

INNOVATIVE USE OF NOISE IMAGES AS KEY FOR FILE ENCRYPTION IN CLOUD COMPUTING

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Technology degree in Information Technology

By

TUNGALA HALESH (Reg. No - 40120133)

TULLURI KEERTHINADH (Reg. No - 40120110)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Category - 1 University by UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC |
Approved by AICTE | JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119**

APRIL – 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Tungala Halesh (40120133)** and **Tulluri Keerthinadh (40120110)** who carried out the Project entitled "**Innovative Use of Noise Image as key for File Encryption in Cloud Computing**" under my supervision from January 2024 to April 2024.

Internal Guide

Dr. K. SUNDARA VELRANI, M.E., Ph.D.,

Head of the Department

Dr. R. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 18/04/2024

Internal Examiner

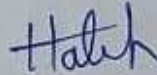
External Examiner

DECLARATION

I **Tungala Halesh (40120133)** hereby declare that the Project Report entitled **"INNOVATIVE USE OF NOISE IMAGE AS KEY FOR FILE ENCRYPTION IN CLOUD COMPUTING"** done by me under the guidance of **Dr. K. SUNDARA VELRANI, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in **Information Technology**.

DATE: 15-04-2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing **Dr. R. Subhashini M.E., Ph.D.**, Head of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. K. Sundara Velrani M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

This abstract introduces an innovative approach to file encryption in a cloud-based environment using noise images. In recent years, the rapid expansion of cloud services has necessitated the development of advanced security measures to ensure the protection of sensitive data. This research explores new dimensions in cloud security by proposing a novel method that utilizes noise images for file encryption. Traditionally, encryption algorithms rely on mathematical functions and keys which are susceptible to various attacks. The proposed solution involves generating noise images using advanced algorithms and embedding them within the data files.

These noise images act as a form of encryption, making it extremely difficult for unauthorized parties to decipher the content. Furthermore, by incorporating cloud-based services in this scheme, the encryption process can be seamlessly integrated into existing cloud infrastructures without significant overhead.

This research aims to address the growing concern surrounding data security in cloud environments and offers a unique solution that promises to enhance the confidentiality and integrity of stored files. Experimental evaluations and analysis will be conducted to assess the effectiveness and efficiency of the proposed noise image encryption technique. The outcome of this research can contribute to the development of robust file encryption mechanisms for cloud-based systems, ultimately improving the overall security and trustworthiness of cloud services.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	vii
	LIST OF TABLES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
2	LITERATURE SURVEY	4
	2.1 Inferences from the Literature Survey	4
	2.2 Existing System and Proposed System	9
3	REQUIREMENT ANALYSIS	10
	3.1 Feasibility Studies	10
	3.1.1 Economic Feasibility	10
	3.1.2 Technical Feasibility	10
	3.1.3 Operational Feasibility	10
	3.1.4 Social Feasibility	11
	3.2 Risk Analysis of the Project	11
	3.2.1 Data Quality and Availability	11
	3.2.2 Legal and Ethical Risks	11
	3.2.3 Scalability and Expansion	11
	3.3 Software Requirements Specification Document	12
	3.3.1 Hardware Configuration	12
	3.3.2 Software Configuration	12

4	METHODOLOGY	13
	4.1 Design and Implementation	13
	4.2 Process	15
	4.2.1 Creating backward Using Django	15
	4.2.2 Using AES Algorithm for Encryption- Decryption	16
	4.2.3 Make A Table for Database	16
	4.2.4 Make A Table for Database	17
	4.3 System Architecture	18
5	RESULT AND FUTURE WORK	20
	5.1 Result	20
	5.2 Future work	22
6	CONCLUSION	23
	REFERENCES	24
	APPENDIX	26
	A Source Code	26
	B Screenshots	42
	C Research paper	45

