National College of Ireland

Enhancing Cloud Security by integrating optimized Symmetric Key Encryption Algorithm in Ethereum Blockchain

MSc Research Project
Cloud Computing

Sarang Shrikhande
x22202226

School of Computing
National College of Ireland

Supervisor: Prof. Diego Lugones

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Sarang Shrikhande
Student ID:	x22202226
Programme:	MSc in Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Prof Diego Lugones
Submission Due Date:	12th August 2024
Project Title:	Enhancing Cloud Security by integrating optimized Symmetric Key Encryption Algorithm in Ethereum Blockchain
Word Count:	7425
Page Count:	24

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sarang Shrikhande
Date:	16th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Cloud Security by integrating optimized Symmetric Key Encryption Algorithm in Ethereum Blockchain

Sarang Shrikhande
x22202226

Abstract

Enhancing the Lightweight Bcrypt Symmetric Key (LBSK) encryption algorithm with a data sensitivity classification system provides a balanced solution to modern cloud-based data security challenges. Today, the ever-increasing reliance on cloud service providers (CSPs) for data encryption has led to organizations facing risks such as vendor lock-in and centralized control over their security infrastructure. This method divides data into groups that are sensitive and non-sensitive in order to address these problems. Sensitive data, which demands higher security, is encrypted using the LBSK Encryption algorithm. Meanwhile, non-sensitive data benefits from the Blowfish Encryption Algorithm. This dual approach ensures encryption strength to data sensitivity, optimizing both security and performance. Sensitive data receives enhanced protection through heavier encryption, ensuring robust security, while non-sensitive data is encrypted more efficiently, reducing computational overheads and improving the overall performance. By managing location of encryption keys and file path on a decentralized Ethereum blockchain, this method also decreases dependency on CSPs and provides more control to the user for its data. The findings suggest that this adaptive encryption strategy achieves noteworthy performance improvements for non-sensitive data, with faster encryption times and reduced resource consumption. At the same time, it maintains high security for sensitive data. This solution effectively balances the need for strong data protection with the efficiency demands of modern cloud computing, providing a flexible and practical approach to managing data encryption based on sensitivity.

Keywords- Cloud Security, LBSK encryption algorithm, Blowfish Algorithm, Ethereum blockchain.

1 Introduction

The increasing growth of technology is quite exciting, and at the centre of many developments is cloud computing when it comes to the organization's use of data. Cloud computing means the use of services that are made available over the internet or other 'clouds.' The transition from having internal IT environments or on-premises services to cloud-based services provides virtually limitless elasticity, adaptability, and affordability Chang et al. (2010).

Integral to cloud computing are the data centres; data centres are physical structures that hold computers, networking equipment for accumulating extensive data. These

data centres are situated in a manner that gives them the best shot at phenomenal service, diversity, and safety for the information stored in them. It is at this juncture that CSPs play a central function within this setup. They operate large data centres networks and provide several cloud services to companies and clients. Current CSPs such as AWS, Microsoft Azure, and Google cloud remain as the market giants mainly because of the secure, reliable, and elastic cloud solutions for numerous services and applications Choudhary et al. (2022).

It also means that the power and importance of CSPs are increasing proportionally to the migration of organizations' workloads to the cloud. Since there are more companies relying on the cloud, CSPs' authority and territory keep on enlarging. This centralization tendency is changing the very nature of the IT industry and transforming several giant CSPs into key players controlling the digital environment of millions of companies around the world. This dynamic and its implications, therefore, form the basis for this paper.

1.1 Motivation and Background

The adoption of cloud services has created a trend of centralization of cloud services where most of the cloud service providers are dominated by very few CSPs today. This centralization is explained to some extent by the fact that a large amount of capital is needed to construct and support the enormous data centres and networks that are at the core of cloud applications. There are pros and cons to this and some of the advantages include reduced costs in terms of scaling, better and reliable services, and better security configurations but on the flip side there are issues to do with data lock in, sovereignty, and the fact that centralization of this important internet function puts it in the hands of a few players Al Nafea and Almaiah (2021).

Cloud computing is the new advancement in the process of data storage and utilization, enabling higher efficiency and opportunities for companies and users. However, these advantages are always accompanied by serious concerns, including those related to the protection of information and its owners. The recent rise in cybercrimes and hacking incidents also emphasizes the importance of strong security mechanisms to guard information assets stored in the cloud environmentHussain et al. (2023). Storing data in the cloud has become popular because it offers solutions that are scalable, flexible, and cost efficient. Many of these CSPs which include AWS, Microsoft Azure, and Google Cloud are not only data storage spaces but also provide top-notch cryptography services for data protection. While this centralization must be helpful since it eases the processes of encryption and storage, it poses a huge reliance and vulnerabilities on CSPs. Operational risks that organizations receiving encryption from these providers encounter include loss of data, security control, vendor lock-in, and heightened risk in instances where the vendor's security is weak Raj et al. (2021). Hence a number of organizations are adopting blockchain technology for same data storage and you can see in Figure 1 how blockchain's business value addition is growing year-on-year.

1.2 Project Specification

This research looks to enhance the existing state-of-the-art LBSK encryption algorithm implemented in the public blockchain. The implementation relies on the Ethereum Block-

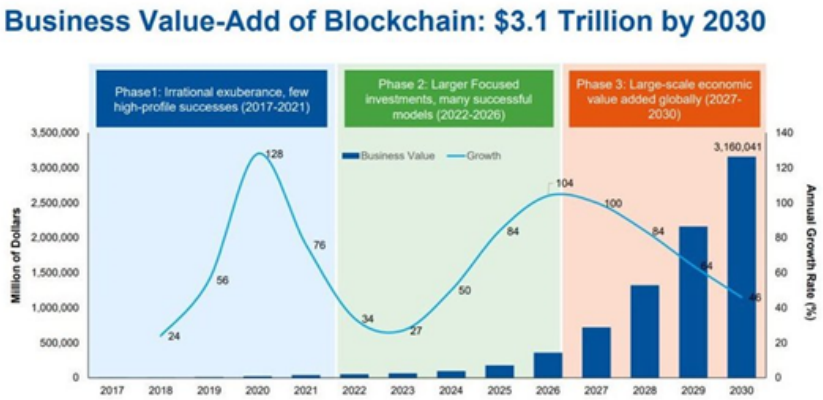


Figure 1: Business Value-Add of Blockchain: $3.1 Trillion by 2030 Beloglazov and Buyya (2015)

chain. We will use the Ethereum test net to simulate the network and the number of nodes and smart contracts. Below is the research question for the proposed implementation.

1.2.1 Research Question

How can the Ethereum blockchain with advanced symmetric key encryption algorithms, beyond Lightweight Bcrypt, further enhance the security and efficiency of cloud-based data storage and transactions, while maintaining or improving upon the system's performance metrics?

In cloud security the passwords and data should be encrypted and not stored in plaintext form. Several algorithms help in this, and our work is based on improving the LBSK on the public blockchain. Evidently, the first objective is based on enhancing the primary performance parameters of LBSK algorithm which include security rating, throughput, delay, and access control on public blockchain such as Ethereum blockchain. The current iteration of the LBSK encryption boasts impressive metrics: a 94.5% security rating, 700 transactions per second (TPS), a 2-second authentication delay for 300 MB of data, and 95.5% access control. Although these figures are notable, the security rating isn't a perfect 100%, indicating potential vulnerabilities that could be exploited. Thus, enhancing these metrics is essential.

In my case, I anticipate that by categorizing user data into different types, the outcome obtained from the existing LBSK algorithm will be boosted and it is expected to increase efficiency while at the same time maintaining security in the complete encryption process on the Ethereum public blockchain.

