT SYSTEM", is being submitted by ***K.PRIYANKA, M.SAI VENKAT GANESH, P.AHARON POUL AND M.LAKSHMI NARAYANA***, bearing the **REGD.NOS: 20PA1A0567, 20PA1A0592, 20PA1A05C2 and 20PA1A05A7** submitted in fulfilment for the award of the degree of **"BACHELOR OF TECHNOLOGY"** in **"COMPUTER SCIENCE AND ENGINEERING"** is a record of work carried out by them under my guidance and supervision during the academic year 2023-2024 and it has been found worthy of acceptance according to the requirements of university.

Internal Guide
CH. Lakshmi Veenadhari

Head of the Department
Dr. Sumit Gupta

External Examiner

ACKNOWLEDGEMENT

It is natural and inevitable that the thoughts and ideas of other people tend to drift into the subconscious due to various human parameters, where one feels the need to acknowledge the help and guidance derived from others. We express our gratitude to those who have contributed to the fulfillment of this project.

We take the opportunity to extend our sincere thanks to **Dr. D. Suryanaryana**, the director of VIT, Bhimavaram, whose guidance from time to time helped us complete this project successfully.

We also express our sincere gratitude to **Dr. M. Venu**, the principal of VIT, Bhimavaram, for his continuous support in helping us complete the project on time.

We are thankful to **Dr. Sumit Gupta**, Head of the Department of Computer Science and Engineering, for his continuous and unwavering support and guidance. We acknowledge our gratitude for his valuable guidance and support extended to us from the conception of the idea to thecompletion of this project.

We are very thankful to **CH. Lakshmi Veenadhari**, Assistant Professor, our internal guide, whose guidance from time to time helped us complete this project successfully.

Project Associates

K.Priyanka (20PA1A0567)

M.Sai Venkat Ganesh (20PA1A0592)

P.Aharon Poul (20PA1A05C2)

M.Lakshmi Narayana (20PA1A05A7)

ABSTRACT

Our project addresses the contemporary challenge of securing cloud infrastructure by proposing a comprehensive solution. Utilizing the AES-256-GCM algorithm with a 256-bit key and Galois/Counter Mode ensures robust encryption for data security in the dynamic cloud environment. An advanced attack detection mechanism, employing a decision tree classifier and machine learning models, proactively identifies and responds to potential threats in real-time, bolstering overall security.Additionally, we leverage the InterPlanetary File System (IPFS) blockchain platform to upload encrypted files, generating unique hash values for enhanced data integrity. The integration with IPFS provides a decentralized and tamper-resistant environment for secure storage. Machine learning-based hash value classification further enhances security, dynamically analyzing and categorizing hashvalues to determine potential attacks on encrypted files. This synergy between AES-256-GCM, IPFS blockchain, and machine learning creates a comprehensive and adaptive solution to evolving cybersecurity challenges within cloud environments.

Keywords: AES-256-GCM, encryption, Attack detection mechanism, Decision tree classifier, Machine learning models, InterPlanetary File System (IPFS) blockchain, Tamper-resistant environment, cybersecurity challenges.

TABLE OF CONTENTS

LIST OF DIAGRAMS

1.INTRODUCTION

In the rapidly advancing landscape of cloud computing, our project stands as a beacon of innovation, offering a comprehensive solution to the critical challenge of securing cloud infrastructure. At its core, our approach centre’s around the formidable AES-256-GCM algorithm, utilizing a 256-bit key and Galois/Counter Mode to establish robust encryption within the dynamic and intricate cloud environment. Going beyond conventional security measures, our project incorporates an advanced attack detection mechanism. This cutting-edge system, powered by a decision tree classifier and machine learning models, proactively identifies and responds to potential threats in real-time, significantly enhancing the overall security posture. In tandem, we harness the Interplanetary File System (IPFS) blockchain platform to upload encrypted files, generating unique hash values that not only fortify data integrity but also establish a decentralized and tamper-resistant storage environment.

The symbiosis of AES-256-GCM, IPFS blockchain, and machine learning-based hash value classification forms the crux of our project's strength. This synergy creates a comprehensive and adaptive security solution, dynamically analysing and categorizing hash values to discern potential attacks on encrypted files. In essence, our project offers a forward-thinking and resilient defence against the evolving cybersecurity challenges within the intricate realm of cloud environments. For businesses making the transition to the cloud, robust cloud security is imperative. Security threats are constantly evolving and becoming more sophisticated, and cloud computing is no less at risk than an on-premise environment. For this reason, it is essential to work with a cloud provider that offers best-in-class security that has been customized for your infrastructure. One of the benefits of utilizing cloud storage and security is that it eliminates the need to invest in dedicated hardware. Not only does this reduce capital expenditure, but it also reduces administrative overheads. Where once IT teams were firefighting security issues reactively, cloud security delivers proactive security features that offer protection 24/7 with little or no human intervention.

Cloud computing services offer the ultimate in dependability. With the right cloud security measures in place, users can safely access data and applications within the cloud no matter where they are or what device they are using. Cloud data security becomes increasingly important as we move our devices, data centre’s, business processes, and more to the cloud. Ensuring quality cloud data security is achieved through comprehensive security policies, an organizational culture of security, and cloud security solutions. Selecting the right cloud security solution for your business is imperative if you want to get the best from the cloud and ensure your organization is protected from unauthorized access, data breaches and other threats.

1.1 OBJECTIVE

The objective of our project, titled "An Efficient AES-Based Cloud User Data Security with Attack Detection Mechanism," is to enhance file security in the cloud, focusing on safeguarding data privacy and preventing theft. Developed using Python libraries such as PyCryptodome, scikit-learn, NumPy, Pandas, as well as additional ones like binascii, os, and script, our solution comprises two main components:

1. Cryptography - Encryption and Decryption:

- **Encryption:** In this phase, we employ cryptographic algorithms to encrypt the contents of files stored in the cloud, ensuring the confidentiality and integrity of user data.
- **Decryption:** The corresponding decryption phase allows authorized users to securely retrieve and access the original contents of the files.

2. Attack Detection Mechanism:

Machine Learning Algorithms: The second part of our project focuses on implementing a robust attack detection mechanism utilizing machine learning algorithms. Specifically, we leverage a decision tree classifier to proactively identify and respond to potential security threats in real-time. In addition to these core functionalities, our project integrates the Interplanetary File System (IPFS) blockchain platform. This integration involves uploading the encrypted files to generate unique hash values. Within the IPFS blockchain environment, we classify these hash values using machine learning models. This classification helps determine whether the encrypted files are under attack, providing an additional layer of security. By combining efficient AES-based cryptography with a proactive attack detection mechanism and leveraging IPFS blockchain for secure file storage and hash value classification, our project aims to offer a comprehensive solution for ensuring the utmost security of user data in cloud environments.

2.SYSTEM ANALYSIS

2.1 Hardware and software requirements

Software requirements

- **Ethereum:** Ethereum Client (e.g., Geth, Besu) for interacting with the Ethereum blockchain.
- **Remix:** Remix IDE for Ethereum smart contract development and testing.
- **MetaMask:** MetaMask browser extension for interacting with Ethereum decentralized applications
- **Solidity Compiler:** Solidity Compiler for compiling smart contracts written in the Solidity programming language.
- **IPFS Environment:** IPFS software for setting up and managing the Interplanetary File System environment.
- **Node.js:** Node.js for server-side JavaScript runtime, often used in blockchain development.
- **Web3:** Web3.js library for interacting with Ethereum nodes and smart contracts using JavaScript.
- **Python:** Python (already mentioned) for implementing other project functionalities.
- **Web Browser:** A modern web browser compatible with MetaMask and other web-based components.
- **Text Editor or IDE:** A text editor or integrated development environment (IDE) for Ethereum smart contract and general code editing.
- **Google colab:** A google colab allows anybody to write and execute arbitrary python code through the browser and is well suited to machine learning.

Hardware requirements

Server Infrastructure:

- Sufficient processing power and memory capacity to handle encryption/decryption operations, machine learning tasks, and blockchain transactions.
- Multi-core processors (at least quad-core) for parallel processing.
- Adequate RAM (at least 16GB) to support concurrent tasks.

Storage:

- High-capacity storage for encrypted files and data sets.
- Fast and reliable storage mediums to ensure efficient data retrieval and processing.

Network Infrastructure:

- High-speed internet connection for seamless communication with cloud services.
- Reliable networking equipment to facilitate smooth data transfer.

2.2 Existing System

There are many cloud service providers like AWS Cloud(Amazon Web Services), Microsoft Azure, GCP(Google Cloud Platform), VMware Cloud, IBM Cloud and so on, which use cryptographic algorithms for safe and secure data storage. These applications- utilise manifold cryptographic algorithms like DES, 3- DES, AES, RSA etc.

Drawbacks of the existing system

➢ No attack detection mechanism

➢ Data leakage

➢ Insecure interfaces

➢ Data loss or theft

➢ Technical vulnerabilities, especially on shared environments

2.3 Proposed System

Our innovative system fortifies cloud security through AES-256-GCM encryption, ensuring robust data protection. An advanced attack detection mechanism, employing a decision tree classifier and machine learning models, proactively identifies and responds to potential threats in real-time. Leveraging IPFS blockchain guarantees secure file storage with unique hash values, creating a decentralized and tamper-resistant environment. Machine learning-based hash value classification dynamically analyses and categorizes hashes, adding an extra layer of security against potential attacks on encrypted files. The synergy between AES-256-GCM, IPFS blockchain, and machine learning forms a comprehensive solution, effectively addressing cloud security challenges.

Objectives of proposed system

- **Dynamic Adaptability:** The system's dynamic configuration allows for adaptive adjustments, ensuring it effectively responds to the evolving nature of cybersecurity threats in cloud environments.
- **Comprehensive Data Integrity:** The combination of IPFS blockchain and machine learning-based hash value classification ensures thorough data integrity assurance, detecting and preventing unauthorized modifications.
- **Multi-Layered Security:** Employing a multi-layered security approach, the system combines encryption, proactive threat detection, and blockchain-backed storage for a robust defence against various attack vectors.
- **Real-time Threat Response:** The integration of advanced mechanisms enables real-time responses to potential threats, enhancing overall security posture and reducing the window of vulnerability.
- **Reduced Centralized Risks:** Decentralized storage through IPFS lessens risks associated with single points of failure, enhancing system resilience against potential breaches.
- **Holistic Defence Strategy:** The synergy between encryption, attack detection, and blockchain technologies creates a holistic defence strategy, covering multiple facets of cloud security.
- **Adaptive Security Measures:** The system's adaptability to evolving cybersecurity challenges positions it as an effective solution for addressing contemporary threats within cloud environments.

2.4 Feasibility study

A feasibility study is an analysis that takes all a project's relevant factors into account including economic, technical, legal, and scheduling considerations to ascertain the likelihood of completing the project successfully. A feasibility study is important and essential to evolute any proposed project is feasible or not. A feasibility study is simply an assessment of the practicality of a proposed plan or project.

To determine if the product is technically and financially feasible to develop, is the main aim of the feasibility study activity. A feasibility study should provide management with enough information to decide:

Whether the project can be done.

• To determine how successful your proposed action will be.

• Whether the final product will benefit its intended users.

• To describe the nature and complexity of the project.

• What are the alternatives among which a solution will be chosen

• To analyse if the software meets organizational requirements. There are various types of feasibility that can be determined. They are:

Operational : Define the urgency of the problem and the acceptability of any solution, includes

people-oriented and social issues: internal issues, such as manpower problems, labour objections, manager resistance, organizational conflicts, and policies; also, external issues, including social acceptability, legal aspects, and government regulations.

Technical: Is the feasibility within the limits of current technology? Does the technology exist at all? Is it available within a given resource?

Economic: Is the project possible, given resource constraints? Are the benefits that will accrue from the new system worth the costs? What are the savings that will result from the system, including tangible and intangible ones? What are the development and operational costs?

Schedule: Constraints on the project schedule and whether they could be reasonably met

2.4.1 Operational feasibility

Operational feasibility is a measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The operational feasibility refers to the availability of the operational resources needed to extend research results beyond on which they were developed and for which all the operational requirements are minimal and easily accommodated. In addition, the operational feasibility would include any rational compromises farmers make in adjusting the technology to the limited operational resources available to them. The operational Feasibility also perform the tasks like Does the current mode of operation provide adequate response time?

• Does the current of operation make maximum use of resources.

• Determines whether the solution suggested by the software development team is acceptable.

• Does the operation offer an effective way to control the data?

Our project operates with a processor and packages installed are supported by the system.

2.4.2 Economic feasibility

Economic analysis could also be referred to as cost/benefit analysis. It is the most frequently used method for evaluating the effectiveness of a new system. In economic analysis the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. Economic feasibility study related to price, and all kinds of expenditure related to the scheme before the project starts. This study also improves project reliability. It is also helpful for the decision-makers to decide the planned scheme processed latter or now, depending on the financial condition of the organization. This evaluation process also studies the price benefits of the proposed scheme. Economic Feasibility also performs the following tasks.

• Cost of packaged software/ software development.

• Cost of doing full system study.

• Is the system cost Effective?

2.4.3 Technical feasibility

A large part of determining resources has to do with assessing technical feasibility. It considers the technical requirements of the proposed project. The technical requirements are then compared to the technical capability of the organization. The systems project is considered technically feasible if the internal technical capability is sufficient to support the project requirements. The analyst must find out whether current technical resources can be where the expertise of system analysts is beneficial, since using their own experience and their contact with vendors they will be able to answer the question of technical feasibility. Technical Feasibility also performs the following tasks.

• Is the technology available within the given resource constraints?

• Is the technology have the capacity to handle the solution

• Determines whether the relevant technology is stable and established.

• Is the technology chosen for software development has a large number of users so that they can be consulted when problems arise, or improvements are required.

3.SYSTEM DESIGN

3.1 Data flow diagram

Definition: A model is an abstract representation of the system, constructed to understand the system's priority in building or modifying it. A model is a simplified representation of reality and it provides a means for conceptualization and communication of ideas in a precise and ambiguous form. We build models so that we can better understand the system we are developing. The elements are like components that can be associated in different ways to make a complete UML picture, which is known as a diagram. Thus, it is essential to understand the different diagrams to implement the knowledge in real-life systems. UML (Unified Modelling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It is a method for describing the system architecture in detail using the blueprint. We use UML diagrams to portray the behaviour and structure of a system. This is the step while developing any product after analysis. The goal of this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities that are to be used in the product being developed needs to be designed

There are various kinds of methods in software design:

- Use case Diagram
- Sequence Diagram
- Class Diagram
- Activity Diagram
- State Chart Diagram.

3.2 Building blocks of UML diagram

The vocabulary of the UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

Things in UML:

There are four kinds of things in the UML:

- Structural things
- Behavioural things
- Grouping things
- Annotational things

These things are the basic object-oriented building blocks of the UML. You use them to write well-formed models.