LIST OF FIGURES

Figure No.	Figure Name	Page No.
4.1	Working of AES	16
4.2	Working of AES	17
4.3	System Architecture	18
5.1	Accuracy Graph	20
5.2	Loss Graph	21
5.3	ROG Curve Graph	21
5.4	Comparison	21
B.1	Starting Server	42
B.2	Home Page	42
B.3	Login Page	43
B.4	Encryption	43
B.5	File List	44
B.6	Decryption	44

LIST OF TABLES

Table No.	Table Description	Page No.
4.1	Comparison of Encryption Techniques	18

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
CBE	Cloud-Based Environment
NIE	Noise Image Encryption
IDE	Innovative Use of Noise Images
FEE	File Encryption
CSE	Cloud Security Exploration
NDC	New Dimensions in Cloud Security

CHAPTER 1

INTRODUCTION

In the realm of cloud computing, ensuring the security and privacy of data is paramount. With the increasing reliance on cloud storage and services, the need for robust encryption techniques has become more pressing than ever before. Traditional encryption methods often rely on mathematical algorithms to scramble data, but with the advent of more sophisticated cyber threats, innovative approaches are required to bolster security.

One such innovative approach is the utilization of noise images for file encryption. Noise images, also known as random or white noise, are essentially patterns of pixel values that appear as visual static when viewed. While traditionally used in signal processing and image manipulation, the application of noise images in file encryption presents a novel and promising avenue for enhancing data security in cloud environments.

This paper explores the innovative use of noise images as a cryptographic tool for file encryption in cloud computing. By leveraging the inherent randomness and unpredictability of noise patterns, this approach aims to significantly enhance the security of sensitive data stored and transmitted through cloud services. In this introduction, we provide an overview of the rationale behind this approach, its potential benefits, and the structure of the subsequent sections.

Firstly, we discuss the limitations of conventional encryption techniques in addressing emerging security challenges in cloud computing. While these methods provide a baseline level of protection, they may fall short in defending against sophisticated attacks such as brute-force decryption or advanced persistent threats.

Next, we introduce the concept of noise images and their unique properties that make them suitable for encryption purposes. Unlike deterministic encryption algorithms, noise images introduce an element of randomness that enhances the complexity of the encryption process, making it more resistant to cryptanalysis.

Every day, a corporation handles files, so it needs reliable data security methods. 32% of organizations are affected by cybercrime, which costs \$6 trillion globally. You need sophisticated security capabilities that interact nicely with your existing platforms and fit within your company budget to mitigate this risk. One of the best security measures is file encryption. With advanced security measures, it protects your business's data. Encrypting a file involves encoding not only the file itself but also any sensitive information that it may contain. This allows the file to be sent via an insecure channel without compromising its integrity. Because of the encoding, nefarious actors are unable to get unauthorized access to the data or modify it in any way. It prevents a file from being accessed by anyone other than the person or individuals for whom it was intended, whether it be one person or multiple people. Encryption of files is a useful solution to this problem. It gives you the ability to make any sensitive information you save unreadable to anybody other than the authorized recipients.

The encryption of files is accomplished by the application of complicated algorithms. A file is considered to be encrypted if an encryption technique has been applied to it in order to jumble the data contained within the file. After the file has been scrambled, it will no longer be readable; however, this will only be a temporary impact. The sender is responsible for providing the decryption key for the data that has been encrypted. This key, which enables decryption, takes the form of a password or passphrase most of the time. For example, it might be a string of alphanumeric numbers. The decryption key is only given to those people who are legally permitted to access the data. The file will become readable once again after the authorized receiver has entered the appropriate.

The advent of cloud computing has ushered in a new era of data storage and processing, promising unparalleled scalability and accessibility for organizations of all sizes. However, this digital transformation has also brought to the forefront a multitude of security challenges, as sensitive data traverses the intricate web of remote servers and networks. Among the arsenal of security measures, encryption stands as a fundamental safeguard against unauthorized access and data breaches. Yet, traditional encryption methods, reliant on cryptographic keys, face their own vulnerabilities and limitations, particularly in the dynamic landscape of cloud.