1.2.2 Research Objective

The research question addresses the following research objectives: Objective 1: Implementing the LBSK and Blowfish encryption algorithms basis whether the data is classified as sensitive or non-sensitive. Objective 2: Creating smart contracts to store the encryption keys location in AWS Lambda securely and the Base64 encoding of the file path on

the blockchain. Objective 3: Storing the encrypted files onto the cloud and ensuring to store the location of keys and encoded file path on blockchain using the smart contract which will be used for decryption. Objective 4: To measure the performance metrics like the efficiency, latency and quantitatively analyse how the integration of encryption algorithms will affect the performance and the security standards.

1.2.3 Research Contribution

In this thesis we dive into enhancing the cloud storage solutions within the versatile Ethereum blockchain framework. We explore a pressing research question stated above. Our approach is to manually classify data into sensitive and non-sensitive categories, tailoring encryption methods accordingly. This strategy aims to strengthen security for sensitive data and streamline processing for less critical information. We hypothesize that this selective encryption technique will improve overall system security and efficiency which are the key factors in mitigating the risks associated with data breaches and unauthorized access in cloud computing. By varying encryption intensity based on data sensitivity, we not only boost security measures but also enhance operational efficiency, a crucial advantage for real-time applications in cloud environments.

2 Related Work

The research that has been done so far on the Ethereum and other security features, such as blockchain encryption methods, will be covered in this part. Any gaps in the research will be noted.

2.1 Blockchain Technology

The two well-known public blockchain systems, Bitcoin and Ethereum, are contrasted in Rankhambe and Khanuja (2019) study, which also emphasizes the fundamental distinctions in functionality between them. The first decentralized digital currency, or bitcoin, was unveiled in 2008 by an unidentified person going by the name Satoshi Nakamoto. Its purpose was to enable peer-to-peer transactions without the use of a bank or other central authority. Rankhambe and Khanuja (2019) highlight how the Proof of Work (POW) consensus mechanism and the SHA256 cryptographic method help Bitcoin address problems such as double spending. Founded by Vitalik Buterin, Ethereum is regarded as more than just a cryptocurrency; it is a platform for creating and hosting smart contracts and decentralized apps (DApps) using the Ethash consensus process, which employs the Keccak SHA-3 cryptography function Buterin (2016).

2.2 Public Blockchain vs Private Blockchain

Cachin et al. (2016) provides details of operational mechanisms in Hyperledger Fabric like the use of Docker containers for chaincode which is the language of smart contracts in the Fabric and the distinction between validating and non-validating peers. This structure improves the security and integrity of transactions and ensures that only authorized nodes can participate in the consensus process. It is possible to control access and maintain data confidentiality with this setup in a secure cloud environment.

Aggarwal and Kumar (2021) discusses the broader umbrella of the Hyperledger projects and puts forward the diverse application and flexibility of the Hyperledger Fabric to meet various industry demands. It entails the use of Hyperledger for private transactions within an organization and highlights its capacity to handle sensitive information securely.

While Muthe et al. (2020) describe a novel architecture with the use of Ethereum blockchain and the Inter Planetary File System (IPFS) for the new internet. This setup intends to eliminate the problem of centralized control that usually results in data manipulation and privacy. With proxy re-encryption and zero knowledge proof, the system is able to achieve private and secure transaction of data without revealing the original data. It also helps in faster information transfer as it is evident from the proposed distributed architecture, and they have lower latency than other methods such as HTTP.

He et al. (2023) discusses the enhancing charitable donations with the help of Ethereum blockchain and searchable encryption. Traditional charity systems can lack transparency and they can be easily falsified. If all the donation details are recorded on the Ethereum blockchain, all transactions associated with this system are transparent and immutable, increasing donors' trust. Searchable encryption can help to search through encrypted data and, at the same time, preserve donors' and beneficiaries' identities. Also, smart contracts simplify the process of donations, which minimizes human intervention in this process and improves the level of transparency and security of charities.

Both these papers focus on applying Ethereum blockchain to enable security, transparency, and efficiency in their respective domains. Muthe et al. (2020) focuses more on the creating a decentralized internet infrastructure, addressing broader issues of data control and privacy on the web side of things. While He et al. (2023) discusses a specific application of blockchain to solve the lack of transparency and trust issue in philanthropic donations. Both of these systems highlight a need for further research like using stronger encryption systems and integrating them with the existing technologies.

2.3 Cloud security analysis using Blockchain Technology

A system for safe exchange of medical records with blockchain and searchable encryption technologies is covered by Tang et al. (2021). It combines searchable encryption with blockchain technology to protect and simplify the exchange of electronic medical records across various entities. To guarantee that only authorized users can access and confirm the integrity of medical records, the created approach stores a worldwide encrypted search index and a verification index on a blockchain network. By streamlining the blockchain's storage and management of medical data, the approach outperforms conventional cloud-based searchable encryption, emphasizing security without sacrificing efficiency. The capacity to manage access through fully functioning access control, which guarantees that patients and institutions can maintain data privacy securely, is one of the proposed scheme's key advantages. The scheme's feasibility within a local Ethereum network is confirmed by the experimental results, which demonstrate effective data sharing with low overhead.

A secure public-key encryption system with keyword search (PEKS) that can withstand keyword guessing attacks (KGA) is presented by Zhang et al. (2019). It explains a

unique technique called SEPSE that uses superior key management procedures along with blockchain technology to mitigate these vulnerabilities. The five methods are included for cloud storage security. Cryptographic parameters are generated, and the foundation is laid by the Setup algorithm. It comes with four key server setups as well as public and private keys. The main algorithm, PEKS, manages data encryption using keywords for searches. Data is encrypted so that keywords can still be used to search through it without disclosing the information. Keyword-based requests to retrieve data are generated by the trapdoor algorithm. The KeyRenew method makes it easier for keyword servers to update their keys on a regular basis because over time, these keys may become weaker or more vulnerable. Tang et al. (2021)'s study directly addresses our research topic, which concerns the possibility for enhanced cloud-based data storage and transactions through the use of improved symmetric key encryption in conjunction with private blockchain technology. The ideas of searchable encryption, data integrity, and limited access are relevant to any blockchain implementation even if they use Ethereum and concentrate on medical data. The paper's shortcomings, like its lack of a thorough analysis of how various encryption algorithms affect performance metrics, present a clear opportunity to discuss further how various, more sophisticated encryption techniques might be incorporated into blockchain systems like Ethereum to improve cloud security and efficiency.

Zhang et al. (2019)'s paper is vital to our work since it goes beyond conventional encryption techniques and presents a blockchain-enhanced framework that might be customized for Ethereum. With these variations, the ideas and conclusions from the study based on Ethereum can still offer insightful information for our investigation into Ethereum blockchain. The concepts of improving data security using blockchain and improved encryption are applicable to our research with public blockchain, even though the underlying blockchain technology is different.

2.4 Decentralized Storage in Cloud

Shah et al. (2020) paper deals with the system where data is encrypted and distributed with the help of Inter Planetary File System (IPFS), while it is protected by the blockchain and smart contracts with the application of cryptocurrency payment. In Doan et al. (2022), the main focus is made upon the modularity to address socio-technical issues and the potential strategies that need focus in the course of the future research on the decentralized storage. Both papers are clear on the decentralization in enhancing the security and availability of data although this is done from a different technological angle.The two research papers, Shah et al. (2020) and Doan et al. (2022) explore decentralization, stressing on security, and data retrievability.

The nature of centralized cloud services presents security and privacy challenges and threats since such providers manage vast amounts of data and it becomes a primary point of focus for attackers and possibly could be misused by third parties.Sriram (2022) Blockchain technology provides a solution to the problem by decentralizing data storage where the data is stored with the help of nodes, and it cannot be tampered or changed as it is encrypted with the help of smart contracts. Basically, this causes the approach to be safer, more reliable and more secure compared to the central approach which is always risky in terms of storing data. Adding the utilization of blockchain with IPFS can also

help expand on these advantages by creating a strong barrier against cyber attacks along with the strength of distributed storage.