Structural Things

Structural things are the nouns of UML models. These are the mostly static parts of a model, representing elements that are either conceptual or physical. Collectively, the structural things are called classifiers. A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics. A class implements one or more interfaces. Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations.

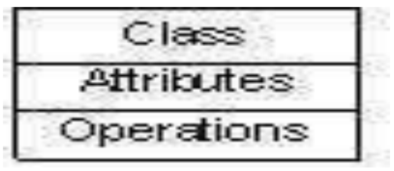
Class diagram

In our system's class diagram, several key classes play pivotal roles in ensuring the functionality and security of the application. The Encryption Manager class oversees the encryption and decryption processes of files within the system. It possesses attributes such as encryption Key and encryption Algorithm, crucial for ensuring the security and integrity of encrypted data. Methods like encryptFile and decryptFile facilitate the encryption and decryption operations, respectively, providing essential functionality for securing sensitive data.

The BlockchainIntegration class manages the integration with the IPFS blockchain, a critical aspect of our decentralized storage solution. This class holds attributes like blockchainNodeURL and contractAddress, enabling communication with the blockchain network. Methods like uploadToIPFS and getHashValue facilitate file uploads to the IPFS network and retrieval of hash values associated with uploaded files, ensuring data integrity and tamper resistance.

Our system also incorporates the MachineLearningModel class, responsible for hash value classification using machine learning techniques. This class maintains attributes like modelType and trainedData, essential for configuring and training the machine learning model. The classifyHashValue method analyzes hash values to determine potential attacks on encrypted files, providing an additional layer of security through proactive threat detection.

Lastly, the FileUploader class handles file uploading operations within the system. It possesses attributes like uploadURL, specifying the destination for file uploads. The uploadFile method facilitates the uploading of files to designated locations, ensuring seamless integration with other components of the system.Together, these classes form the backbone of our system, encompassing essential functionalities for data encryption, decentralized storage, threat detection, and file management. Through their coordinated efforts, our system provides a comprehensive approach to enhancing data security in decentralized environments, leveraging encryption, blockchain technology, machine learning, and seamless file management capabilities.



Sequence diagram

The sequence diagram for encrypting and uploading a file to the IPFS blockchain illustrates the sequential interactions between various components involved in the process. Initially, the process begins with a user initiating the file encryption operation by interacting with the encryption manager component. Upon receiving the request, the encryption manager invokes the encryptFile method, passing the file to be encrypted as a parameter. Subsequently, the encryption manager employs the AES-256-GCM encryption algorithm to encrypt the file, utilizing the encryption key and algorithm specified. Once the encryption is completed, the encrypted file is returned to the encryption manager.Next, the encrypted file is passed to the blockchain integration module, which facilitates its upload to the IPFS blockchain. The blockchain integration module initiates the upload process by invoking the uploadToIPFS method, which sends the encrypted file to the IPFS blockchain network. During this interaction, the blockchain integration module communicates with the IPFS blockchain nodes, providing the necessary authentication and authorization credentials for the upload process. Upon successful completion of the upload, the IPFS blockchain generates a unique hash value for the encrypted file and returns it to the blockchain integration module.

Finally, the blockchain integration module communicates the hash value back to the encryption manager, which then passes it to the file uploader component. The file uploader component utilizes the hash value to categorize and organize the encrypted file within the IPFS blockchain network. This completes the sequential process of encrypting and uploading a file to the IPFS blockchain, demonstrating the coordinated interactions between the encryption manager, blockchain integration module, and file uploader component in ensuring data security and integrity within the decentralized environment.

Activity diagram

The activity diagram for the system's workflow from file encryption to uploading to the IPFS blockchain and subsequent hash value classification with machine learning models provides a high-level overview of the system's activities. Initially, the process begins with the "Encrypt File" activity, where the file encryption operation is initiated. This activity triggers the "Encryption" process, wherein the file is encrypted using the AES-256-GCM encryption algorithm by the EncryptionManager component. Upon successful encryption, the process transitions to the "Upload Encrypted File" activity, initiating the upload process to the IPFS blockchain. The "Blockchain Integration" process manages this operation, communicating with the IPFS blockchain to upload the encrypted file securely.

Once the encrypted file is successfully uploaded to the IPFS blockchain, the process proceeds to the "Classify Hash Value" activity. This activity involves the classification of the hash value associated with the uploaded file using machine learning models. The "Machine Learning Classification" process handles this task, analyzing the hash value to determine potential attacks on the encrypted file. Based on the classification results, the process may transition to additional activities, such as "Take Action on Threat" if a potential attack is detected, or "Secure File Storage" if the file is deemed secure.Throughout the workflow, the activities are interconnected through transitions, representing the sequential flow of operations within the system. The activity diagram provides a comprehensive visualization of the system's activities and workflows, offering insights into the overall process flow from file encryption to hash value classification, and facilitating a better understanding of the system's functionality and interactions.

Use case diagram

The use case diagram for our system "SecureData" identifies the key functionalities or use cases along with the actors involved in the system. The primary actors in the system are the "User" and the "System." The "User" initiates various actions within the system, while the "System" encompasses all the components and processes that execute these actions.One of the essential use cases depicted in the diagram is "Encrypt File," where the user encrypts a file for secure transmission or storage. This action involves interactions with the "EncryptionManager" component. Another crucial use case is "Upload Encrypted File to IPFS," where the user uploads the encrypted file to the IPFS blockchain network for decentralized storage. This interaction is facilitated by the "BlockchainIntegration" component.

Additionally, the use case "Classify Hash Value" involves the classification of hash values associated with uploaded files using machine learning models. This action is performed by the "MachineLearningModel" component. Other potential use cases may include "Retrieve Encrypted File," "Decrypt File," and "Take Action on Threat" for handling potential security threats.The use case diagram provides a comprehensive overview of the system's functionalities and the interactions between actors and components. It serves as a valuable tool for understanding the system's requirements and defining the scope of its functionality from a user's perspective.

State diagram

The state diagram for our system illustrates the different states and transitions of key components such as the encryption manager, blockchain integration module, and machine learning model throughout their lifecycle. For the encryption manager, states may include "Idle," "Encrypting," and "Decryption." The manager transitions from the Idle state to the Encrypting state when it receives a request to encrypt a file, and then to the Decryption state when decrypting a file. Transitions between these states occur based on user actions or system events triggering encryption or decryption processes.Similarly, the blockchain integration module undergoes states such as "Connecting," "Uploading," and "Complete." It starts in the Connecting state, establishing connections with the IPFS blockchain network. Upon successful connection, it moves to the Uploading state to upload files to the blockchain. Finally, it transitions to the Complete state when the upload process is finished. Transitions between states occur as the module interacts with the blockchain network and completes upload operations.

The machine learning model component may have states such as "Training," "Ready," and "Classifying." Initially, it starts in the Training state, where it trains on available data. Once training is complete, it transitions to the Ready state, indicating readiness for classification tasks. During classification tasks, it enters the Classifying state, where it analyzes hash values to determine potential threats. Transitions between states occur based on completion of training, availability of data, and initiation of classification tasks.

Overall, the state diagram provides a detailed depiction of the lifecycle of key components in the system, illustrating their various states and transitions as they perform encryption, blockchain integration, and machine learning tasks. This visualization aids in understanding the behavior and interactions of components within the system.

Component diagram

The component diagram for our system "SecureData" provides a comprehensive overview of the physical components or modules and their dependencies. At the core of the system, we have several key components, including the Encryption Module, Blockchain Module, and Machine Learning Module. The Encryption Module is responsible for encrypting and decrypting files using the AES-256-GCM encryption algorithm. It encapsulates functionalities related to file encryption and decryption and interacts with other modules for seamless data security.The Blockchain Module manages interactions with the IPFS blockchain network for secure file storage and retrieval. It handles file uploads, hash value generation, and communication with the IPFS blockchain nodes. This module ensures the integrity and tamper resistance of uploaded files by leveraging blockchain technology. Additionally, the Machine Learning Module plays a crucial role in hash value classification for threat detection. It utilizes machine learning algorithms to analyze hash values and identify potential security threats, enhancing the system's proactive defense capabilities.

These core components are interconnected through dependencies, representing the relationships and interactions between them. For example, the Encryption Module may depend on the Blockchain Module for secure file storage, while the Machine Learning Module may depend on both the Encryption Module and the Blockchain Module for accessing encrypted files and hash values. The component diagram provides a visual representation of the system's architecture, illustrating the physical components/modules and their dependencies, which is essential for understanding the system's structure and design.

3.3 Use case diagram

A Use Case Diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. The main purpose of a use case diagram is to show what system functions are performed for which actor.

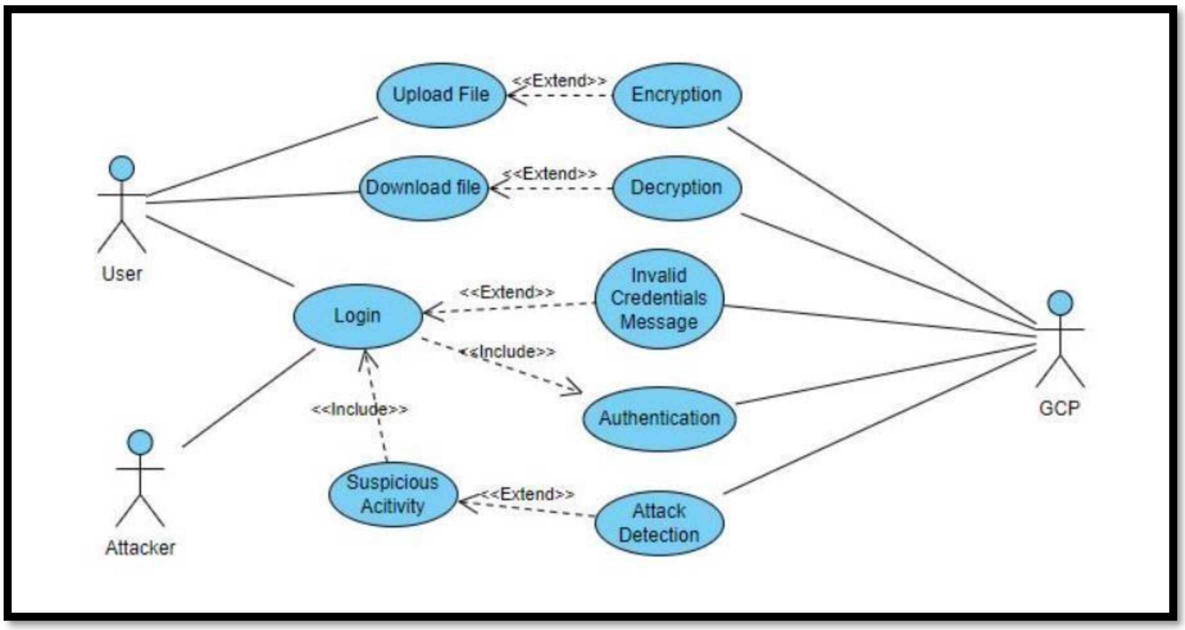


Fig 3.3.1 Use case diagram

3.4 Flow chart

Flow chart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

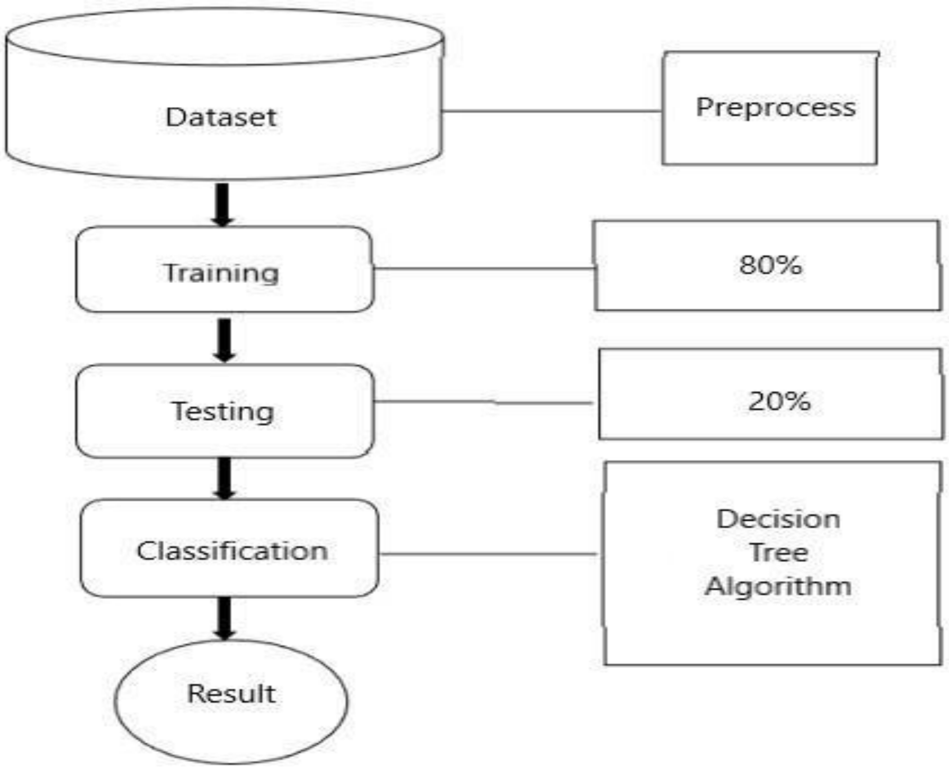


Fig 3.4.1 Attack Detection Mechanism

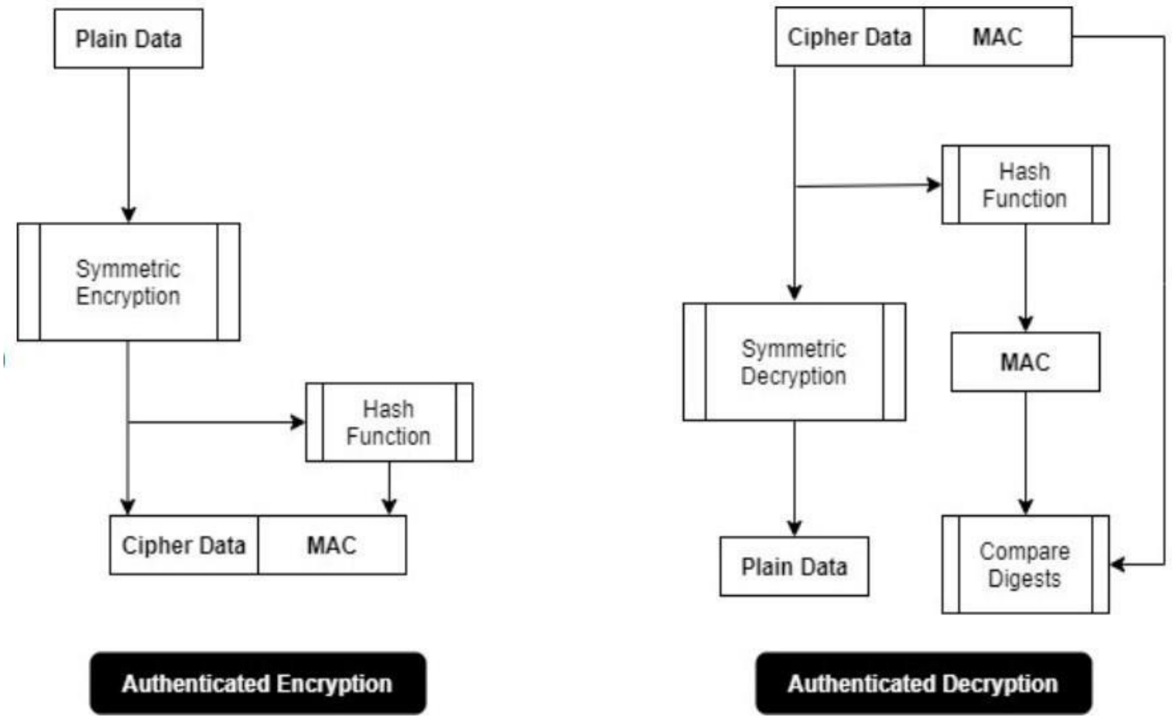


Fig 3.4.2 Cryptography

4.MODULER DESCRIPTIONS

There are three different modules in our project that are useful for categorizing the implementation part of the project.