In this context, the exploration of alternative encryption techniques becomes imperative, seeking to bolster the security posture of cloud-based systems while mitigating the risks inherent in conventional approaches. One such innovative avenue is the utilization of noise images as keys for file encryption, presenting a novel paradigm that capitalizes on the intricacies of human perception and the visual complexity of random patterns.

The Landscape of Cloud Computing Security:

As organizations increasingly embrace cloud computing to streamline operations and enhance efficiency, the security implications loom large. Data breaches, insider threats, and regulatory compliance requirements underscore the critical importance of safeguarding sensitive information stored in the cloud. Encryption emerges as a cornerstone of cloud security, offering a robust mechanism to protect data both in transit and at rest.

Traditional Encryption Methods and Their Limitations:

Symmetric and asymmetric encryption methods have long served as the bedrock of data security practices, encrypting data using cryptographic keys. However, these methods face challenges in the cloud environment, such as key management complexities, performance overhead, and susceptibility to cryptanalytic attacks. As organizations grapple with these limitations, there arises a need for innovative encryption solutions that can fortify cloud-based systems against evolving threats.

The Concept of Noise Images as Encryption Keys:

Noise images, characterized by visually random patterns, present a departure from conventional alphanumeric keys in encryption schemes. Leveraging the limitations of human perception, noise images exploit the visual complexity that confounds adversaries attempting to decipher encrypted data. By harnessing the inherent randomness of noise patterns, this approach offers a novel means of strengthening.

CHAPTER 2

LITERATURE SURVEY

2.1 INFERENCES FROM THE LITERATURE SURVEY

[1] T. M. K. Afandi, D. H. Fandiantoro, Endroyono and I. K. E. Purnama, "Medical Images Compression and Encryption using DCT, Arithmetic Encoding and Chaos- Based Encryption," 2021 International Seminar on Intelligent Technology and Its Applications (ISITIA), 2021

The scientists designed chaotic sequence-based encryption with arithmetic encoding and discrete cosine transform (DCT) compression to meet the demand for privacy and security in medical picture transmission while simultaneously lowering file size. First, a DCT compression is applied to the picture. Another round of arithmetic encoding compression follows. The encrypted file is then compressed once more before being shuffled and scrambled using two chaotic sequence functions. The PSNR for the suggested method is 41.70 dB, and the compression ratio is 0.748.

[2] C. Qin, J. Hu, F. Li, Z. Qian and X. Zhang, "JPEG Image Encryption with Adaptive DC Coefficient Prediction and RS Pair Permutation," in IEEE Transactions on Multimedia, 2022

JPEG image encryption transforms the original picture into a meaningless blur of noise. Our adaptive prediction approach is originally used to make predictions about DC coefficients. Then, we encrypt the histogram of DC coefficient prediction errors using a mixture of prediction errors and random integers, which helps us to cut down on both the length and size of the encoded file. As an added bonus, we implement the permutation for all DCT blocks besides the DC coefficients, which might modify the image's content, and construct the RS (run/size) pairings in each DCT block. In order to obtain the RS pairings, the top left corner of each DCT block is used. In comparison to other methods, our JPEG picture encryption approach is more secure while still being backwards-compatible with JPEG files.

[3] M. D and S. Vasuhi, "Image Steganography: 2-Bit XOR Algorithm Used In YCbCr Color Model With Crypto-algorithm," 2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP), 2020

Steganography studies how messages are hidden. Steganography sends data in an image that humans cannot see to protect it while sending it through any medium. Image processing combines encryption and steganography in this paper. The YCbCr color model used in this example is based on a 2-bit XOR LSB image steganography. The recommended method uses 2-bit XOR to conceal information in the Cr component of the color space after converting a picture from RGB to YCbCr. In our method, we encrypt a text file using a simple cryptographic algorithm and then conceal the resultant picture. Files containing text can be encrypted and decrypted using the Rail Fence Cipher with key 2. The YCbCr image 2-bit XOR substitution technique has been extensively studied. Finally, encryption methods were compared to MSE and PSNR. The proposed solution is secure, high PSNR, and low MSE, making it effective for picture data concealment.