2.5 Symmetric Key Encryption in Cloud

Similar to text data, the image data collected by different sensors and other Industrial Internet of Things(IIoT) devices can be encrypted and stored on the blockchain Khan and Byun (2020). They used an encryption method to secure the image data itself with blockchain technology for data integrity and access control. This dual technique makes sure that without the matching decryption keys, data stays unintelligible even if it is intercepted. Image pixels are converted into encrypted values during the process, and these values are then tracked and controlled via blockchain transactions. Guerrero-Sanchez et al. (2020) suggest that symmetric key encryption techniques can be implemented in blockchain technology to enhance security in wireless sensor networks (WSNs). However, there is a research gap here as neither of the articles specifically address cloud-based data storage and transactions in the context of employing blockchain.

Kumar et al. (2023) discusses a novel approach to enhance data security in cloud computing through a combination of compression and encryption techniques. In order to make obtain optimal computational process combined with effective data protection, the proposed method relies on Identity-Based Encryption (IBE) and the LZ4 compression algorithm. This is because the specific method of putting the data in an encrypted form and then compressing it leads to the reduction of storage space and a high level of protection against unauthorized access. The system further employs a two-factor authentication approach to widen protection even more, with the major focus being placed on the protection of data from access by unauthorized persons and secure transfer of data through the cloud medium.

Gan et al. (2021) deals with issues related to privacy and security of data is multi-client cloud computing environment. To achieve this, the authors present forward private Secure Searable Encryption (SSE) with a new data structure of the private link and the public search tree, which makes use of XOR-homomorphic functions. These innovations refer to settings that must prevent the newly entered documents from disclosing earlier search terms, thus guarding against file injection attacks. The idea of the scheme is to provide fast searching for documents as well as fast updating of documents, all the while being able to operate under multiple clients at low computational cost. Based on experimental simulations, the effectiveness of the proposed scheme is higher than that of other approaches and its security is enhanced.

Both papers Gan et al. (2021) nad Kumar et al. (2023) aim to improve cloud data security but from different angles: while one focuses on the searchable encryption for multiple clients, the other increases the efficiency of data compression and encryption. Some of the research issues associated with the extension of the fundamental results include the trade-off of the computational overhead in Multi-Client SSE and the real-time performance in large-scale cloud settings using state-of-art cryptographic processing capabilities.

Table 1 shows the summary of the related work.

Article	Methodology	Research Domain	Achievements	Limitations	Differentiation
Rankhambe and Khanuja (2019)	Comparison of Bitcoin and Ethereum, POW vs. Ethash	Blockchain	Highlighted consensus and cryptographic differences	Limited to Bitcoin and Ethereum	Focus on Bitcoin and Ethereum mechanisms
Cachin et al. (2016)	Analysis of Hyperledger Fabric's mechanisms	Public vs Private Blockchain	Improved transaction security	Focused on Hyperledger only	Detailed Hyperledger security features
Aggarwal and Kumar (2021)	Overview of Hyperledger's flexibility	Public vs Private Blockchain	Secure handling of sensitive info in organizations	Limited to Hyperledger	Exploration of Hyperledger's applications
Muthe et al. (2020)	Proposed Ethereum + IPFS architecture	Public vs Private Blockchain	Faster, secure transactions, low latency	Limited by current encryption	Ethereum for decentralized internet
He et al. (2023)	Ethereum blockchain with searchable encryption	Charitable Donations	Improved transparency, security in donations	Need for stronger encryption	Blockchain for transparent philanthropy
Tang et al. (2021)	Blockchain + searchable encryption for medical records	Cloud Security	Secure, efficient medical data exchange	Lacks encryption algorithm analysis	Blockchain for medical data security
Zhang et al. (2019)	PEKS system with SEPSE for cloud security	Cloud Security	Mitigated keyword guessing attacks	Focused on cloud, not blockchain	Advanced key management for cloud security
Shah et al. (2020)	IPFS + blockchain for decentralized storage	Decentralized Cloud Storage	Enhanced security, retrievability	Complex implementation	IPFS + blockchain for data security
Doan et al. (2022)	Modularity, socio-technical issues in storage	Decentralized Cloud Storage	Addressed security, availability issues	Focus on socio-technical aspects	Socio-technical challenges in storage
Sriram (2022)	Blockchain for decentralized data storage	Decentralized Cloud Storage	Secured data against tampering	Limited platform exploration	General benefits of blockchain storage
Khan and Byun (2020)	Image data encryption in IIoT with blockchain	Symmetric Key Encryption	Secured image data integrity	Limited to IIoT	Blockchain for IIoT image security
Guerrero-Sanchez et al. (2020)	Symmetric key encryption in WSNs with blockchain	Symmetric Key Encryption	Enhanced WSN security	Doesn't address cloud explicitly	Focused on WSNs with cloud implications
Kumar et al. (2023)	Compression + encryption for cloud security	Symmetric Key Encryption	Improved security, storage efficiency	Computational overhead in large-scale	Compression + encryption for cloud security
Gan et al. (2021)	SSE with XOR-homomorphic functions in clouds	Symmetric Key Encryption	Fast, secure multi-client document search	Potential computational overhead	Enhanced privacy, security in multi-client clouds

Table 1: Summary of Related Work

3 Methodology

In the last few years, we have seen the growing trend of using Cloud Computing and that too primarily from the top cloud service providers like Amazon Web Services, Google Cloud Platform and Microsoft Azure[1]. To tackle this over dependence, the research methodology will carry out encryption of our stored data and the store the location of encryption keys and filepath on the blockchain and use the AWS Cloud services like S3 bucket to store encrypted files. This approach aims to give more control to user over how his data(sensitive or non-sensitive) is stored in the cloud.

In this section, we present a methodology for the solution to our research question whereby we will aim to reduce the dependence on the CSPs and implement our encryption algorithms with the help of Ethereum blockchain and analyse the gaps identified in the Section 2. The proposal also takes inspiration from Devi et al. (2015)'s work in evaluating security and efficiency of different encryption algorithms. We will also discuss the challenges that we faced during our implementation.

3.1 Data Classification based on sensitivity

We will start with gathering data samples that will be used for classification. We will then define the criteria for classifying data as sensitive or non-sensitive based on the context.

Sensitive Data Types	**Non-Sensitive Data Types**
Social Security Number (SSN)	Name
Credit Card Information	Email Address
Bank Account Numbers	Phone Number
Passport Numbers	Mailing Address
Driver's License Numbers	Date of Birth

Table 2: Sensitive and Non-sensitive data types

The criteria to identify data as sensitive includes:

- Personal identifiable information (PII) like social security numbers.
- Financial data like account numbers or transaction details.
- Health information protected under regulations like Health Service Executive (HSE).
- Confidential business information like trade secrets or strategic plans.
- Legal documents.

Non-sensitive data includes things which are already available in the public domain. The criteria to identify data as non-sensitive includes:

- Name
- Email
- Address

[1]https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/

- Phone Number
- Date of Birth
- Job Title

Currently, we have provided the user with two different text box inputs for sensitive or non-sensitive respectively. The user himself marks it whether the data entered is sensitive or not. Automating this process can be a part of future scope.

3.2 Encryption and Decryption Process

3.2.1 Encryption Process

After classification of data based on sensitivity, we will encrypt the data using the appropriate algorithm.

• **AWS Lambda and Elastic Container Registry:** Lambda is a serverless compute service by AWS where we run our encryption and decryption code for LBSK and Blowfish encryption algorithm Barrak et al. (2024). For Blowfish algorithm, the dependencies required to run it required a size that exceeded Lambda's size limit of 250 mb. So, for this reason, the Blowfish algorithm was hosted on a docker image in AWS Elastic Container Service. This was one of the challenges faced during the implementation. In the next section, we will discuss how we tackled this problem. Both of these algorithms are triggered when user uploads a text file.
• **LBSK Encryption Algorithm:** It is a symmetric block cipher which is known for its security. Apply the LBSK algorithm to sensitive data.
• **Blowfish Encryption Algorithm:** Applied to non-sensitive data.
• **File Upload:** Upload the encrypted files to an Amazon S3 bucket.