- AES-256-GCM Encryption
- IPFS Blockchain Integration
- Hash Value Classification
- MetaMask Integration
- Attack Detection

4.1 AES-256-GCM Encryption

The AES Algorithm

The AES Encryption algorithm (also known as the Rijndael algorithm) is a symmetric block cipher algorithm with a block/chunk size of 128 bits. It converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the ciphertext. Basically, encrypting a large input data works like this: the encryption algorithm state is initialized (using the encryption key + a random salt), then the first portion of data (e.g. a block or part of block) is encrypted, then the encryption state is transformed (using the encryption key and other parameters), then the next portion is encrypted, then the encryption state is transformed again and the next portion is then encrypted and so on, until all the input data is processed. The decryption works in a very similar way.

AES-256-GCM

AES-256-GCM is the AES cipher with a 256-bit encryption key and GCM block mode. The GCM mode uses a counter, which is increased for each block and calculated a message authentication tag (MAC code) after each processed block. The final authentication tag is calculated from the last block. Like all counter modes, GCM works as a stream cipher, and so it is essential that a different IV is used at the start for each stream that is encrypted.

1. AES-GCM is the AES (Rijndael) block cipher in GCM block mode (integrated authenticated AEAD encryption), behaves like a stream cipher

2. Required 256-bit key and random 128-bit nonce (initial vector)

AES-256-GCM+Scrypt

During the encryption, the Scrypt KDF function is used (with some fixed parameters) to derive a secret key from the password. The randomly generated KDF salt for the key derivation is stored together with the encrypted message and will be used during the decryption. Then the input message is AES-encrypted using the secret key and the output consists of ciphertext + IV (random nonce) + authTag. The final output holds these 3 values + the KDF salt.

During the decryption, the Scrypt key derivation (with the same parameters) is used to derive the same secret key from the encryption password, together with the KDF salt (which was generated randomly during the encryption). Then the ciphertext is AES-decrypted using the secret key, the IV (nonce) and the authTag. In case of success, the result is the decrypted original plaintext. In case of error, the authentication tag will fail to authenticate the decryption process and an exception will be thrown**.**

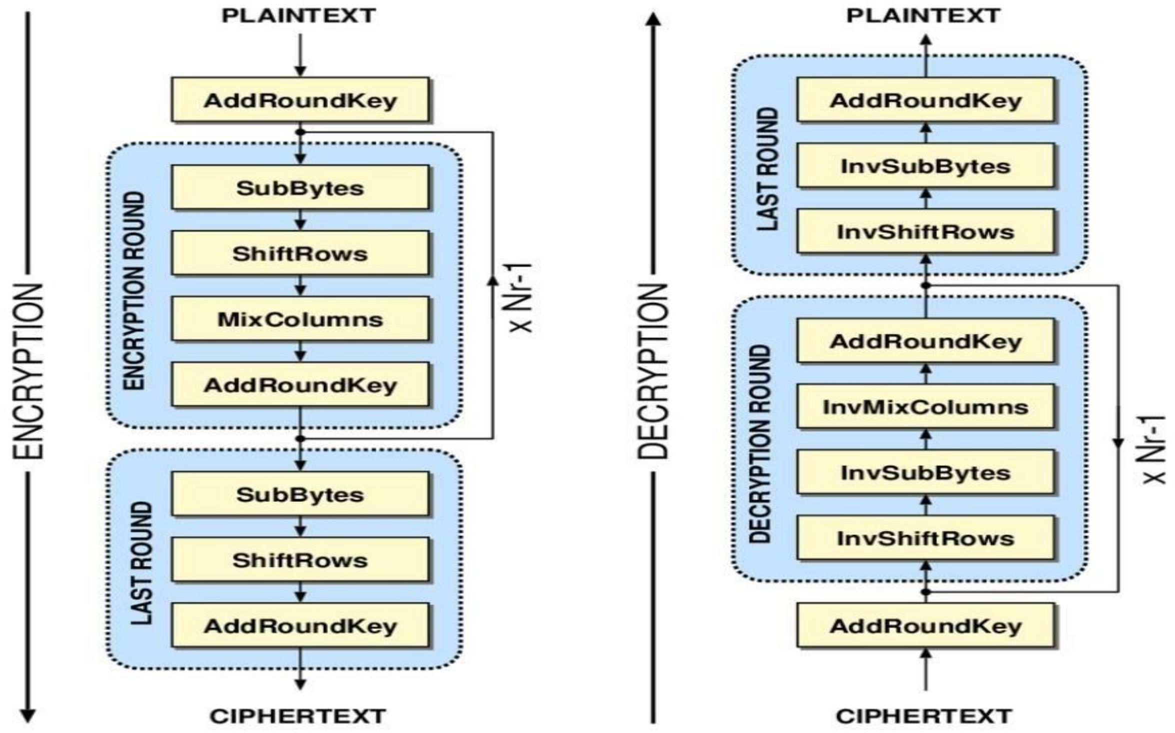


Fig 4.1.1 Working of AES-256

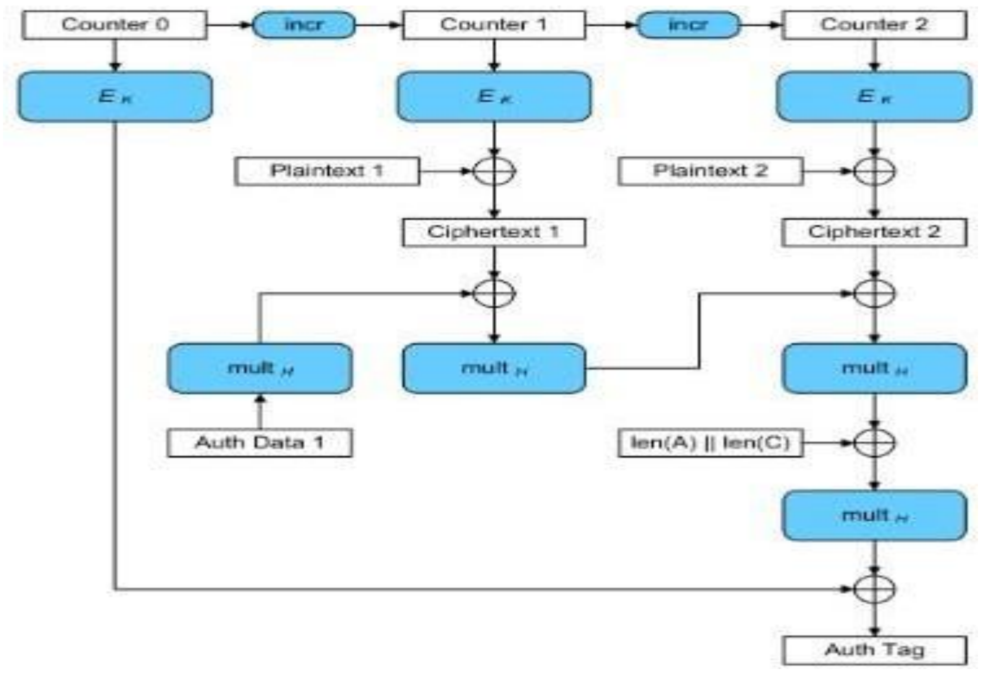


Fig 4.1.2 Working of GCM

4.2 IPFS Blockchain Integration

The IPFS Blockchain Integration Module serves as a pivotal component in our project, seamlessly linking with the Interplanetary File System (IPFS) blockchain platform. This module is dedicated to enhancing the security of our encrypted file storage system. By integrating with IPFS blockchain, it enables a decentralized and tamper-resistant environment for storing sensitive data. This integration ensures that encrypted files undergo secure uploads to the IPFS blockchain, leveraging the platform's robust distributed ledger for transparent and immutable record-keeping. The module actively participates in the generation of unique hash values associated with each uploaded file, a critical aspect in ensuring data integrity. Through this collaborative integration, our project not only embraces cutting-edge blockchain technology but also establishes a secure foundation for file storage, with a focus on transparency, integrity, and decentralization.

4.3 Hash value Classification

The Hash Value Classification Module plays a vital role in our project's security infrastructure by leveraging machine learning models. Specifically designed to dynamically analyse and categorize hash values, this module provides a proactive approach to identifying potential attacks on encrypted files. Through the utilization of advanced algorithms, it assesses patterns within hash values to distinguish between normal and potentially malicious activities. This dynamic analysis ensures real-time threat detection, enabling the system to respond swiftly to emerging security concerns. The module's integration enhances our overall security posture, adding a layer of intelligence that contributes to the robust defense against cyber threats within our encrypted file storage environment. Its adaptive nature aligns with the project's commitment to staying ahead of evolving cybersecurity challenges through innovative machine learning-driven security measures.

4.4 MetaMask Integration

The MetaMask Integration Module serves as a crucial gateway in our project, ensuring a seamless and secure user experience with Ethereum Decentralized Applications (DApps). Integrated as a browser extension, MetaMask simplifies user interactions by acting as a bridge between the user's browser and the Ethereum blockchain. It enables users to securely manage cryptocurrency transactions directly within the extension, eliminating the need for a separate Ethereum client. This integration not only enhances the overall user experience but also contributes to heightened security in cryptocurrency transactions. MetaMask streamlines the process of engaging with Ethereum DApps, providing a convenient and trustworthy means for users to interact with blockchain-based applications directly from their browsers. Its seamless integration aligns with our project's commitment to user-friendly, yet secure, interactions with decentralized technologies.

4.5 Attack Detection

Decision Tree Algorithm

The Decision Tree Algorithm is one of the popular supervised type machine learning algorithms that is used for classifications. This algorithm generates the outcome as the optimized result based upon the tree structure with the conditions or rules. The decision tree algorithm associated with three major components as Decision Nodes, Design Links, and Decision Leaves. It operates with the Splitting, pruning, and tree selection process. It supports both numerical and categorical data to construct the decision tree. Decision tree algorithms are efficient for large data set with less time complexity. This Algorithm is mostly used in customer segmentation and marketing strategy implementation in the business. A Decision Tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers to the question, and the leaves represent the actual output or class label. They are used in non-linear decision making with a simple linear decision surface. The Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can solve regression and classification problems. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred

from prior data(training data). In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the

branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure(ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node