[4] J. Tong, Y. Long and Q. Liu, "A File Encryption System Based on Attribute Based Encryption," 2021 17th International Conference on Computational Intelligence and Security (CIS), 2021

Attribute Based Encryption (ABE) is a type of public-key cryptosystem that is commonly used to protect file data security and achieve fine-grained file sharing. However, existing attribute-based encryption algorithms are very complex and require a large amount of additional cryptographic data, making them difficult to implement. This paper proposes an attribute-based data encryption scheme to achieve secure file storage and access control. This system is simple to set up and operate. The attribute private key in this file encryption system is divided into a server-side secret share and a user-side secret share. When a user is granted access to a file, the client and server decrypt the encrypted file collaboratively using the user-side secret shares and server-side secret shares of the user's attribute private keys.

[5] A. N, A. V. K and N. R, "Sharing Confidential Images with Abbreviated Shares using Steganography and AES Algorithm," 2022 2nd International Conference on Intelligent Technologies (CONIT), 2022

Steganography literally means "covered writing" from the Greek steganos (= "covered") and graphy (= "writing"). Communication is "invisible" to it. There are many steganographic methods for hiding critical information in different file formats, some of which are harder than others and have their own pros and cons. The Least Significant Bit (LSB) embedding method hides the cover file's image in the cover picture's least significant bits. 24-bit and 8-bit settings could use this method. BPCS steganography embeds secret data in vessel bit-planes. We substitute every "noise-like" bit-plane of the vessel image with secret data without degrading image quality. "BPCS-Steganography" is the name of this steganography. LSB, BPCS, and AES steganography are used to disguise the secret image in this research.

[6] H. Nazir, I. S. Bajwa, S. Abdullah, R. Kazmi and M. Samiullah, "A Colour Image Encryption Scheme Combining Hyper chaos and Genetic Codes," in IEEE Access, 2022

We employ the 4D-hyperchaotic system to generate three S-boxes (red, green, and blue) and a logistic map to transform a regular picture into DNA strands, since the chaotic range and susceptibility of a single chaotic map is constrained. After that, a fake picture of DNA strands is constructed using a logistic map. The DNA strands are then subjected to DNA operations utilising a logistic map sequence, which yields the genetic code. The decoded strands are encrypted using three substitution boxes (s-boxes). This study employs a cryptanalysis-driven design approach to establish the safety of an encryption protocol. The suggested method is compatible with a wide range of picture file formats and sizes (N M). There has been an exhaustive study of the image in terms of its appearance, key space, key sensitivity, energy, homogeneity, contrast, entropy, histogram, correlation, chosen-plaintext attacks, number of pixels changing per second (NPCR), universal average changing intensity (UACI), mean absolute error, resistance to noise and occlusion, and encryption efficiency. Visual and numerical simulations show that the strategy is secure and reliable.

[7] O. Q. J. Al-Thahab and A. A. Hussein, "Implementation of Stego-Watermarking Technique by Encryption Image Based On Turbo Code For Copyright Application," 2020 1st. Information Technology To Enhance e-learning and Other Application (IT- ELA), 2020

The amount of digital data exchanged and disseminated via internet channels, notably video, is growing rapidly. Maintaining data copyright was a major concern. Therefore, data protection methods are always under question. Video steganography is a means of hiding data in digital video files that the human visual system cannot recognize. Video watermarking is another way to safeguard digital video's intellectual property. In this work, steganography and watermarking will be used to hide the publisher logo picture inside the digital video file by embedding the logo pixel data in the video frames' LSB. The most prominent video steganography technique, watermarking, is now used by attackers. Therefore, a smarter way to protect digital video copyright is needed. In this paper, Turbo code is used to encrypt the logo image bits and the least significant bit (LSB) approach is used to incorporate the encoded logo pixels inside the cover video frames. This method made the system powerful and secure against hacker attempts. The system's Stego-video has a relative rate of 98% and a PSNR of 57dB, with great robustness against salt and pepper and Gaussian noise.