3.2.2 Decryption Process

When the user wants to read the encrypted file, following is the process to decrypt and download the file:

•**File Retrieval:** We will use encoded file path stored in the smart contract to get the encrypted file that needs to be retrieved from Amazon S3.
•**Decryption Lambda Function:** A backend API with parameters like the decryption key and encoded file path to the S3 bucket where it is stored(from the smart contract) will trigger a Lambda function to decrypt the file.
•**File Download:** Provide the decrypted file to the user for download.

3.2.3 Key rotation in the Decentralized System

Key rotation in the decentralized system was the challenging part. We have kept the key in AWS lambda function variables. We can manually rotate keys in lambda. Storing keys directly on the public blockchain was not an option as it is not secure. The location of the encryption keys and the encoded file path are securely stored by the smart contract on the Ethereum blockchain. The keys are not stored directly on the blockchain but securely handled within an AWS Lambda function. We can create a hash of the encryption keys to store them more securely and this can be a part of the future work.

3.3 Integration with Smart Contracts

Once the file is encrypted and uploaded onto the AWS S3 bucket, the key will be stored in the Lambda function and the encoded path of the key will be stored in the smart contract.

We will develop a smart contract for storing the location of encryption keys which are generated after applying the encryption algorithm through AWS lambda function.

We will use the Ganache Truffle suite which will host our private Ethereum blockchain for deploying the smart contract Satyam et al. (2023). The smart contract is to be written in Solidity programming language and we haveinstalled the Ganache Truffe suite on an EC2 instance.

3.4 Tools and Equipments used:

3.4.1 Ethereum Test Network

We will use the Ganache from Truffle Suite for launching of the Ethereum test network[2]. It is installed on an EC2 instance which is easier to run our own test blockchain since the blockchain development tools are quite flexible[3]. This setup provides us with an isolated environment to run test without putting in real money to buy trade ether for smart contracts.

3.4.2 AWS Lambda Function and S3 bucket

AWS Lambda function is one of the core components of the proposed encryption and storage solution that is supposed to be triggered right after user uploads a text file[4]. During its execution, the user decides whether the data uploaded is sensitive or not and then the file will be encrypted with Lightweight Bcrypt Symmetric Key Encryption or Blowfish Encryption Algorithm. The selected algorithm guarantees that the file will be protected according to its level of sensitivity. Then, the Lambda function saves the encrypted file into the AWS S3 bucket to take advantage of S3's capability in secure and scalable data storage.[5].

We will store the location of decryption key and the encoded S3 file path in a smart contract on the Ethereum blockchain to maintain a secure and immutable record. This smart contract guarantees the management of the encryption keys, as well as the storing of file paths that cannot be changed or deleted. In this way, the idea of the system is based on the possibility of using the decentralized and transparent feature of the blockchain to ensure that only the authorized person will have the ability to open the encrypted files. Such integration of AWS Lambda, S3, and Ethereum blockchain lays the foundation of a reliable and secure system to process the sensitive data while maintaining high availability and the security measures.

[2]https://archive.trufflesuite.com/docs/truffle/quickstart
[3]https://www.youtube.com/watch?v=wA2ltwauvRw
[4]https://docs.aws.amazon.com/lambda/
[5]https://docs.aws.amazon.com/s3/

4 Design Specification

Reducing over dependence on the Cloud Service Providers as well as empowering the users to secure their own data is a challenge. In this section, we will discuss the techniques as well as the architecture for our underlying implementation and secure storage solution. The system that we proposed leverages the combined power of blockchain technology, cloud services as well as the encryption algorithms to ensure the user data is handled in a secure and efficient manner. The current state of the art system employs a similar blockchain integration for securing user data Banushri and Karthika (2023). The goal is to take it even further by segregating the data into two categories: sensitive data and non-sensitive data. The data types that are as mentioned in the Section 3.

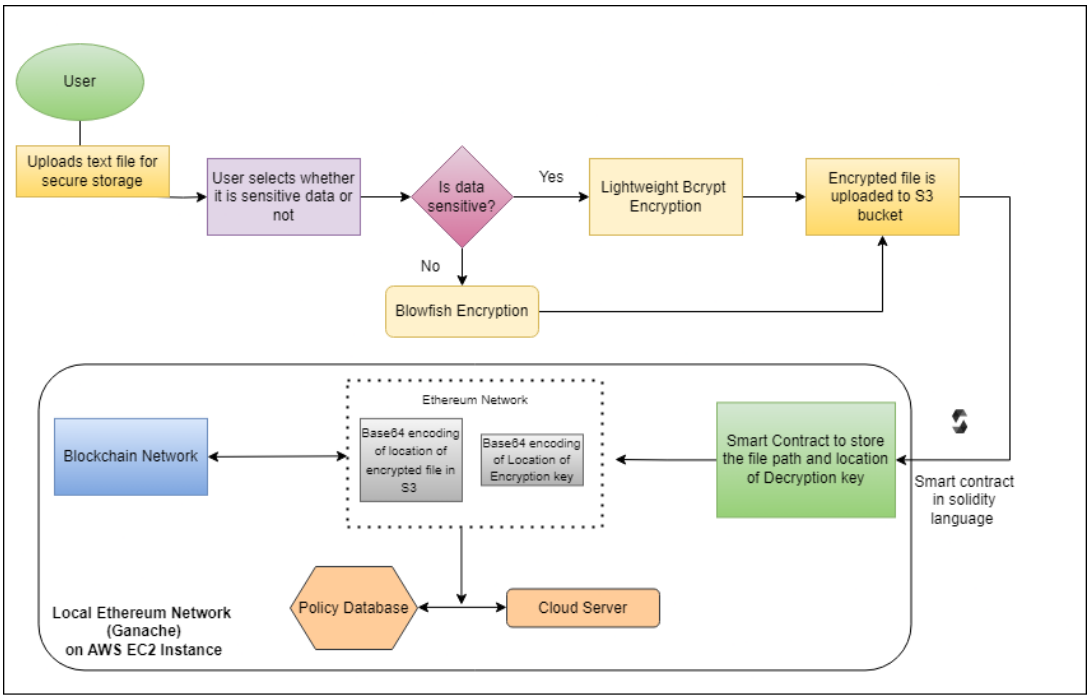


Figure 2: Proposed architecture

Our proposed architecture in the above Figure 2 shows the complete breakdown of the steps that the user will follow to secure their data and reduce the dependency on the CSPs in terms of the security of his data.

The user will first upload the data or the files which he/she wants to securely store on the cloud. A front end is created for the same to provide the user with the Graphical user interface to interact with the system. The user will have the option to choose whether to encrypt his data using either Blowfish Encryption or the LBSK encryption depending on the type of data being uploaded (Blowfish for non-sensitive data and LBSK for sensitive data).

So, the file will be first uploaded, the Lambda function that contains the encryption code will open and read the contents of the file and then begin the encryption process.

The start time and the end time of the encryption process will be recorded using the Node.js script using the 'Date' object. This data will be useful to carry out various tests and evaluations for judging how efficient our system actually is. The encrypted content of the file is then saved to two new files on S3 containing sensitive and non-sensitive data.

We also want to securely store the location or the file path in the S3 bucket onto the smart contract. Hence, when the file is encrypted and stored in S3 bucket, a smart contract runs that asks user to execute the smart contract that will store the location of encryption keys of both the files as well as the file path on the Ethereum blockchain.

Finally, the response is then returned with status 200 to indicate the successful execution. The location of Encryption key that is present in the lambda function along with the encoded file path in S3 bucket will be stored as a smart contract on the Ethereum blockchain. The smart contract for storing these details is written in the Solidity language[6].