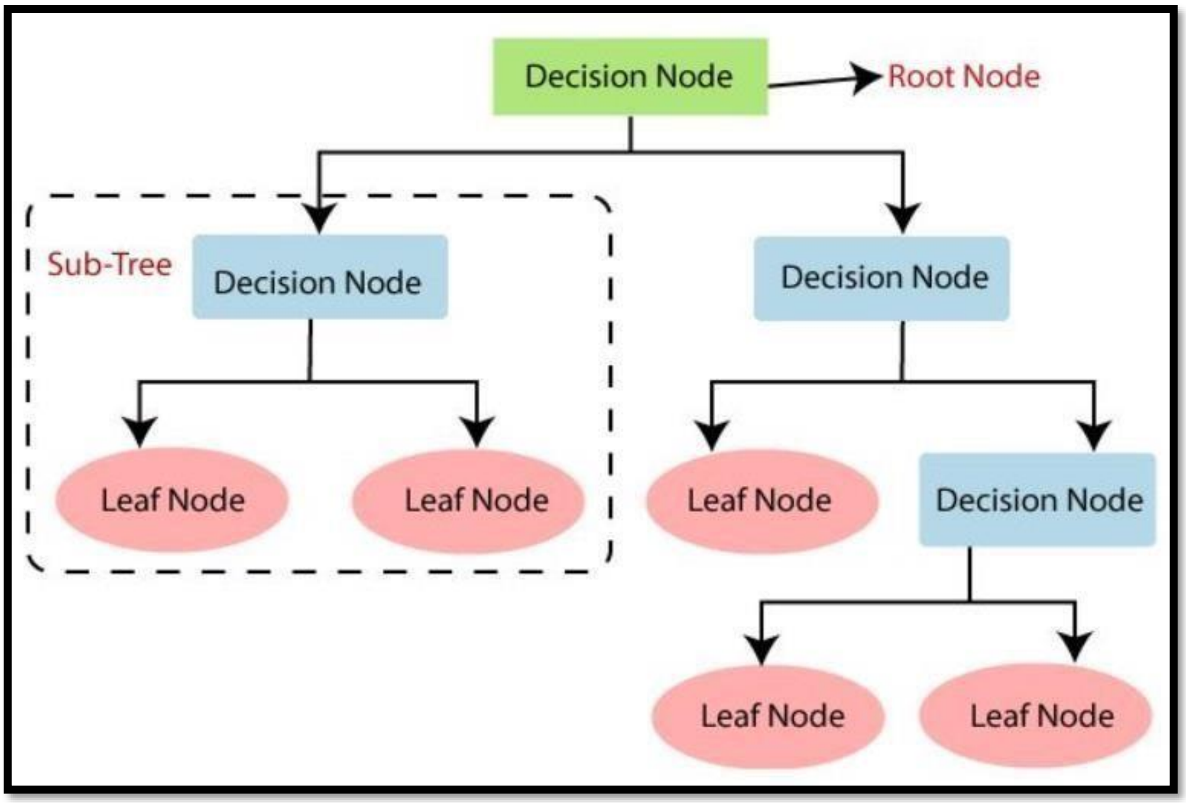


Fig 4.5.1 Working of Decision Tree Algorithm

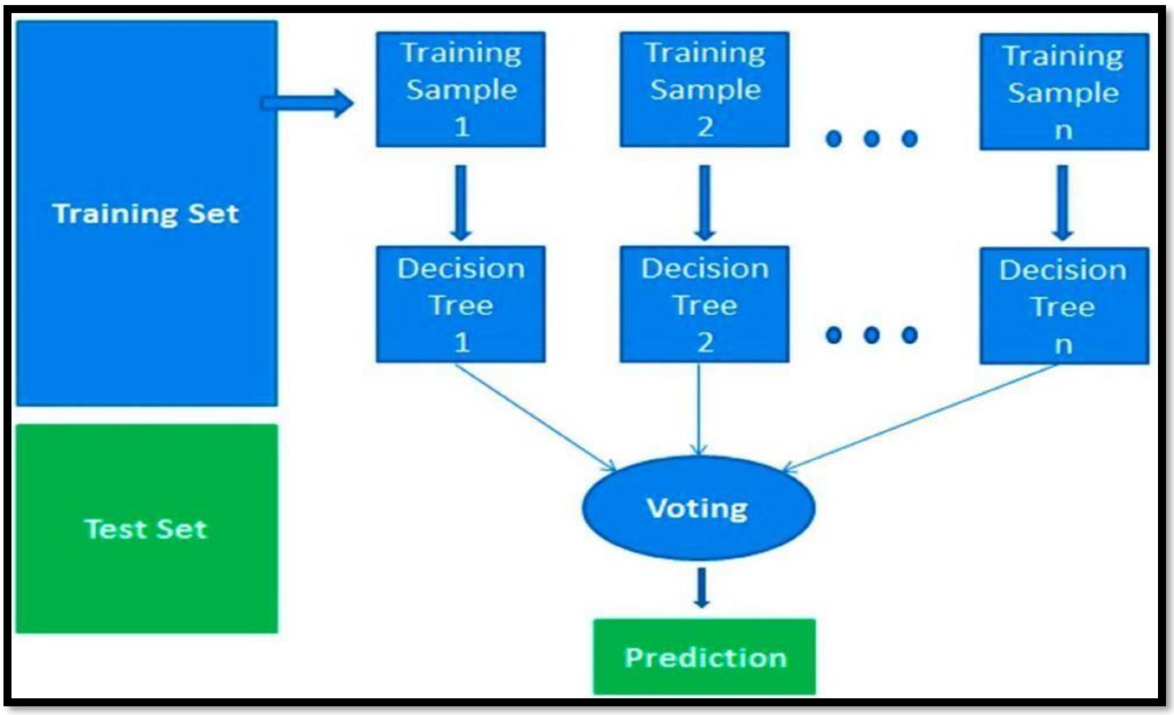


Fig 4.5.2 Workflow of Decision Tree Algorithm

5.IMPLEMENTATION

5.1 Technologies Used

Blockchain Integration

Blockchain technology, as a distributed ledger, fortifies security through it inherent characteristics of transparency and immutability. Within the project, blockchain plays a pivotal role in safeguarding the integrity of encrypted files. By leveraging cryptographic hashing and consensus algorithms, each transaction, including file uploads and modifications, is securely recorded across a decentralized network of nodes. This ensures that once data is added to the blockchain, it becomes tamper-resistant and virtually impossible to alter retroactively. Consequently, the decentralized framework established by blockchain not only enhances data security but also fosters trust by providing a transparent and unalterable record of all activities related to encrypted file storage. The decentralized and distributed nature of blockchain contributes to a resilient and trustworthy foundation for secure data storage in the cloud environment.

5.2 Sample Code

5.2.1 Code for Cryptography

Code for installing libraries

```
pip install cryptography
pip install pycryptodome
pip install scrypt
```

Code for importing libraries

```
from cryptography.fernet import Fernet
from Crypto.Cipher  import AES
import scrypt, os, binascii
import numpy as np
```

Code for connecting to google drive

```
from google.colab import drive drive.mount('/content/drive')
```

Code for loading the file from google drive

```
 File=np.loadtxt('/content/drive/MyDrive/Project/AESfile',delimiter=',',skiprows=0,dtype=str)
```

Code for creating files and writing content

```
 with open('password', 'wb') as pw_file:
   pw_file.write(Key)
with open('msg', 'wb') as msg_file:
        msg_file.write(File)
```

Code for reading files and storing the data

```
with open('password', 'rb') as pw_file:
   passwordFileContent = pw_file.read()
with open('msg', 'rb') as msg_file:
    msgFileContent = msg_file.read()
```

Code for encryption

```
def encrypt_AES_GCM(msg, password):
    kdfSalt = os.urandom(16)
    secretKey = scrypt.hash(password, kdfSalt, N=16384, r=8, p=1, buflen=32)
    aesCipher = AES.new(secretKey, AES.MODE_GCM)
    ciphertext, authTag = aesCipher.encrypt_and_digest(msg)
    return (kdfSalt, ciphertext, aesCipher.nonce, authTag)
```

Code for decryption

```
def decrypt_AES_GCM(encryptedMsg, password):
    (kdfSalt, ciphertext, nonce, authTag) = encryptedMsg
    secretKey = scrypt.hash(password, kdfSalt, N=16384, r=8, p=1, buflen=32)
    aesCipher = AES.new(secretKey, AES.MODE_GCM, nonce)
    plaintext = aesCipher.decrypt_and_verify(ciphertext, authTag)
    return plaintext
```

Code for assigning contents of the file and key

```
msg = msgFileContent
password = passwordFileContent
```

Code for encryption of file

```
encryptedMsg = encrypt_AES_GCM(msg, password)
print("encryptedMsg=", binascii.hexlify(encryptedMsg[1]))
```

Code for overwriting encrypted file

```
with open('/content/drive/MyDrive/Project/AES file','wb') as writefile:
    writefile.write(binascii.hexlify(encryptedMsg[1]))
```

Code for decryption of file

```
decryptedMsg = decrypt_AES_GCM(encryptedMsg, password)
print("decryptedMsg", decryptedMsg)
```

Code for overwriting decrypted file

```
with open('/content/drive/MyDrive/Project/AES file','wb') as writefile:
    writefile.write(decryptedMsg)
```

5.2.2 Code for Attack Detection Mechanism

Code for importing libraries

```
import os

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier
```

Code for connecting to google drive

```
from google.colab import drive drive.mount('/content/drive')
```

Code for loading the file from google drive

```
df=pd.read_csv('/content/drive/MyDrive/Project/Share ADM-file.csv')
```

Code for dividing the dataset for training and testing

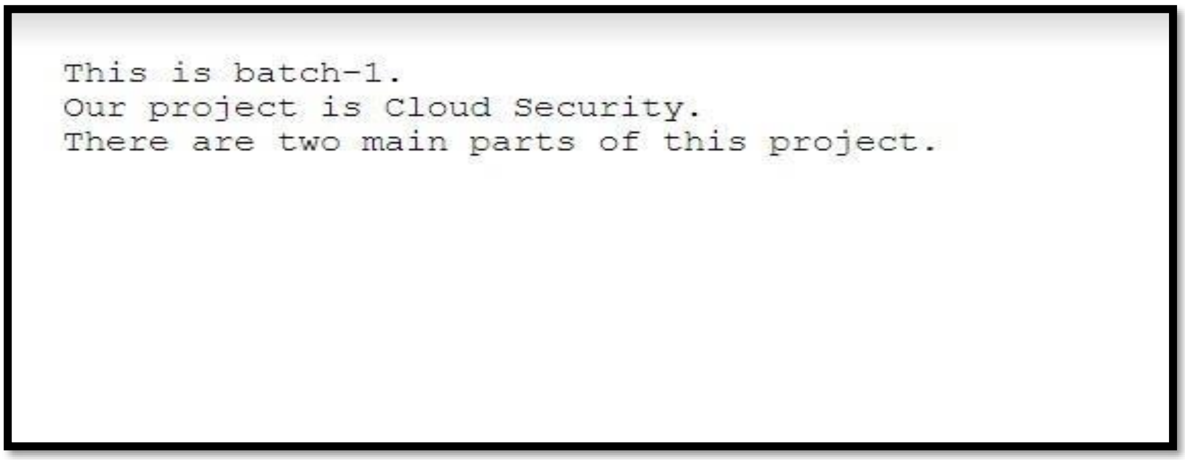
```
temp=df.drop('type',axis=1)

x=temp.drop('hash',axis=1)

y=df['type']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
```

Code for creating, training and testing the model

```
dtc=DecisionTreeClassifier()

dtc.fit(x_train,y_train)

y_pred=dtc.predict(x_test)
```

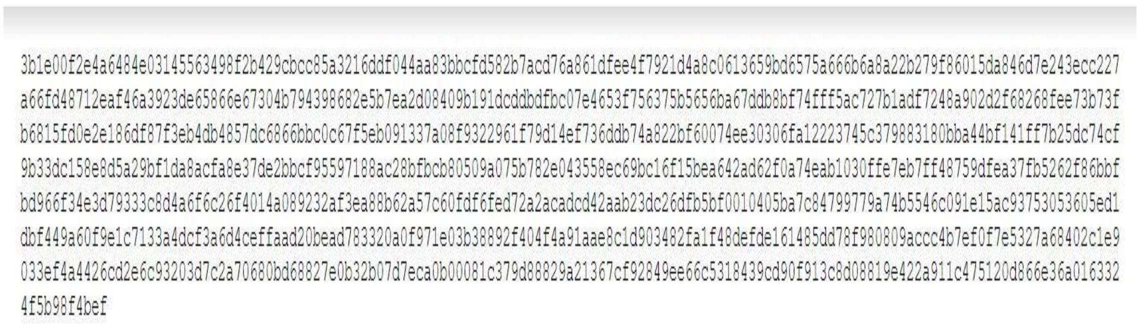
5.3 Results of the Project

Contents of original file



```
This is batch-1.
Our project is Cloud Security.
There are two main parts of this project.
```

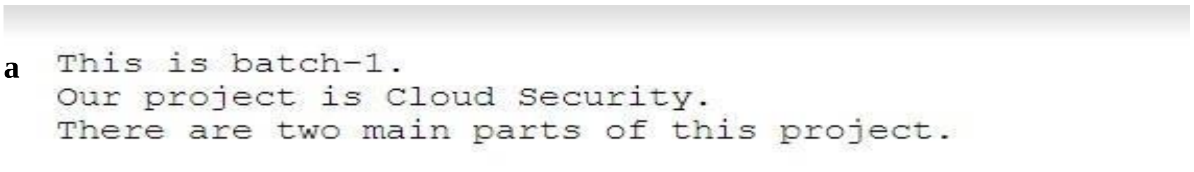
Contents of encrypted file



```
3b1e00f2e4a6484e03145563498f2b429cbcc85a3216ddf044aa83bbcfd582b7acd76a861dfee4f7921d4a8c0613659bd6575a666b6a8a22b279f86015da846d7e243ecc227
a66fd48712eaf46a3923de65866e67304b794398682e5b7ea2d08409b191dcddbdfbc07e4653f756375b5656ba67ddb8bf74fff5ac727b1adf7248a902d2f68268fee73b73f
b6815fd0e2e186df87f3eb4db4857dc6866bbc0c67f5eb091337a08f9322961f79d14ef736ddb74a822bf60074ee30306fa12223745c379883180bba44bf141ff7b25dc74cf
9b33dc158e8d5a29bf1da8acfa8e37de2bbcf95597188ac28bfbcb80509a075b782e043558ec69bc16f15bea642ad62f0a74eab1030ffe7eb7ff48759dfea37fb5262f86bbf
bd966f34e3d79333c8d4a6f6c26f4014a089232af3ea88b62a57c60fdf6fed72a2acadcd42aab23dc26dfb5bf0010405ba7c84799779a74b5546c091e15ac93753053605ed1
dbf449a60f9e1c7133a4dcf3a6d4ceffaad20bead783320a0f971e03b38892f404f4a91aae8c1d903482fa1f48defde161485dd78f980809accc4b7ef0f7e5327a68402c1e9
033ef4a4426cd2e6c93203d7c2a70680bd68827e0b32b07d7eca0b00081c379d88829a21367cf92849ee66c5318439cd90f913c8d08819e422a911c475120d866e36a016332
4f5b98f4bef
```

Contents of decrypted file

a



```
This is batch-1.
Our project is Cloud Security.
There are two main parts of this project.
```

Attack Detection

Testing results

dtc.score(x_test,y_test)

0.9955476402493322

{Model accuracy = 99.55% }

6.TESTING

Software Testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free. It involves the execution of a software component or system component to evaluate one or more properties of interest. It is required for evaluating the system. This phase is the critical phase of software quality assurance and presents the ultimate view of coding.