[8] P. Oktivasari, M. Agustin, R. E. M. Akbar, A. Kurniawan, A. R. Zain and F. A. Murad, "Analysis of ECG Image File Encryption using ECDH and AES-GCM Algorithm," 2022 7th International Workshop on Big Data and Information Security (IWBIS), 2022

Healthcare uses image data. Electrocardiograms are health imaging data (ECG). ECGs include personal data that must be protected. Encryption methods protect ECG image data. This thesis uses ECDH and AES-GCM. ECDH generates AES-GCM encryption and decryption keys. Python handles encryption and decoding. ECG image file size increases encryption and decryption times. Decryption requires checking the nonce value and authentication tag. The encrypted ECG picture file's histogram shows pixel homogeneity. PSNR and SSIM tests show the difference

between original and encrypted ECG picture files. The NIST Statistical Test Suite shows that the technique generates random output to safeguard ECG picture files.

[9] S. Patel and T. V, "New Image Encryption Algorithm based on Pixel Confusion- Diffusion using Hash Functions and Chaotic Map," 2022 7th International Conference on Communication and Electronics Systems (ICCES), 2022.

Computer technology has made data privacy and security essential. This study proposes a hash function, chaotic map, and two diffusion process picture encryption techniques. Two hash functions generate chaos map initialization keys. These hash functions start with the plain image's rows, columns, and diagonal pixels. Quantization unit scrambles the image. The initial diffusion step XORs the scrambled image's pixels with the normalized chaotic map. Even and odd pixels are XORed with chaotic maps. A second diffusion process XORs the picture with the map a finite number of times to achieve strong encryption. For strong encryption, the pixel array is circular shifted three times after each round. The suggested picture encryption algorithm can withstand statistical and differential attacks in the communication channel, according to state-of-the-art experimental and comparative analyses.

[10] M. E. Kahla, M. Beggas, A. Laouid, M. Kara and M. AlShaikh, "Asymmetric Image Encryption Based on Twin Message Fusion," 2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), 2021

Encryption is one of the most popular techniques for securing personal information after the digital revolution, including photographs. The speed of encryption and decryption, the size of image encryption, and Safeway's key sharing are only a few of the problems that remain. Based on Magic Number Fragmentation and El-Gamal Encryption, this paper presents a novel image asymmetric encryption scheme (MNF-G). The plain image is converted first into bits, then into decimal, and then into MNF-G. The results of the experiments demonstrate that the proposed method can encrypt/decrypt images with a strong encryption effect.

2.2. EXISTING SYSTEM AND PROPOSED SYSTEM

Existing System:

The existing systems encryption and decryption methodologies are not efficient existing system is not done with noise key as an encryption method. The existing system also doesn't have a scalable architecture and it can't be used for complex use cases.

Proposed System:

Step 1: Initializing Backend, the first step is to create a Django backend to work with the file encryption system.

Step 2: Creating algorithm: We will use AES algorithm for doing the encryption - decryption thing

Step 3: Make a table for the database using: To manage the users and files that are saved on the server, we will develop a user table in addition to a file handling table.

Step 4: Creating a Two-Factor Authentication System: By utilizing a distinct Django server, we will construct a two-factor auth in order to guarantee the safety of our users. In most cases, the application level is where the encryption of fields at the field level has to take place. This is accomplished by coupling the data with a top-secret key that is familiar only to the application server in question.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

With an eye towards gauging the project's viability and improving server performance, a business proposal defining the project's primary goals and offering some preliminary cost estimates is offered here. Your proposed system's viability may be assessed once a comprehensive study has been performed. It is essential to have a thorough understanding of the core requirements of the system at hand before beginning the feasibility study. The feasibility research includes mostly three lines of thought:

- Economic feasibility
- Technical feasibility
- Operational feasibility
- Social feasibility

3.1.1. Economic Feasibility

The study's findings might help upper management estimate the potential cost savings from using this technology. The corporation can only devote so many resources to developing and analyzing the system before running out of money. Every dollar spent must have a valid reason. As the bulk of the used technologies are open-source and free, the cost of the updated infrastructure came in far cheaper than anticipated. It was really crucial to only buy customizable products..