4.1 The Encrpytion Process

1. User logs in and enters the data to be encrypted.
2. When the user clicks save, lambda function is triggered for LBSK and Blowfish algorithm.
3. LBSK algorithm runs on the sensitive data and Blowfish algorithm runs on the non-sensitive data.
4. Both these types of datas are stored in an encrypted format in the S3 bucket securely.
5. After the data is stored, a smart contract runs and the metamask wallet prompts the user to pay gas fee to store the location of encryption keys(encoded) and the file paths on the Ethereum blockchain.
6. We can check the logs and the timestamps on the EC2 instance of a new block on the Ehtereum blockchain.
7. To retrieve the data, the user clicks the download button.
8. This retrieves the files from the S3 bucket using the details stored on the smart contract.
9. The file is decrypted using the keys and User can view the downloaded file.

4.2 Ethereum Test Network

We will use the Ethereum Test network that will help us in the simulation of the Encryption and Decryption algorithms. We are using the Ganache from Truffle Suite for our smart contract development purpose.The Truffle Suite provides us with a framework for personal blockchain development and it includes various tools for the same. It can provide an environment for writing and testing smart contracts. It can also streamline various processes like compilation and deployment of smart contracts to the blockchain and manages the binaries. With the ability to perform automated testing for contracts written in Solidity, it is easy to ensure that the smart contracts work as intended. Ganache provides us with personal Ethereum blockchain for development purpose and to test our smart contracts. It can mine transactions instantly as it uses test Eth to test the functionality

[6]https://docs.soliditylang.org/en/v0.8.26/

of the smart contracts. It is easily customizable with parameters like block time, gas price, etc to have different simulations. We have deployed the Ganache on AWS EC2 instance.

Hence with the help of these development tools, a smart contract is created to store the location of the file path of the encrypted file in the AWS S3 bucket and the location of the key that will be used to decrypt the data. This will ensure that unauthorized access to the file and decryption key is avoided as the key is not directly stored on the blockchain but within AWS lambda function. This is especially helpful as we are using a public blockchain in the form of Ethereum. This way of storing the encoded file path and location of keys is much safer.

4.3 Docker Containerization

Docker containerization is used for a work around in our implementation part. We eliminated the common "working on single machine" problem, as docker provides a uniform deployment environment. The Blowfish encrpytion algortihm had dependencies of larger size than what is permitted on the AWS Lambda function. AWS Lambda permits maximum Deployment package size of 50 mb when compressed and when the package is unzipped, the limit is 250 mb. Due to this technical limitation, we are creating a Docker image and of Blowfish algorithm with all the dependencies and libraries packaged within it and pushed the Docker image in the AWS Elastic Container Registry.

4.4 AWS Lambda functions and S3 bucket

When a user uploads a file to the system, he will have the option to declare whether the data is sensitive or non-sensitive. When this file is uploaded, an AWS Lambda function is triggered. This function applies the Lightweight Bcrypt Symmetric Key (LBSK) encryption for sensitive data or the Blowfish encryption algorithm for non-sensitive data. The function then encrypts the file and uploads the encrypted version to S3 bucket in a scalable and highly durable manner. Also, the decryption process is handled by Lambda function that retrieves the encrypted file from S3 using the base64 encoding of the file path from our smart contract and the path to where the encrpytion keys are stored securely within lambda environment, decrypts it using the appropriate key, and makes it available for download. This integration of AWS Lambda and S3 creates a robust, automated mechanism for secure data storage and retrieval which enhances the overall efficiency and security of our system. This serverless compute service from Lambda makes our system both cost-effective as well as helps in scalability.

5 Implementation

5.1 Smart contract deployment

The smart contract to store the path of the Encryption key and the base64 encoding of the file path to the AWS S3 bucket is deployed on the Ganache Truffle suite which is an Ethereum test network to deploy smart contracts.

The Figure 3 shows smart contract successfully deployed as Block number 2 on our Ethereum test Net.

Figure 3: Smart contract deployed on Ethereum Test Net

We installed the Node.js dependencies which are defined in our package.json file. This installs all the necessary packages and dependencies for the smart contract deployment.

The smart contracts were first compiled using Truffle. Truffle uses the Solidity compiler that compiles the ".sol" file in the "contracts" directory.

We then migrated our smart contracts to the blockchain using the "truffle migrate" command according to the specification written in the file "truffle-config.js".

5.2 File upload for Encryption

When the user uploads the data and clicks on save button, the data is encrypted and stored in the S3 bucket. Consecutively, he will be asked to confirm the transaction on the blockchain. For this, metamask wallet is setup to interact with our website.
We added a network in the metamask with Remote Procedure Call(RPC)
URL: http://52.91.73.245:8545/ and Chain ID: 1337.
This RPC is used to connect the Metamask wallet to the Ethereum Network and it enables us to use the Ethereum Compatible APIs. The RPC url tells the wallet where to send transactions and interact with the smart contract. When the user confirms the blockchain transaction, the encoded path of the encrypted file that is uploaded on S3 bucket and the location of the encryption keys is stored on the Ethereum blockchain.

5.3 Encrpytion and Decryption process

In the Figure 4 and Figure 5, the request is first sent to the endpoint URL to the AWS with the POST request method and type JSON. The request is asking to encrypt the data using LBSK algorithm and the text to be encrypted contains the SSN number and the Credit card number.

The same can be done for Blowfish algorithm as well. To encrypt data using Blowfish algorithm, we first encode the input text to Base64. This is done because this algorithm operates on binary data. Post encryption, the output data is in bytes that may not be printable in any text encoding. It can range from '0x00' to '0xFF' (hexadecimal) and it may include special characters. The Blowfish encryption by first encoding the data to

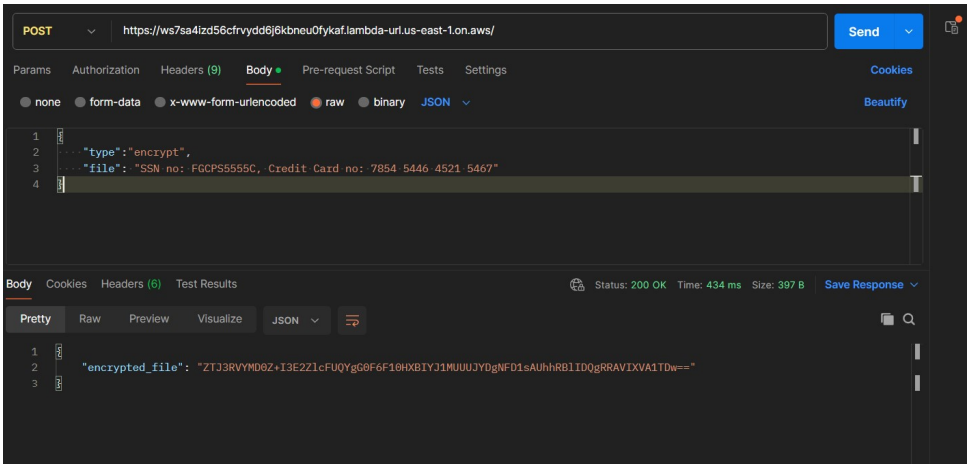


Figure 4: Data encrypted by LBSK algorithm

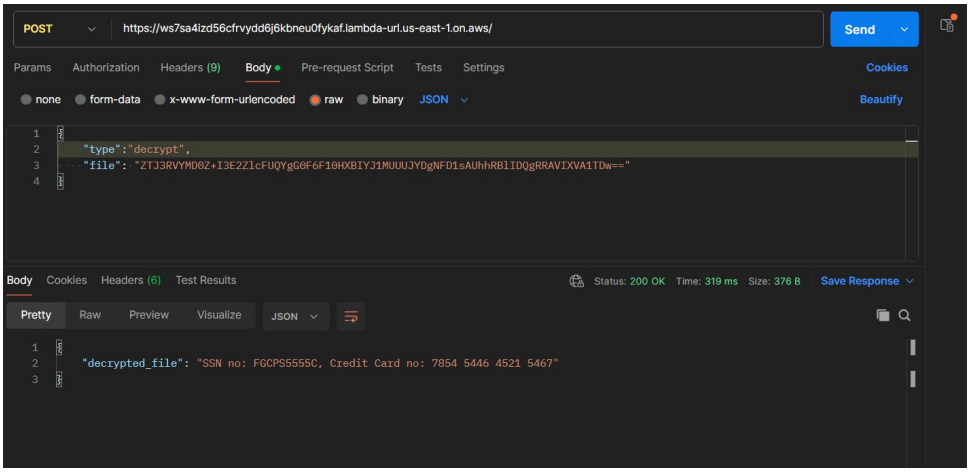


Figure 5: Data decrypted by LBSK algorithm

Base64 then encrypting it as well as decrypting the file to get the data back is shown in the Figure 6 and Figure 7 respectively.