6.1 Test Codes

Test for encryption of file

```
def encrypt_AES_GCM(msg, password):
    kdfSalt = os.urandom(16)
    secretKey = scrypt.hash(password, kdfSalt, N=16384, r=8, p=1, buflen=32)
    aesCipher = AES.new(secretKey, AES.MODE_GCM)
    ciphertext, authTag = aesCipher.encrypt_and_digest(msg)
    return (kdfSalt, ciphertext, aesCipher.nonce, authTag)


encryptedMsg = encrypt_AES_GCM(msg, password)
print("encryptedMsg=", binascii.hexlify(encryptedMsg[1]))


with open('/content/drive/MyDrive/Project/AES file','wb') as writefile:
    writefile.write(decryptedMsg)
```

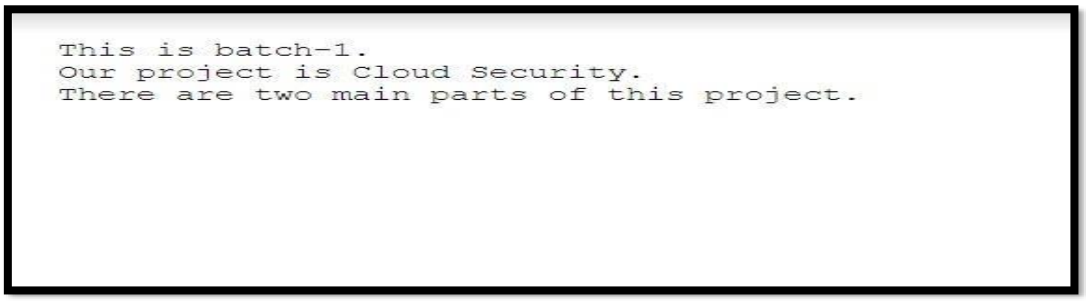
Test for decryption of file

```
def decrypt_AES_GCM(encryptedMsg, password):
    (kdfSalt, ciphertext, nonce, authTag) = encryptedMsg
    secretKey = scrypt.hash(password, kdfSalt, N=16384, r=8, p=1, buflen=32)
    aesCipher = AES.new(secretKey, AES.MODE_GCM, nonce)
    plaintext = aesCipher.decrypt_and_verify(ciphertext, authTag)
    return plaintext
```

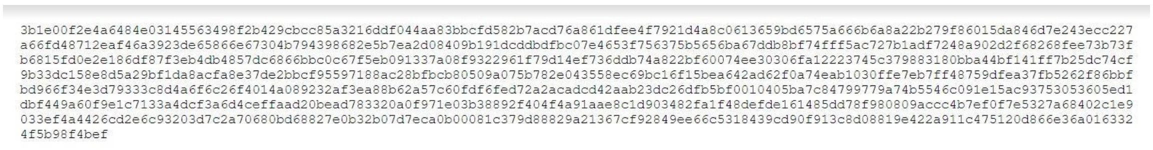
```
decryptedMsg = decrypt_AES_GCM(encryptedMsg, password)
print("decryptedMsg", decryptedMsg)


with open('/content/drive/MyDrive/Project/AES file','wb') as writefile:
    writefile.write(decryptedMsg)
```

6.2 Test Case

Test case for encryption of file

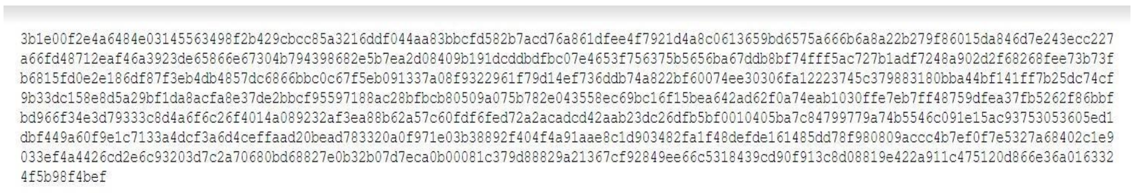


```
This is batch-1.
Our project is Cloud Security.
There are two main parts of this project.
```

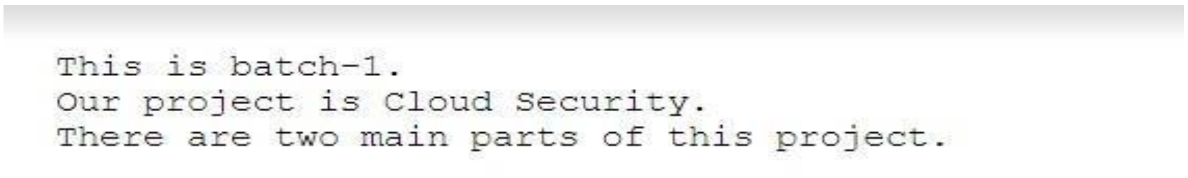


```
3b1e00f2e4a6484e03145563498f2b429cbcc85a3216ddf044aa83bbcfd582b7acd76a861dfee4f7921d4a8c0613659bd6575a666b6a8a22b279f86015da846d7e243ecc227
a66fd48712eaf46a3923de65866e67304b794398682e5b7ea2d08409b191dcddbdfbc07e4653f756375b5656ba67ddb8bf74fff5ac727b1adf7248a902d2f68268fee73b73f
b6815fd0e2e186df87f3eb4db4857dc6866bbc0c67f5eb091337a08f9322961f79d14ef736ddb74a822bf60074ee30306fa12223745c379883180bba44bf141ff7b25dc74cf
9b33dc158e8d5a29bf1da8acfa8e37de2bbcf95597188ac28bfbcb80509a075b782e043558ec69bc16f15bea642ad62f0a74eab1030ffe7eb7ff48759dfea37fb5262f86bbf
bd966f34e3d79333c8d4a6f6c26f4014a089232af3ea88b62a57c60fdf6fed72a2acadcd42aab23dc26dfb5bf0010405ba7c84799779a74b5546c091e15ac93753053605ed1
dbf449a60f9e1c7133a4dcf3a6d4ceffaad20bead783320a0f971e03b38892f404f4a91aae8c1d903482fa1f48defde161485dd78f980809accc4b7ef0f7e5327a68402c1e9
033ef4a4426cd2e6c93203d7c2a70680bd68827e0b32b07d7eca0b00081c379d88829a21367cf92849ee66c5318439cd90f913c8d08819e422a911c475120d866e36a016332
4f5b98f4bef
```

Test case for decryption of file

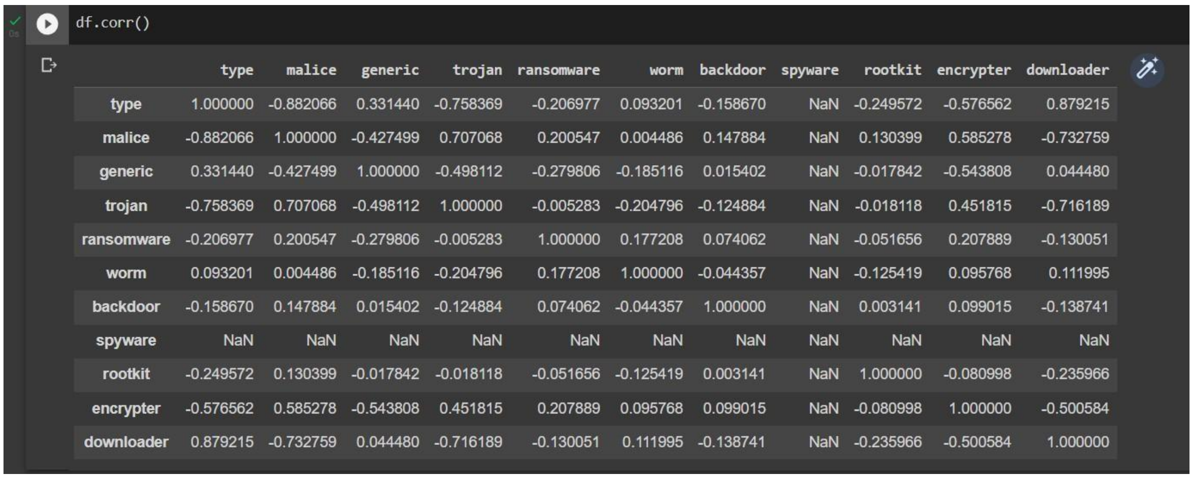


```
3b1e00f2e4a6484e03145563498f2b429cbcc85a3216ddf044aa83bbcfd582b7acd76a861dfee4f7921d4a8c0613659bd6575a666b6a8a22b279f86015da846d7e243ecc227
a66fd48712eaf46a3923de65866e67304b794398682e5b7ea2d08409b191dcddbdfbc07e4653f756375b5656ba67ddb8bf74fff5ac727b1adf7248a902d2f68268fee73b73f
b6815fd0e2e186df87f3eb4db4857dc6866bbc0c67f5eb091337a08f9322961f79d14ef736ddb74a822bf60074ee30306fa12223745c379883180bba44bf141ff7b25dc74cf
9b33dc158e8d5a29bf1da8acfa8e37de2bbcf95597188ac28bfbcb80509a075b782e043558ec69bc16f15bea642ad62f0a74eab1030ffe7eb7ff48759dfea37fb5262f86bbf
bd966f34e3d79333c8d4a6f6c26f4014a089232af3ea88b62a57c60fdf6fed72a2acadcd42aab23dc26dfb5bf0010405ba7c84799779a74b5546c091e15ac93753053605ed1
dbf449a60f9e1c7133a4dcf3a6d4ceffaad20bead783320a0f971e03b38892f404f4a91aae8c1d903482fa1f48defde161485dd78f980809accc4b7ef0f7e5327a68402c1e9
033ef4a4426cd2e6c93203d7c2a70680bd68827e0b32b07d7eca0b00081c379d88829a21367cf92849ee66c5318439cd90f913c8d08819e422a911c475120d866e36a016332
4f5b98f4bef
```



```
This is batch-1.
Our project is Cloud Security.
There are two main parts of this project.
```

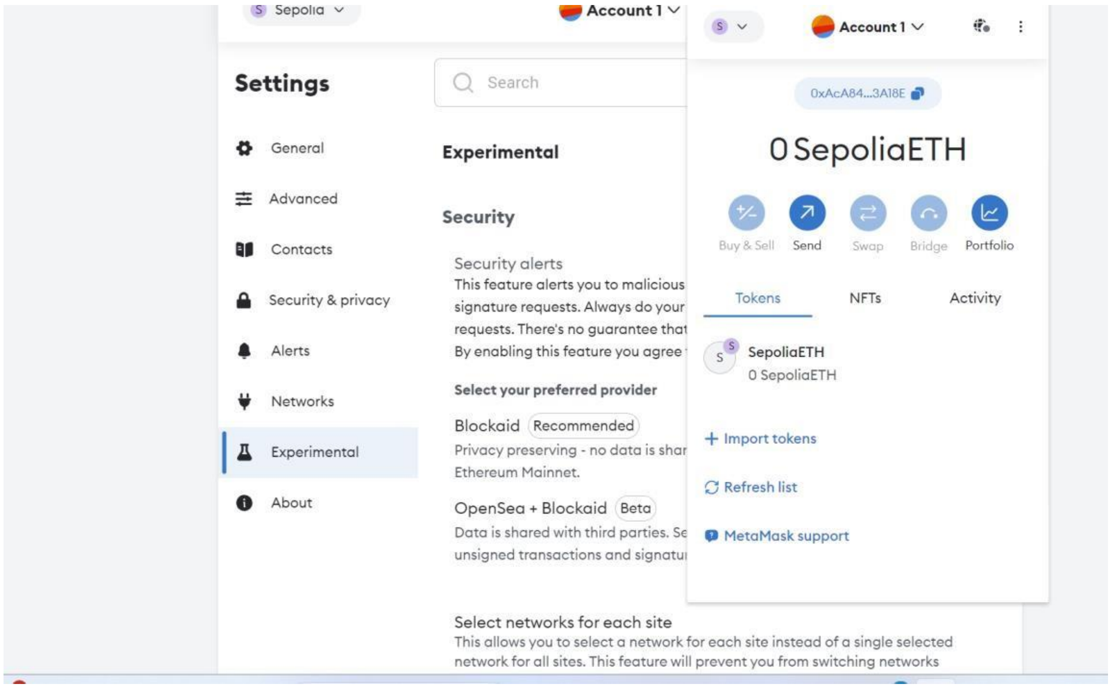
Test case for attack detection mechanism



```
df.corr()
```

	type	malice	generic	trojan	ransomware	worm	backdoor	spyware	rootkit	encrypter	downloader
type	1.000000	-0.882066	0.331440	-0.758369	-0.206977	0.093201	-0.158670	NaN	-0.249572	-0.576562	0.879215
malice	-0.882066	1.000000	-0.427499	0.707068	0.200547	0.004486	0.147884	NaN	0.130399	0.585278	-0.732759
generic	0.331440	-0.427499	1.000000	-0.498112	-0.279806	-0.185116	0.015402	NaN	-0.017842	-0.543808	0.044480
trojan	-0.758369	0.707068	-0.498112	1.000000	-0.005283	-0.204796	-0.124884	NaN	-0.018118	0.451815	-0.716189
ransomware	-0.206977	0.200547	-0.279806	-0.005283	1.000000	0.177208	0.074062	NaN	-0.051656	0.207889	-0.130051
worm	0.093201	0.004486	-0.185116	-0.204796	0.177208	1.000000	-0.044357	NaN	-0.125419	0.095768	0.111995
backdoor	-0.158670	0.147884	0.015402	-0.124884	0.074062	-0.044357	1.000000	NaN	0.003141	0.099015	-0.138741
spyware	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
rootkit	-0.249572	0.130399	-0.017842	-0.018118	-0.051656	-0.125419	0.003141	NaN	1.000000	-0.080998	-0.235966
encrypter	-0.576562	0.585278	-0.543808	0.451815	0.207889	0.095768	0.099015	NaN	-0.080998	1.000000	-0.500584
downloader	0.879215	-0.732759	0.044480	-0.716189	-0.130051	0.111995	-0.138741	NaN	-0.235966	-0.500584	1.000000

6.3 Screenshots of Blockchain Implementation



Sepolia | Account 1 | Account 1

Settings | Search

General | Advanced | Contacts | Security & privacy | Alerts | Networks | Experimental | About

Experimental

Security

Security alerts
This feature alerts you to malicious
signature requests. Always do your
requests. There's no guarantee that
By enabling this feature you agree

Select your preferred provider

Blockaid Recommended
Privacy preserving - no data is shar
Ethereum Mainnet.

OpenSea + Blockaid Beta
Data is shared with third parties. Se
unsigned transactions and signatu

Select networks for each site
This allows you to select a network for each site instead of a single selected
network for all sites. This feature will prevent you from switching networks

0xAcA84...3A18E

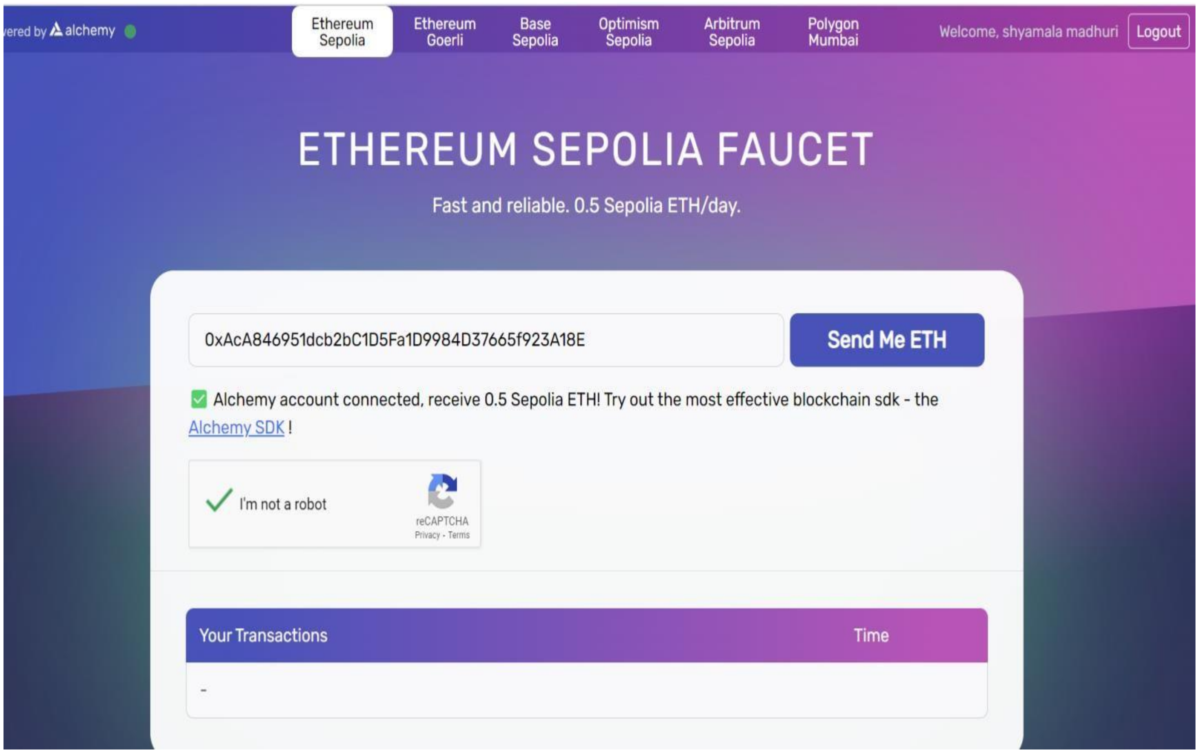
0 SepoliaETH

Buy & Sell | Send | Swap | Bridge | Portfolio

Tokens | NFTs | Activity

SepoliaETH
0 SepoliaETH

+ Import tokens
Refresh list
MetaMask support



ered by alchemy | Ethereum Sepolia | Ethereum Goerli | Base Sepolia | Optimism Sepolia | Arbitrum Sepolia | Polygon Mumbai | Welcome, shyamala madhuri | Logout