5.4 AWS Lambda function and the encrypted data stored

As discussed in the Design Specification, AWS Lambda provided us with a serverless functionality to run our encryption and decryption algorithms based on data sensitivity. It is triggered when a text file is uploaded by the user and then it encrypts the file and stores it in the AWS S3 bucket.

The operation of Blowfish encryption and decryption uses the Cipher Block Chaining (CBC) mode. In encryption process, the algorithm first selects a random Initialization Vector (IV) and then reads the plain text from the input file. It then adds padding to make the size of plaintext divisible by the block size-of Blowfish, which is 8 bytes, before proceeding to encrypt the padded plaintext using Blowfish with the supplied key. The resulting ciphertext is written to the output file with the IV as the first bytes of the output file. In decryption, the algorithm first obtains the IV from the ciphertext and then decrypts the remaining part to get back the padded plaintext and then strips the

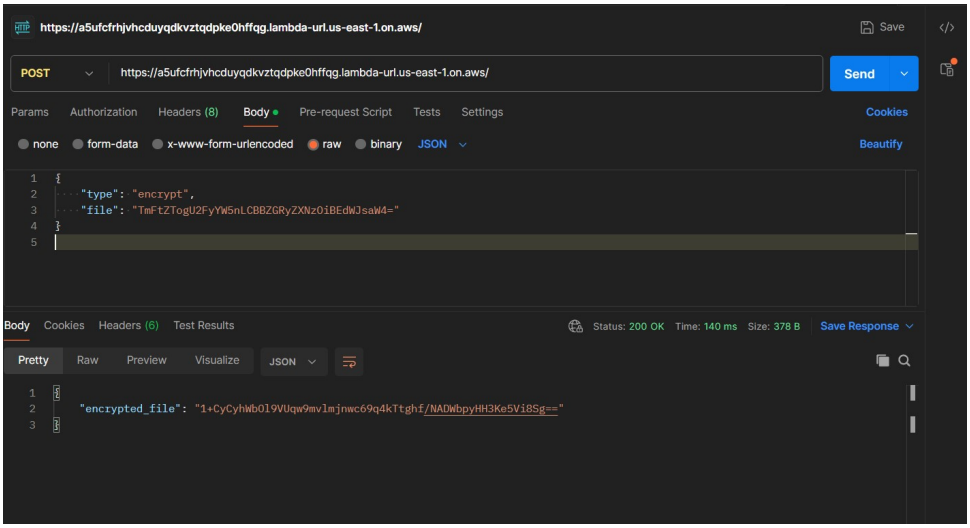


Figure 6: Data encrypted by Blowfish algorithm

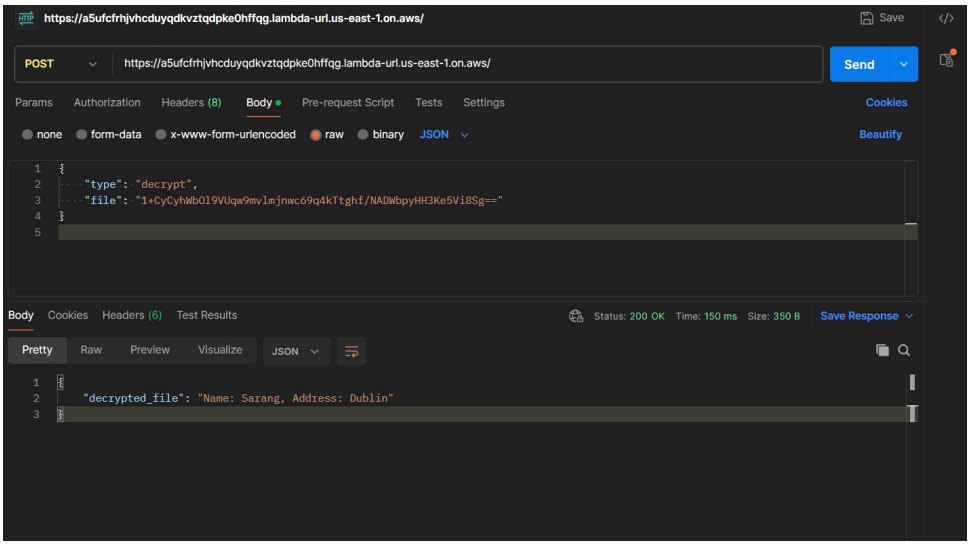


Figure 7: Data decrypted by LBSK algorithm

padding from the plaintext and writes the original plaintext to the output file. This makes sure the content of the files is protected using a proper encryption and decryption mechanism and the IV provides randomness to each encryption exercise.

The AWS Lambda function for LBSK performs encryption and decryption of file data that is passed as a Base64 string. When the trigger occurs, it assesses the incoming request whether to perform an “encrypt” or a “decrypt” operation. For encryption, it generates a key of suitable length then XOR’s the file data with the key and then encodes the result in Base 64; the function then returns the encrypted file. For decryption, it decodes the Base64 input and performs the procedure vice versa, that is, it decodes the encrypted data by performing XOR operation with the key and returns the decrypted content. The’get_key’ function also makes it possible to match the length of the encryption key to that of the entered data by utilizing a base key and its repetitions. The function checks the input data and generates proper responses in accordance with the request type, which

implies data safety and correct action with the files.

6 Evaluation

We carried out two tests by changing the file size and observing how every permutation of algorithm applied give us varied results. We used 50 iterations of encryption and decryption for file size of 100 kilobytes and then for a larger file size of 1 Megabyte. The

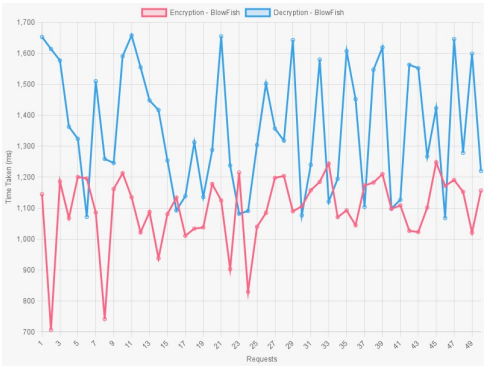
6.1 Experiment 1: Encrypting file of size 100 kilobyte

6.1.1 Encrypt and Decrypt using Blowfish algorithm

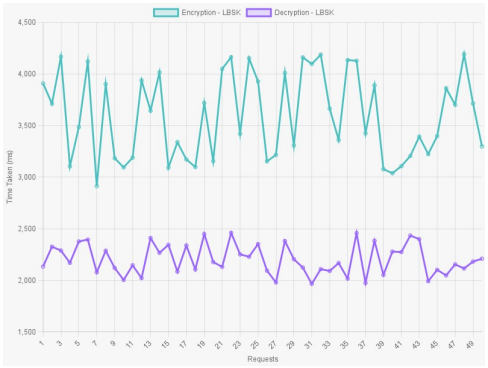
When a user uploads a file and marks it as non-sensitive, it is encrypted using the Blowfish algorithm. The average encryption time is 950 milliseconds and the decryption time is 1350 milliseconds, based on 50 iterations. Figure 8a shows the time per iteration in milliseconds.