ETHEREUM SEPOLIA FAUCET

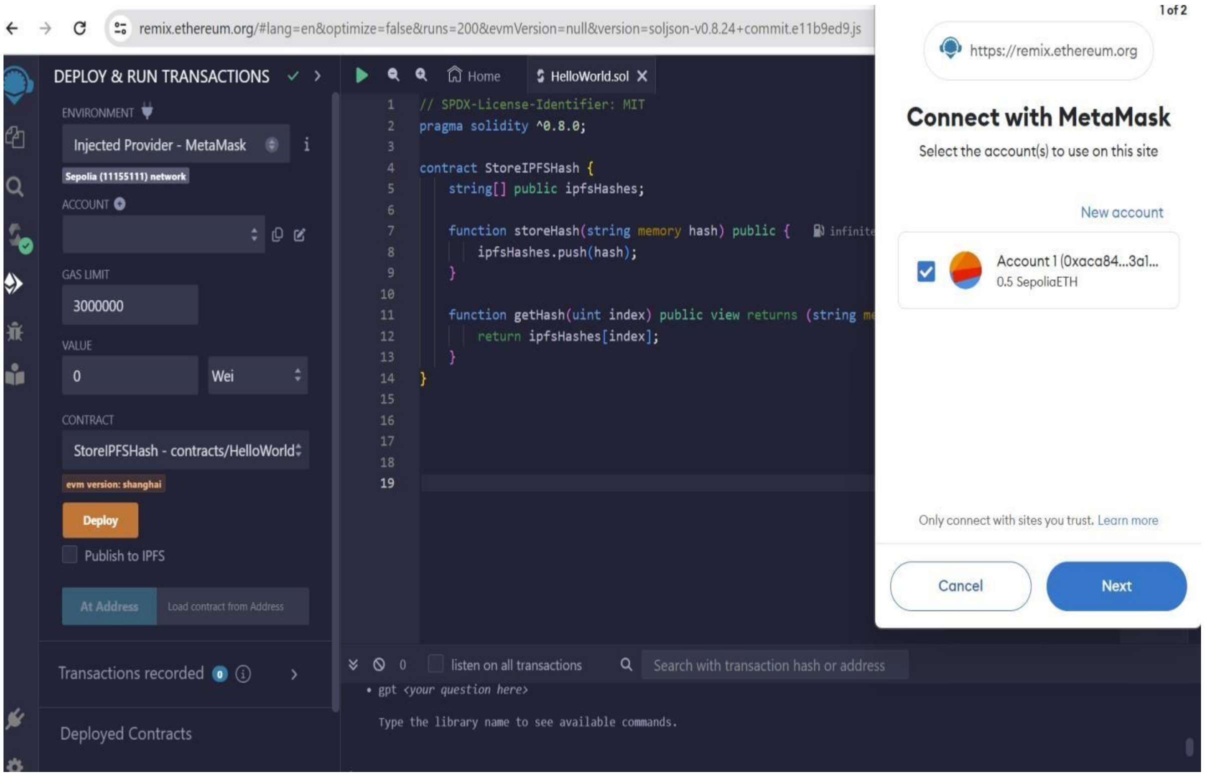
Fast and reliable. 0.5 Sepolia ETH/day.

0xAcA846951dcb2bC1D5Fa1D9984D37665f923A18E | Send Me ETH

✓ Alchemy account connected, receive 0.5 Sepolia ETH! Try out the most effective blockchain sdk - the Alchemy SDK !

✓ I'm not a robot | reCAPTCHA Privacy - Terms

Your Transactions	Time
-	


remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.24+commit.e11b9ed9.js

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - MetaMask

Sepolia (11155111) network

ACCOUNT

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: StoreIPFSHash - contracts/HelloWorld.sol

evm version: shanghai

Deploy

Publish to IPFS

At Address — Load contract from Address

Transactions recorded 0

Deployed Contracts

Home | HelloWorld.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract StoreIPFSHash {
    string[] public ipfsHashes;

    function storeHash(string memory hash) public {    infinite
        ipfsHashes.push(hash);
    }

    function getHash(uint index) public view returns (string me
        return ipfsHashes[index];
    }
}
```

0 listen on all transactions — Search with transaction hash or address

```
• gpt <your question here>

Type the library name to see available commands.
```

1 of 2

https://remix.ethereum.org

Connect with MetaMask

Select the account(s) to use on this site

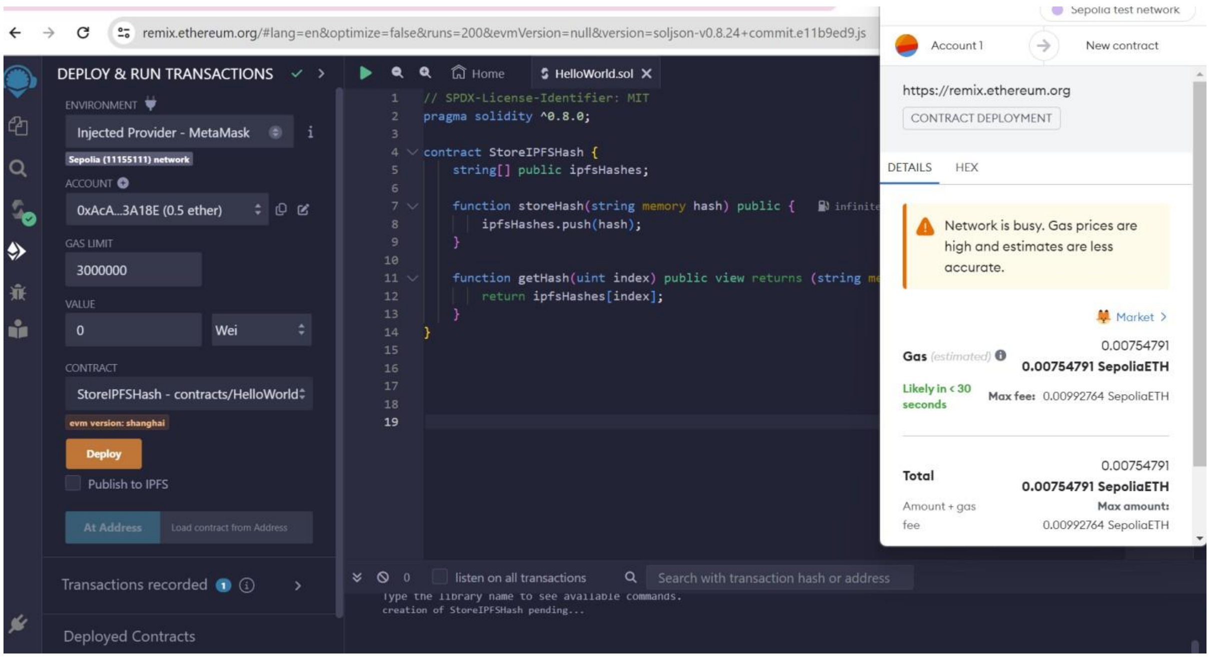
New account

☑ Account 1 (0xaca84...3a1...
0.5 SepoliaETH

Only connect with sites you trust. Learn more

Cancel | Next

After deploy to copy data to Blockchain need gas, we need to buy gas currency with sepolia ETH


remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.24+commit.e11b9ed9.js

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - MetaMask

Sepolia (11155111) network

ACCOUNT: 0xAcA...3A18E (0.5 ether)

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: StoreIPFSHash - contracts/HelloWorld.sol

evm version: shanghai

Deploy

Publish to IPFS

At Address — Load contract from Address

Transactions recorded 1

Deployed Contracts

Home | HelloWorld.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract StoreIPFSHash {
    string[] public ipfsHashes;

    function storeHash(string memory hash) public {    infinite
        ipfsHashes.push(hash);
    }

    function getHash(uint index) public view returns (string me
        return ipfsHashes[index];
    }
}
```

0 listen on all transactions — Search with transaction hash or address

```
Type the library name to see available commands.
creation of StoreIPFSHash pending...
```

Sepolia test network

Account 1 → New contract

https://remix.ethereum.org

CONTRACT DEPLOYMENT

DETAILS | HEX

Network is busy. Gas prices are high and estimates are less accurate.

Market >

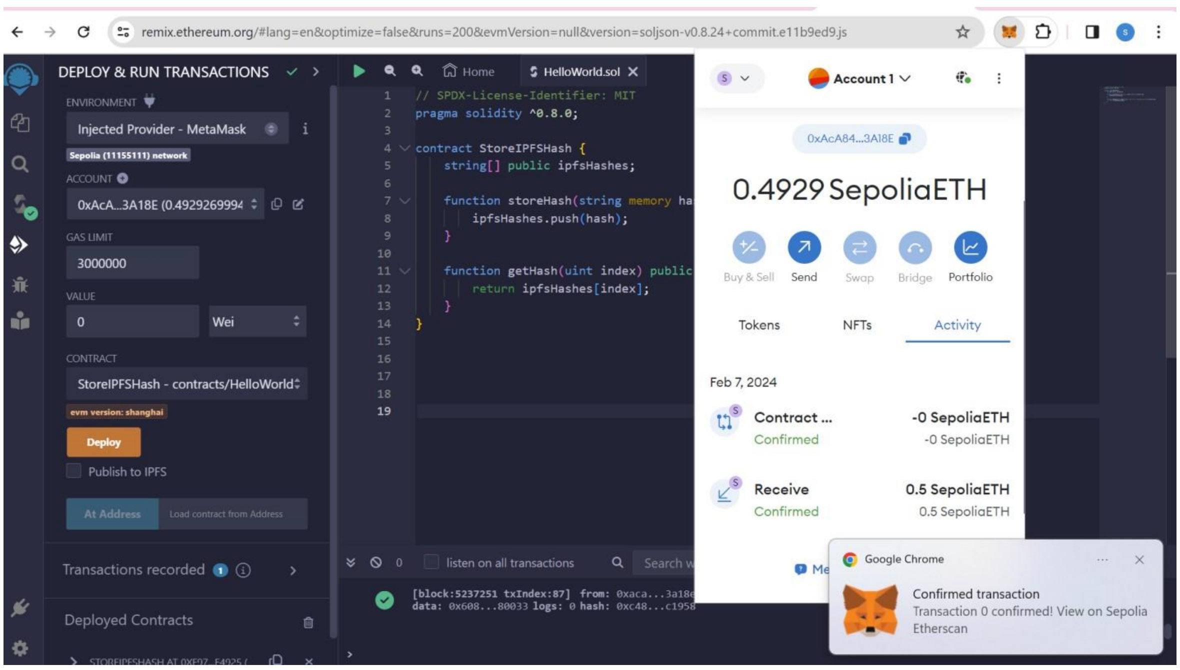
Gas (estimated) 0.00754791
0.00754791 SepoliaETH

Likely in < 30 seconds — Max fee: 0.00992764 SepoliaETH

Total 0.00754791
0.00754791 SepoliaETH

Amount + gas fee — Max amount: 0.00992764 SepoliaETH

Block Created:



remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.24+commit.e11b9ed9.js

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - MetaMask

Sepolia (11155111) network

ACCOUNT: 0xAcA...3A18E (0.4929269994

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: StoreIPFSHash - contracts/HelloWorld

evm version: shanghai

Deploy

Publish to IPFS

At Address | Load contract from Address

Transactions recorded 1

Deployed Contracts

STOREIPFSHASH AT 0XF97...F4925 (

HelloWorld.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract StoreIPFSHash {
    string[] public ipfsHashes;

    function storeHash(string memory ha
        ipfsHashes.push(hash);
    }

    function getHash(uint index) public
        return ipfsHashes[index];
    }
}
```

listen on all transactions

```
[block:5237251 txIndex:87] from: 0xaca...3a18e
data: 0x608...80033 logs: 0 hash: 0xc48...c1958
```

Account 1

0xAcA84...3A18E

0.4929 SepoliaETH

Feb 7, 2024

Contract ... Confirmed	-0 SepoliaETH -0 SepoliaETH
Receive Confirmed	0.5 SepoliaETH 0.5 SepoliaETH

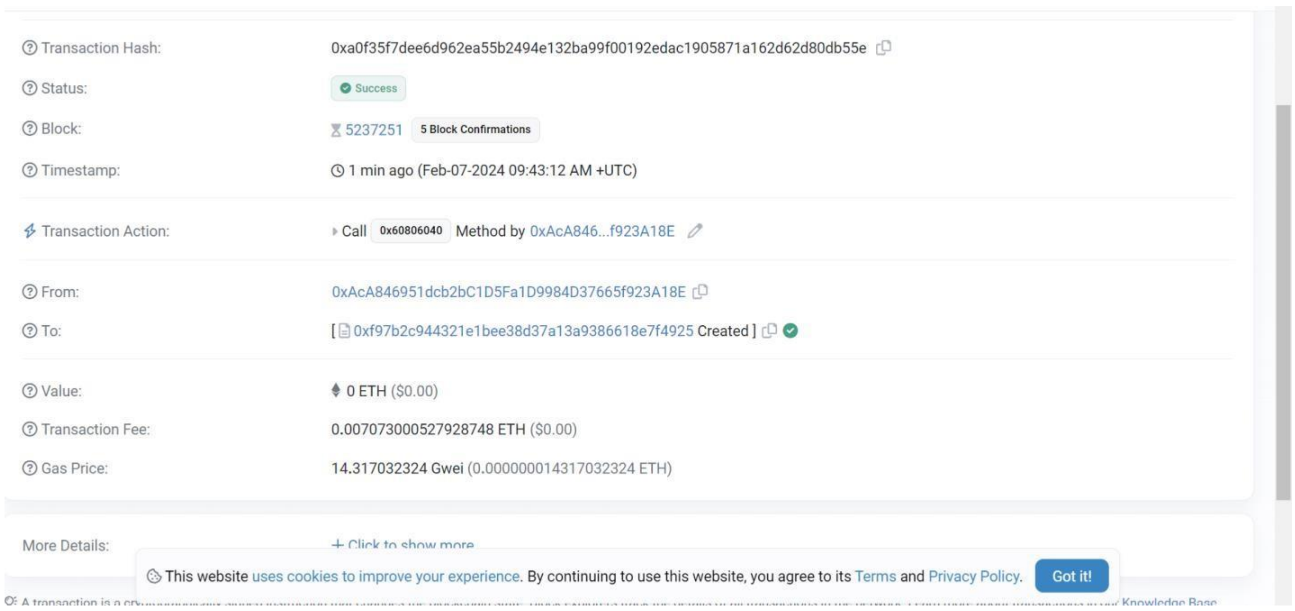
Google Chrome

Confirmed transaction

Transaction 0 confirmed! View on Sepolia Etherscan

View All transaction details:

1. Gas price
2. Acknowledgement



Transaction Hash: 0xa0f35f7dee6d962ea55b2494e132ba99f00192edac1905871a162d62d80db55e

Status: Success

Block: 5237251 5 Block Confirmations

Timestamp: 1 min ago (Feb-07-2024 09:43:12 AM +UTC)

Transaction Action: Call 0x60806040 Method by 0xAcA846...f923A18E

From: 0xAcA846951dcb2bC1D5Fa1D9984D37665f923A18E

To: [0xf97b2c944321e1bee38d37a13a9386618e7f4925 Created]

Value: 0 ETH ($0.00)

Transaction Fee: 0.007073000527928748 ETH ($0.00)

Gas Price: 14.317032324 Gwei (0.000000014317032324 ETH)

More Details: + Click to show more

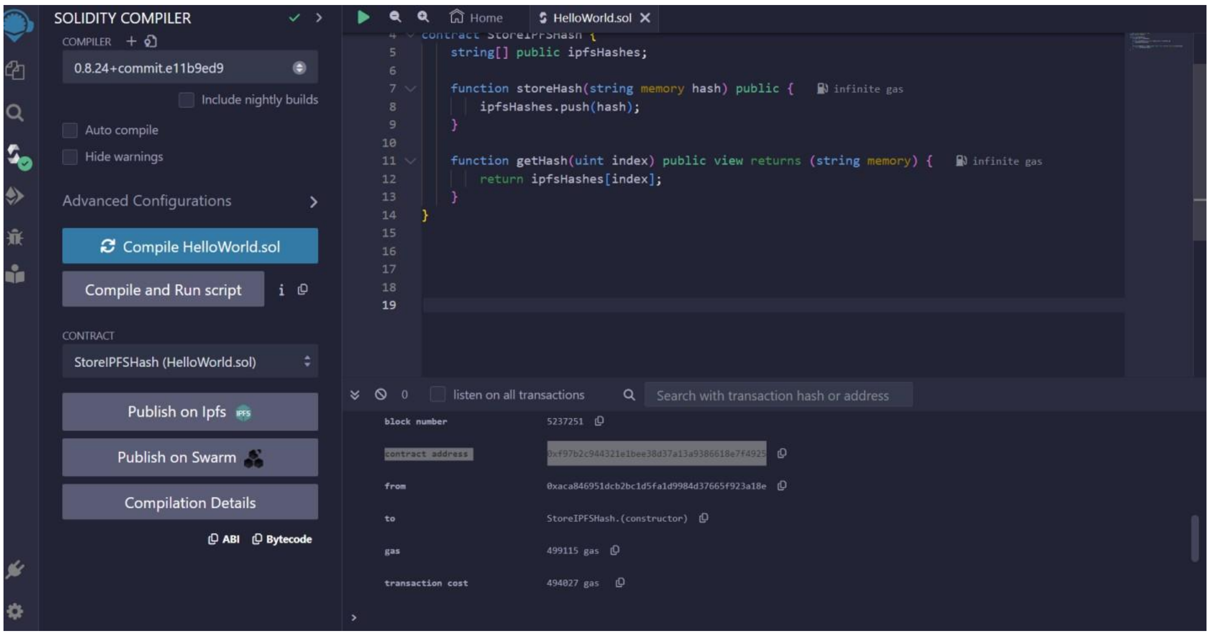
This website uses cookies to improve your experience. By continuing to use this website, you agree to its Terms and Privacy Policy. Got it!

A transaction is a cr [illegible] Knowledge Base

```
contract
address   0xf97b2c944321e1bee38d37a13a9386618e7f4925
```

Solidity Compiler: ABI (application Binary Interface)

Abi address

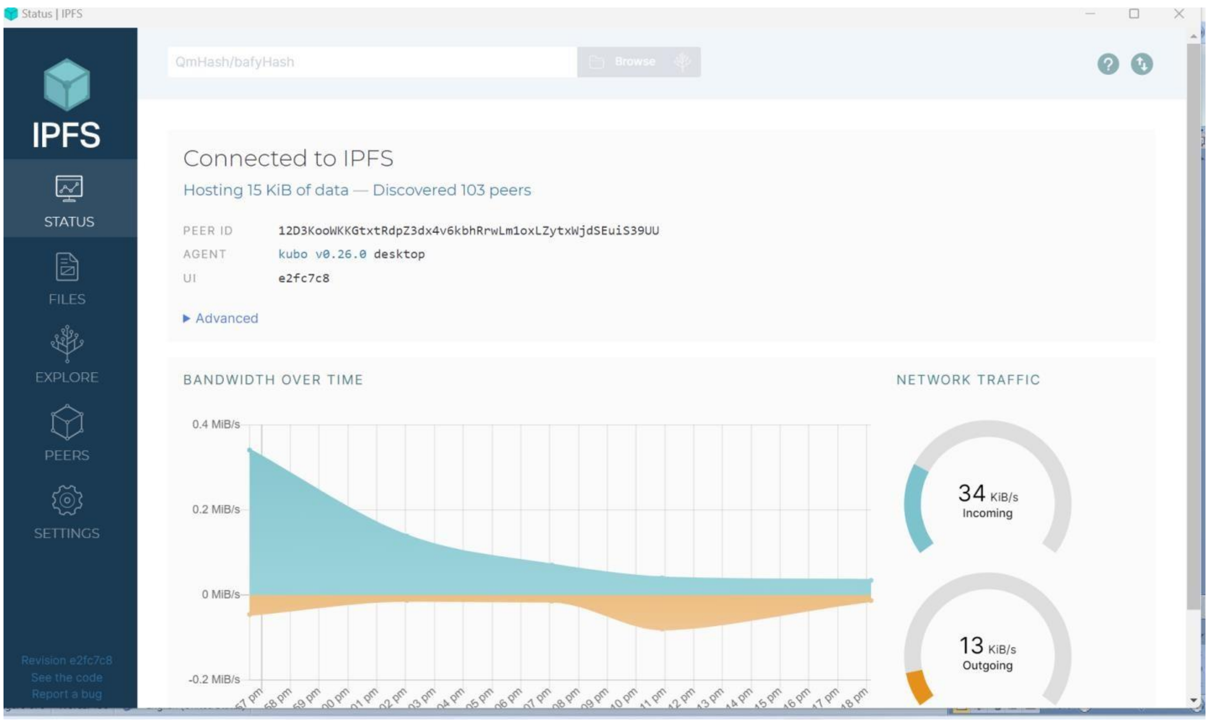


```
contract StoreIPFSHash {
    string[] public ipfsHashes;

    function storeHash(string memory hash) public {    infinite gas
        ipfsHashes.push(hash);
    }

    function getHash(uint index) public view returns (string memory) {    infinite gas
        return ipfsHashes[index];
    }
}
```

```
block number        5237251
contract address    0xf97b2c944321e1bee38d37a13a9386618e7f4925
from                0xaca846951dcb2bc1d5fa1d9984d37665f923a18e
to                  StoreIPFSHash.(constructor)
gas                 499115 gas
transaction cost    494027 gas
```

```
[
{
        "inputs": [
                {
                        "internalType": "string",
                        "name": "hash",
                        "type": "string"
                }

        ],
        "name": "storeHash",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
},
{
        "inputs": [
                {
                        "internalType": "uint256",
                        "name": "index",
                        "type": "uint256"
                }
        ],
        "name": "getHash",
        "outputs": [
                {
                        "internalType": "string",
                        "name": "",
                        "type": "string"
                    }
                ],
                "stateMutability": "view",
                "type": "function"
        },
        {
                "inputs": [
                    {
                        "internalType": "uint256",
                        "name": "",
                        "type": "uint256"
                    }
                ],
                "name": "ipfsHashes",
                "outputs": [
                    {
                        "internalType": "string",
                        "name": "",
                        "type": "string"
                    }
                ],
                "stateMutability": "view",
                "type": "function"
        }
]
```

IPFS ENVIRONEMENT



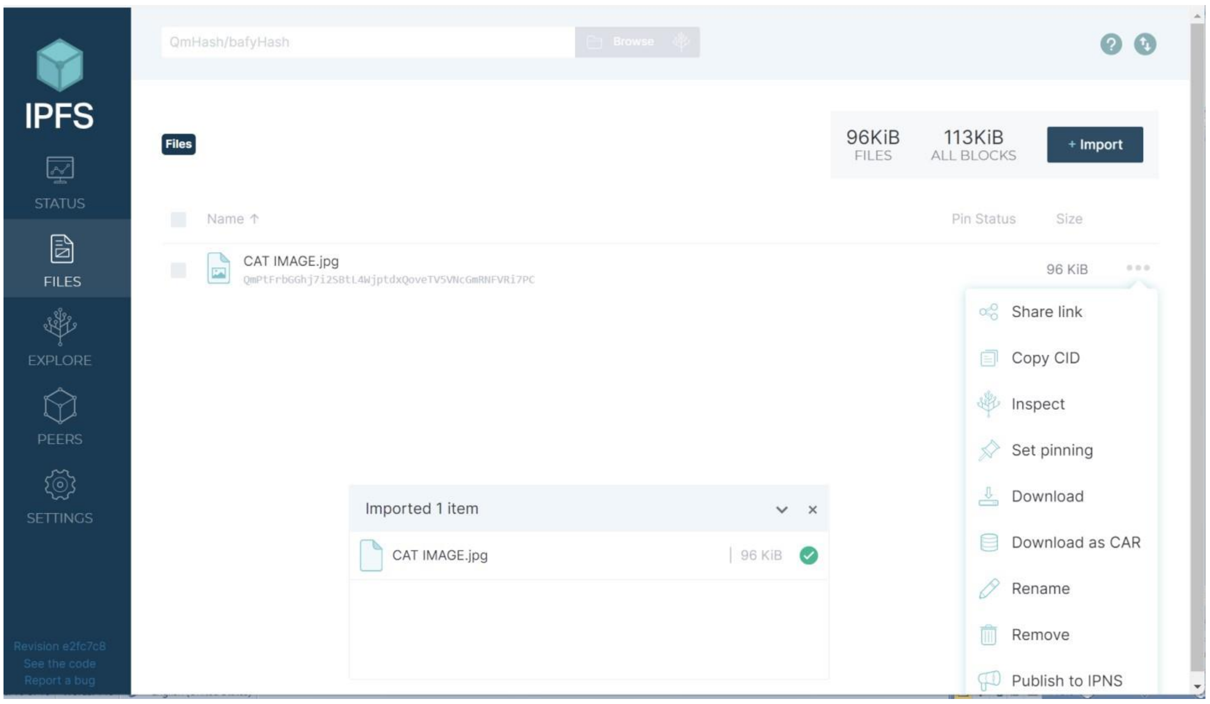
Connected to IPFS
Hosting 15 KiB of data — Discovered 103 peers