6.1.2 Encrypt and Decrypt using LBSK algorithm

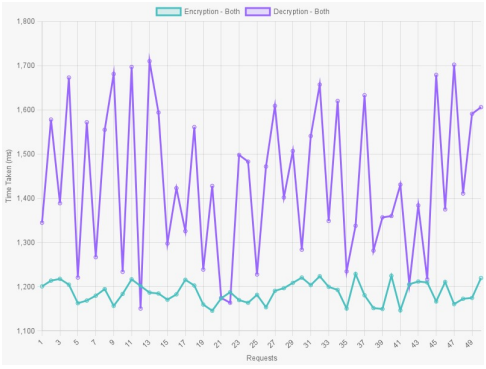
When a user uploads a file and marks it as sensitive, it is encrypted using the LBSK algorithm. The average encryption time is 3500 milliseconds and the decryption time is 2300 milliseconds based on 50 iterations. Figure 8b shows the time per iteration in milliseconds.



(a) Blowfish algorithm on 100kb file



(b) LBSK algorithm on 100kb file



(c) Blowfish and LBSK algorithm on 100kb file

Figure 8: Comparing LBSK and Blowfish Algorithm on 100kb file

6.1.3 Encrypt and Decrypt using both algorithms simulataneously

When half of the data is encrypted by Blowfish encryption and other half by LBSK encryption, it took an average of 1200 miliseconds to encrypt the entire data and average of 1450 miliseconds for decryption as shown in Figure 8c

6.2 Experiment 2: Encrypting file of size 1 Megabyte

6.2.1 Encrypt and Decrypt using Blowfish algorithm

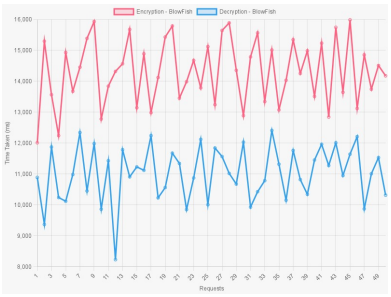
When a user uploads a file and marks it as non-sensitive, it is encrypted using the Blowfish algorithm. The average encryption time is 14000 milliseconds, and the decryption time is 11000 milliseconds based on 50 iterations. Figure 9a shows the time per iteration in milliseconds.

6.2.2 Encrypt and Decrypt using LBSK algorithm

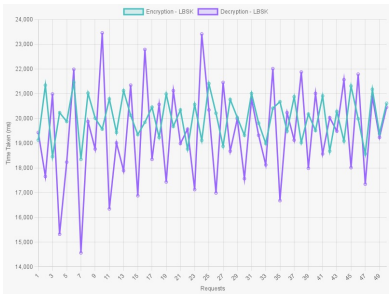
When a user uploads a file and marks its as sensitive, it is encrypted using the LBSK algorithm. The average encryption time is 20000 milliseconds, and the decryption time is roughly the same with high variance, based on 50 iterations. Figure 9b shows the time per iteration in milliseconds.

6.2.3 Encrypt and Decrypt using both algorithms simulataneously

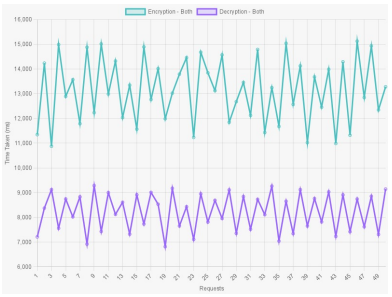
When half of the data is encrypted by Blowfish encryption and other half by LBSK encryption, it took an average of 13000 miliseconds to encrypt the entire data and average of 8000 miliseconds for decryption as shown in Figure 9c



(a) Blowfish algorithm on 1mb file



(b) LBSK algorithm on 1mb file



(c) Blowfish and LBSK algorithm on 1mb file

Figure 9: Comparing LBSK and Blowfish Algorithm on 1mb file

6.3 Experiment 3: Encrpyt Decrypt using variable sensitive data

On a 100kb file, when 20% of the data is encrypted by LBSK and 80% of by Blowfish encryption, it took an average of 2700 miliseconds to encrypt the entire data and average of 2100 miliseconds for decryption as shown in Figure 10

This is as close to real life test case as we could get because not all data is sensitive or non-sensitve all the time.

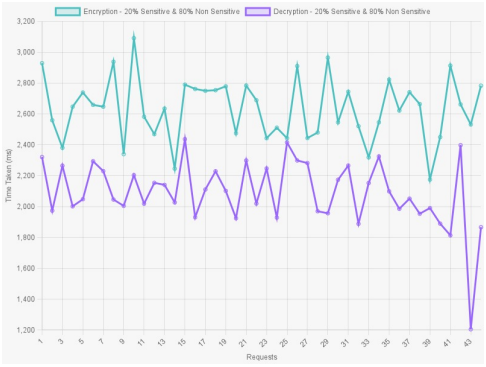


Figure 10: Comparing LBSK and Blowfish Algorithm with 20% sensitive and 80% non-sensitive data

Experiment	**File Size**	**Algorithm**	**Avg Encryption Time(ms)**	**Avg Decryption Time(ms)**
Experiment 1	100 KB	Blowfish	950	1350
Experiment 1	100 KB	LBSK	3500	2300
Experiment 1	100 KB	Blowfish(50kb) + LBSK(50kb)	1200	1450
Experiment 2	1 MB	Blowfish	14000	11000
Experiment 2	1 MB	LBSK	20000	20000 (high variance)
Experiment 2	1 MB	Blowfish(50kb) + LBSK(50kb)	13000	8000
Experiment 3	100 KB	20% LBSK + 80% Blowfish	2700	2100

Table 3: Summary of experiments: Encryption and Decryption Times for Blowfish and LBSK Algorithms

6.4 Potential vulnerabilities and management challenges

- There can be vulnerabilities in smart contract if it contains any bugs. These loopholes can be exploited which can lead to financial losses if they're not audited.
- Management of private keys can be a challenge. If these are lost or a theft takes place, it can lead to irretrievable loss of Intellectual Property.
- Limited scalability of the blockchain can lead to bottlenecks with high transaction volumes. This can lead to slower processing and higher transaction fees.
- Compliance with the regulatory bodies can be challenging when it comes to Decentralized storage.
- Integration of blockchain with the traditional systems can be challenging and it might require more time, effort and expertise.

It is worth the effort and cost to work around these challenges as the enhanced security and transparency could justify the implementation costs for organizations handling sensitive data. It also reduces the dependency on the cloud service providers for securely

storing the encrypted data hence reducing vendor lock-in and improves the control of user over their data.

6.5 Discussion

The above experiments as shown in Table 3 provide meaningful insights about the efficiency and performance of our dual encryption approach. For 100kb file size, the LBSK took around 3500 ms and 2300 ms to encrypt and decrypt the entire file. If we divided the task between LBSK and Blowfish algorithms assuming half of the data is sensitive, it took 1200 ms and 1450ms to encrpyt and decrypt the file. The results got even better when the file size was increased to 1 mb. The LBSK took roughly 20,000 ms to both encrypt and decrypt the file. If we divided the task between LBSK and Blowfish algorithms assuming half of the data is sensitive, it took 13000 ms and 8000 ms to encrpyt and decrypt the file. We can say that as file size increases, this approach only becomes more practical as it results in significant time savings.

To make a direct comparison, the time reduced from 3500ms to 2700 when using both algorithms simualtaneously on a 100kb file as compared to just using LBSK encryption algorithm(20% percent of data was sensitive hence applied LBSK and 80% data was non-sensitive hence applied Blowfish)

These findings have significant inference for data storage security. By integrating LBSK as well as Blowfish encryption in the blockchain framework, our proposed system has effectively achieved increased efficiency as compared to just having LBSK encryption as well as strong encryption for the sensitive data provided by the user. We have also reduced the dependency on the CSPs to encrypt and manage user data by using the decentralized Ethereum blockchain thereby enhancing user control and security. This confirms that data security can be improved beyond LBSK encryption algorithm when using dividing the data and encrypting it according to its sensitivity while also improving upon the efficiency of the entire process while also maintaining the system's security.

Practical impact of your system and industries that can adopt: This system could be valuable for industries requiring secure document management with auditability, such as:

- Healthcare: for managing patient records
- Legal: for contract management and case files
- Financial services: for transaction records and audits
- Government agencies: for secure record keeping

The effectiveness of this system will be high in such industries that need to secure sensitive data efficiently. The dual encryption of LBSK and Blowfish algorithms with our blockchain framework, the system provides security without hampering the performance. Large organizations exposed to heavier volumes of sensitive data or those need to decrease their dependency on CSPs while enhancing their data control could gain a lot from this approach.

7 Conclusion and Future Work

The overall objective of this study,if we refer back to our research question, was to improve the efficiency of the implementation of LBSK encryption algorithm on the blockchain. The proposed encryption security model by using the LBSK algorithm for the sensitive data with a combination of Blowfish algorithm for the non-sensitive data proved the study's effectiveness of a dual type of encryption model with both performance improvement as well maintaining security standards.

The combination of these encryption methods with a blockchain framework addresses such issues related to over dependence on the CSPs: vendor lock-in and loss of control over the data. The Ethereum blockchain was used to store the reference of encryption keys and the encoded file paths. This reduced the reliance on CSPs and enhanced user control over data.The outcomes of experiments proved that this approach enhances the process of encryption and minimizes the amount of time that takes to compute for non-sensitive data and the security of sensitive data remains intact. The classification of data depending on the sensitivity of the information using algorithms LBSK and Blowfish led to an increase efficiency in the encryption and decryption times and thus, the system's performance is higher, but the security did not suffer.

Thus, this strategy can be seen as an effective and realistic solution for the management of data encryption, which contributes to further cloud security within the framework of decentralized blockchain technologies. Possible future work may include automating the classification of user data and exploring the integration of other advanced encryption algorithms to further enhance the system's security and performance. Also, storing the cryptogrpahic hash of the location of the encryption keys in the smart contract and writing efficient smart contracts to keep the gas fee low and further analysis for an alternative blockchain solution can be looked into.

References

Aggarwal, S. and Kumar, N. (2021). Hyperledger, *Advances in Computers*, Vol. 121, Elsevier, pp. 323–343.

Al Nafea, R. and Almaiah, M. A. (2021). Cyber security threats in cloud: Literature review, *2021 international conference on information technology (ICIT)*, IEEE, pp. 779–786.

Banushri, A. and Karthika, R. (2023). Hyperledger blockchain and lightweight bcrypt symmetric key encryption to boost cloud computing security effectiveness, *2023 International Conference on Circuit Power and Computing Technologies (ICCPCT)*, IEEE, pp. 1525–1530.

Barrak, A., Fofe, G., Mackowiak, L., Kouam, E. and Jaafar, F. (2024). Securing aws lambda: Advanced strategies and best practices, *2024 IEEE 11th International Conference on Cyber Security and Cloud Computing (CSCloud)*, IEEE, pp. 113–119.

Beloglazov, A. and Buyya, R. (2015). Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds, *Concurrency and Computation: Practice and Experience* **27**(5): 1310–1333.

Buterin, V. (2016). Ethereum: platform review, *Opportunities and challenges for private and consortium blockchains* **45**: 1–45.

Cachin, C. et al. (2016). Architecture of the hyperledger blockchain fabric, *Workshop on distributed cryptocurrencies and consensus ledgers*, Vol. 310, Chicago, IL, pp. 1–4.

Chang, W. Y., Abu-Amara, H. and Sanford, J. F. (2010). *Transforming enterprise cloud services*, Springer Science & Business Media.

Choudhary, A., Verma, P. K. and Rai, P. (2022). Comparative study of various cloud service providers: A review, *2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)*, IEEE, pp. 1–8.

Devi, A., Sharma, A. and Rangra, A. (2015). A review on des, aes and blowfish for image encryption & decryption, *International Journal of Computer Science and Information Technologies* **6**(3): 3034–3036.

Doan, T. V., Psaras, Y., Ott, J. and Bajpai, V. (2022). Toward decentralized cloud storage with ipfs: opportunities, challenges, and future considerations, *IEEE Internet Computing* **26**(6): 7–15.

Gan, Q., Wang, X., Huang, D., Li, J., Zhou, D. and Wang, C. (2021). Towards multi-client forward private searchable symmetric encryption in cloud computing, *IEEE Transactions on Services Computing* **15**(6): 3566–3576.

Guerrero-Sanchez, A. E., Rivas-Araiza, E. A., Gonzalez-Cordoba, J. L., Toledano-Ayala, M. and Takacs, A. (2020). Blockchain mechanism and symmetric encryption in a wireless sensor network, *Sensors* **20**(10): 2798.

He, B., Feng, T., Fang, J., Liu, C. and Su, C. (2023). A secure and efficient charitable donation system based on ethereum blockchain and searchable encryption, *IEEE Transactions on Consumer Electronics* .

Hussain, S., Ashraf, S., Al Hamadi, H., Abideen, Z. U. et al. (2023). A critical analysis on cybercrimes, *2023 International Conference on Business Analytics for Technology and Security (ICBATS)*, IEEE, pp. 1–7.

Khan, P. W. and Byun, Y. (2020). A blockchain-based secure image encryption scheme for the industrial internet of things, *Entropy* **22**(2): 175.

Kumar, S., Sundaresan, P., Logith, R. and Mathivanan, N. (2023). A data security-based efficient compression and encryption for cloud computing, *2023 7th International Conference on Trends in Electronics and Informatics (ICOEI)*, IEEE, pp. 647–653.

Muthe, K. B., Vemuru, T. S. T., Sharma, K. and Mohammad, N. S. (2020). Decentranet-an ethereum, proxy re-encryption and ipfs based decentralized internet, *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, pp. 1–5.

Raj, M., Tahir, S., Khan, F., Tahir, H. and Zulkifl, Z. (2021). A novel fog-based framework for preventing cloud lock-in while enabling searchable encryption, *2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, IEEE, pp. 1–6.

Rankhambe, B. P. and Khanuja, H. K. (2019). A comparative analysis of blockchain platforms–bitcoin and ethereum, *2019 5th international conference on computing, communication, control and automation (ICCUBEA)*, IEEE, pp. 1–7.

Satyam, K., Sharma, A. and Devi, R. (2023). Implementing blockchain based security in ehr using ganache, *2023 6th International Conference on Contemporary Computing and Informatics (IC3I)*, Vol. 6, IEEE, pp. 276–284.

Shah, M., Shaikh, M., Mishra, V. and Tuscano, G. (2020). Decentralized cloud storage using blockchain, *2020 4th International conference on trends in electronics and informatics (ICOEI)(48184)*, IEEE, pp. 384–389.

Sriram, G. (2022). Resolving security and data concerns in cloud computing by utilizing a decentralized cloud computing option, *International Research Journal of Modernization in Engineering Technology and Science* **4**(1): 1269–1273.

Tang, X., Guo, C., Choo, K.-K. R., Liu, Y. and Li, L. (2021). A secure and trustworthy medical record sharing scheme based on searchable encryption and blockchain, *Computer Networks* **200**: 108540.

Zhang, Y., Xu, C., Ni, J., Li, H. and Shen, X. S. (2019). Blockchain-assisted public-key encryption with keyword search against keyword guessing attacks for cloud storage, *IEEE Transactions on Cloud Computing* **9**(4): 1335–1348.