```
PEER ID   12D3KooWKKGtxtRdpZ3dx4v6kbhRrwLm1oxLZytxWjdSEuiS39UU
AGENT     kubo v0.26.0 desktop
UI        e2fc7c8
```

▶ Advanced

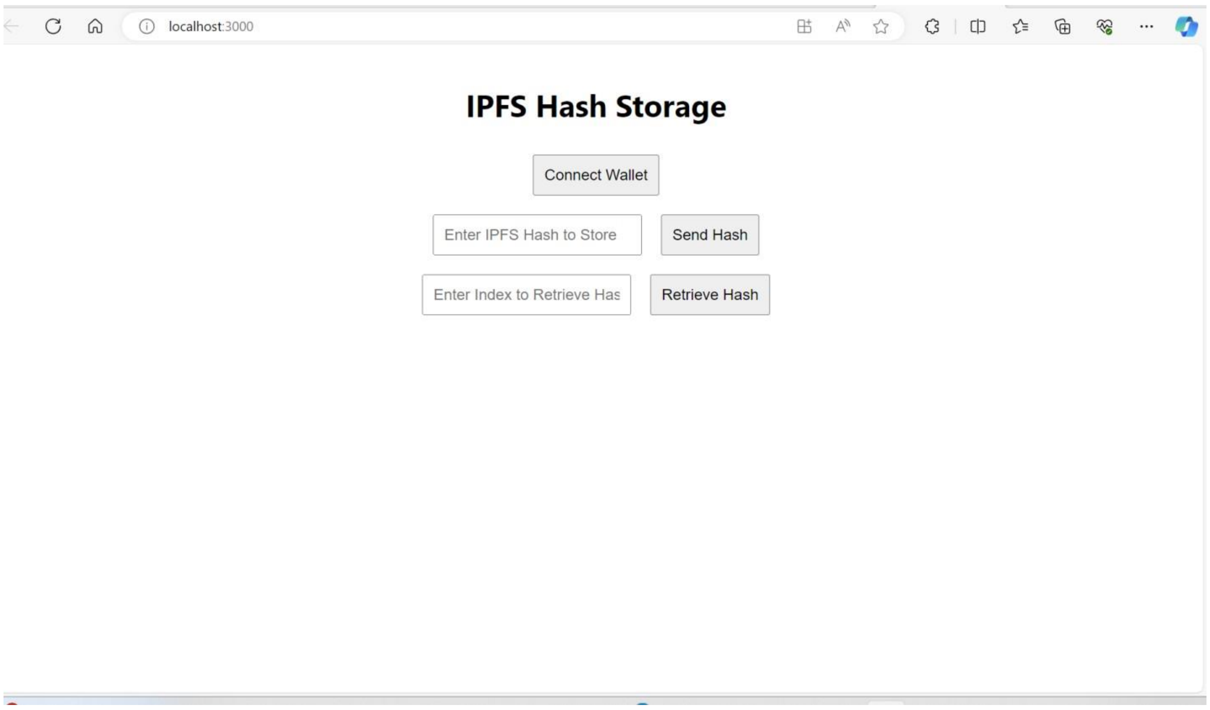
Upload a file / data set:

Then copy cd of that file (simply hash)



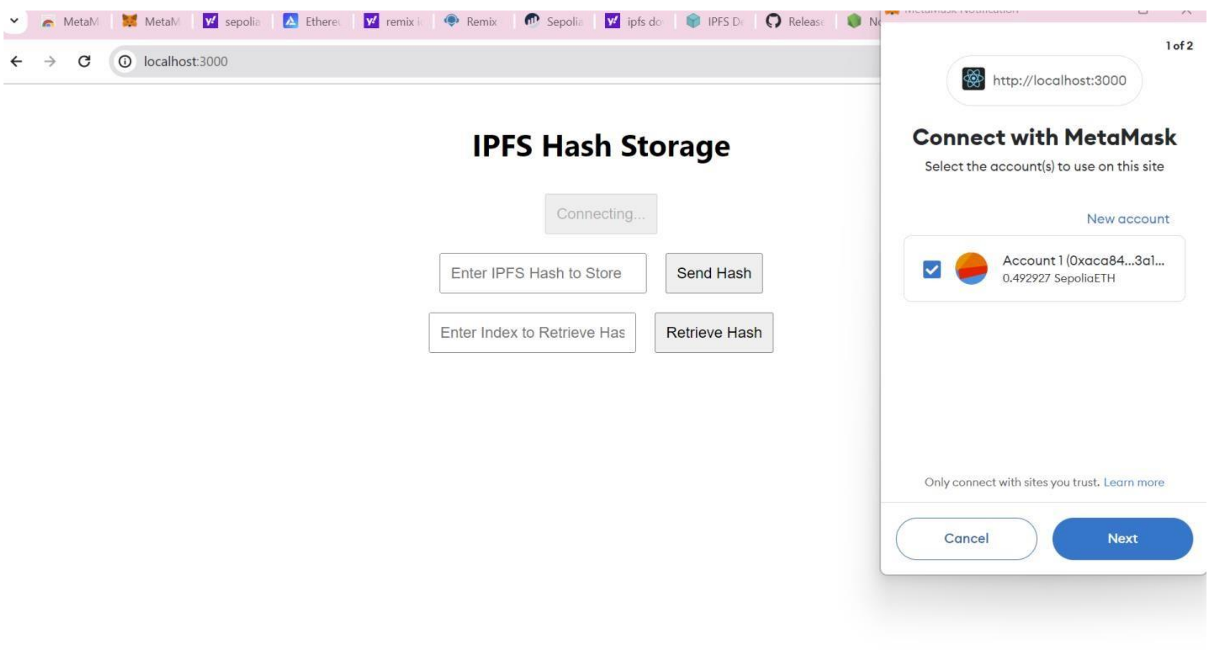
CAT IMAGE.jpg
QmPtFrbGGhj7i2SBtL4WjptdxQoveTV5VNcGmRNFVRi7PC
96 KiB

Imported 1 item
CAT IMAGE.jpg | 96 KiB ✓



IPFS Hash Storage

Connect wallet



localhost:3000

IPFS Hash Storage

Connecting...

Enter IPFS Hash to Store | Send Hash

Enter Index to Retrieve Has | Retrieve Hash

1 of 2

http://localhost:3000

Connect with MetaMask

Select the account(s) to use on this site

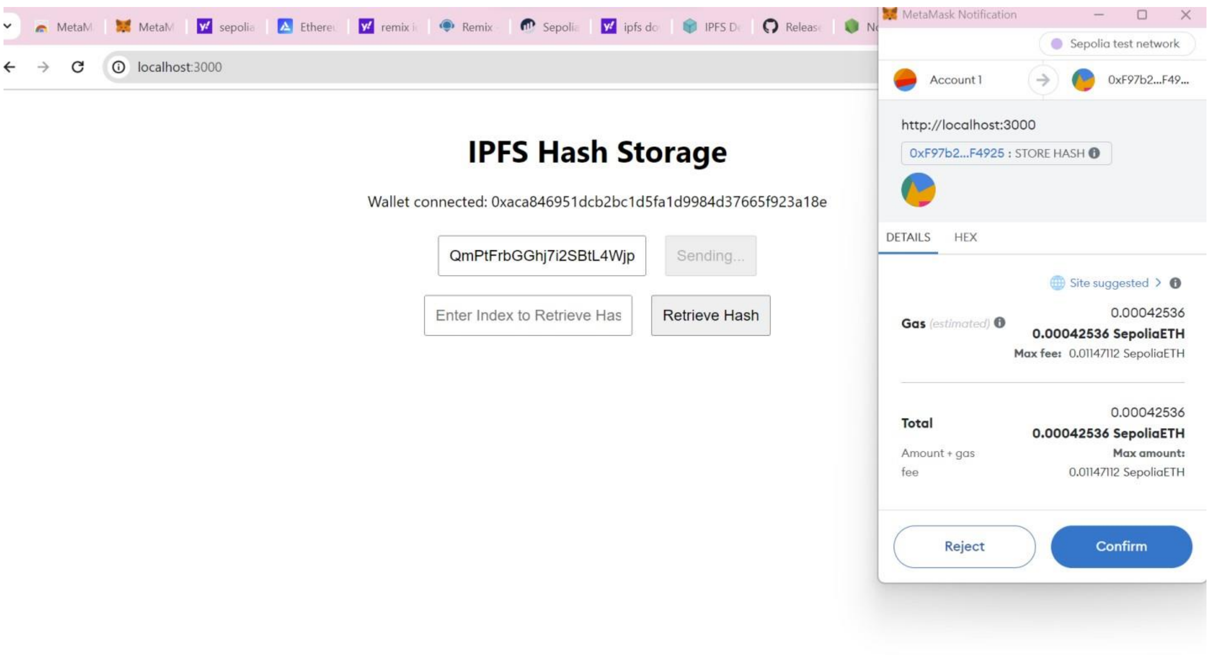
New account

Account 1 (0xaca84...3a1...
0.492927 SepoliaETH

Only connect with sites you trust. Learn more

Cancel | Next

Wallet connected and Send hash amount reduction from wallet



localhost:3000

IPFS Hash Storage

Wallet connected: 0xaca846951dcb2bc1d5fa1d9984d37665f923a18e

QmPtFrbGGhj7i2SBtL4Wjp | Sending...

Enter Index to Retrieve Has | Retrieve Hash

MetaMask Notification

Sepolia test network

Account 1 → 0xF97b2...F49...

http://localhost:3000

0xF97b2...F4925 : STORE HASH

DETAILS HEX

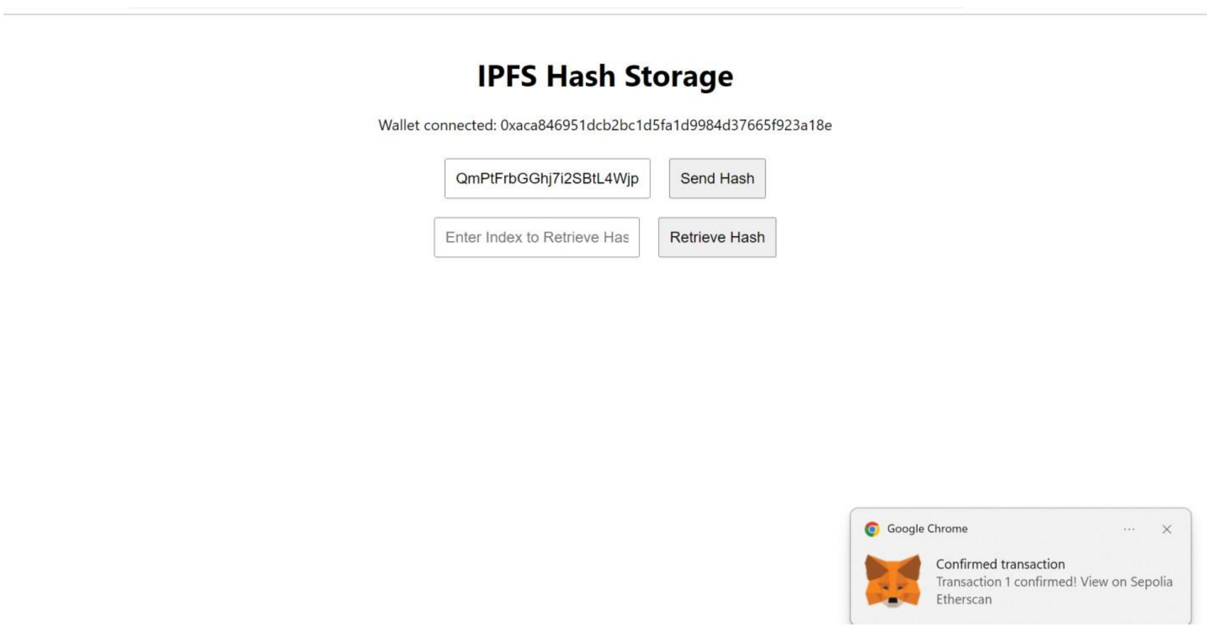
Site suggested

Gas (estimated) 0.00042536
0.00042536 SepoliaETH
Max fee: 0.01147112 SepoliaETH

Total 0.00042536
Amount + gas fee 0.00042536 SepoliaETH
Max amount: 0.01147112 SepoliaETH

Reject | Confirm

Transaction confirmed:

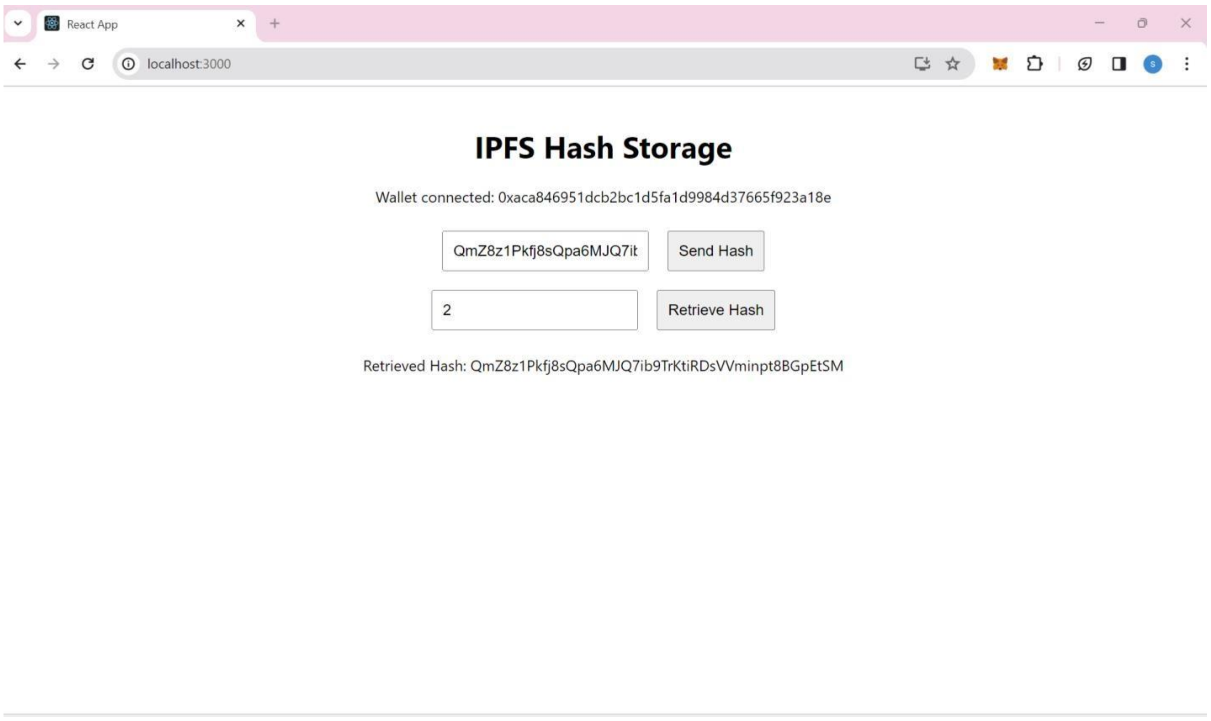


IPFS Hash Storage

Wallet connected: 0xaca846951dcb2bc1d5fa1d9984d37665f923a18e

Confirmed transaction
Transaction 1 confirmed! View on Sepolia Etherscan

Finally hash value retrieved



IPFS Hash Storage

Wallet connected: 0xaca846951dcb2bc1d5fa1d9984d37665f923a18e

Retrieved Hash: QmZ8z1Pkfj8sQpa6MJQ7ib9TrKtiRDsVVminpt8BGpEtSM

7.CONCLUSION

In conclusion, our project represents a ground breaking endeavour in the realm of cloud security, addressing the intricate challenges of safeguarding data in dynamic and decentralized environments. The deployment of the AES-256-GCM encryption algorithm provides an unparalleled level of data security, fortifying our system against unauthorized access. The incorporation of the IPFS blockchain establishes a decentralized and tamper-resistant framework, enhancing data integrity and fostering trust in the storage system. Leveraging machine learning models within the Hash Value Classification Module introduces a proactive approach to identifying potential threats, bolstering our overall security posture. Furthermore, the seamless integration of MetaMask empowers users to interact securely with Ethereum DApps, ensuring a user-friendly and trustworthy experience in cryptocurrency transactions. The synergy between these technologies results in a holistic solution that not only meets the contemporary demands of cloud security but also adapts to the evolving landscape of cybersecurity challenges. By combining cutting-edge encryption, blockchain, and machine learning techniques, our project offers a comprehensive and innovative approach to enhancing data security within the dynamic cloud environment.

8.BIBLIOGRAPHY

Rajput, Snehal, J. S. Dhobi, and Lata J. Gadhavi. “Enhancing Data Security Using AES Encryption Algorithm in Cloud Computing.” Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 2.

Ghavghave, Rashmi S., and Deepali M. Khatwar. “Architecture for Data Security In Multicloud Using AES-256 Encryption Algorithm.” International Journal on Recent and Innovation Trends in Computing and Communication: Volume 3.

Sakr, R. H., F. Omara, and O. Nomir. “An Optimized Technique for Secure Data Over Cloud OS.” International Journal of Emerging Trends & Technology in Computer Science *(IJETTCS)*

“Cloud Data Security Using Hybrid Encryption with Blockchain”:

This paper focuses on cloud data security using hybrid encryption (RSA + proxy-re-encryption) with Blockchain Technology (BCT). It explores high data security, decentralization, transparency, and minimal operational costs.

Kaur, Ranjit, and Raminder Pal Singh. “Enhanced Cloud Computing Security and Integrity Verification via Novel Encryption Techniques.” SSRG International

Pathak, Namita N., and Prof. Meghana Nagori. “Enhanced Security for Multi Cloud Storage using AES Algorithm*.”* International Journal of Computer Science and Information Technologies