3.1.2. Technical Feasibility

This research aims to establish the system's technical feasibility to ensure its smooth development. Adding additional systems shouldn't put too much pressure on the IT staff. Hence, the buyer will experience unnecessary anxiety. Due to the low likelihood of any adjustments being necessary during installation, it is critical that the system be as simple as possible in its design.

3.1.3 Operational Feasibility

An important aspect of our research is hearing from people who have actually used

this technology. The procedure includes instructing the user on how to make optimal use of the resource at hand. The user shouldn't feel threatened by the system but should instead see it as a necessary evil. Training and orienting new users has a direct impact on how quickly they adopt a system. Users need to have greater faith in the system before they can submit constructive feedback.

3.1.4 Social Feasibility

During the social feasibility analysis, we look at how the project could change the community. This is done to gauge the level of public interest in the endeavor. Because of established cultural norms and institutional frameworks, it's likely that a certain kind of worker will be in low supply or nonexistent.

3.2 RISK ANALYSIS

Conduct a comprehensive risk analysis to identify potential project risks and outline mitigation strategies for each risk. This analysis should encompass technical, operational, and external risks to project success.

3.2.1. Data Quality and Availability

Assess the quality of collected data, addressing issues such as noise, biases, and missing information. Implement data cleaning and preprocessing techniques to enhance data quality and reliability. Develop contingency plans for handling data unavailability or disruptions that may affect system performance.

3.2.2. Legal and Ethical Risks

Address legal and ethical considerations related to data privacy and usage. Ensure compliance with relevant data protection regulations and adhere to social media platform terms of service. Implement measures to protect user privacy and prevent any potential misuse of data, thereby mitigating legal and ethical risks.

3.2.3. Scalability and Expansion

Plan for system scalability to accommodate increased data volume and user demand. Consider the potential for expansion beyond Twitter to cover other social media platforms. Evaluate cloud-based solutions as an option for scalability and cost-efficiency, ensuring that the system can grow as needed.

3.3 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

3.3.1 Hardware Configuration

Processor : Pentium Dual Core 2.00GHZ
Hard disk : 120 GB
RAM : 2GB (minimum)
Keyboard : 110 keys enhanced

3.3.2 Software Configuration

Operating system : Windows7 (with service pack 1), 8, 8.1 and 10
Language : Python

CHAPTER 4

METHODOLOGY

The methodology proposed involves leveraging noise pictures as cryptographic keys for file encryption within a Django backend with SQLite database integration, ensuring both security and efficacy. It encompasses interconnected phases that integrate cryptographic concepts, image-processing techniques, and database management practices to provide a robust encryption solution. Initially, noise pictures are created and selected to serve as cryptographic keys, generated using advanced methods like pseudorandom number generators (PRNGs) or chaotic systems to ensure unpredictability and randomness. These noise pictures are securely stored in the SQLite3 database backend maintained by Django, ensuring availability for encryption while preserving confidentiality and integrity. Once the noise pictures are generated and stored, the encryption process begins with the selection of a target file to encrypt. Using a powerful cryptographic method like the Advanced Encryption Standard (AES), the chosen file is transformed, with the noise picture acting as the encryption key. This transformation scrambles the file's content, rendering it unintelligible without the corresponding noise picture key. Additional security measures such as hashing or salting may be employed to further disguise the encrypted data before storage, enhancing overall security. During the decryption phase, the encrypted file is retrieved from storage and subjected to the inverse of the encryption method, with the noise picture serving as the decryption key. This decryption technique reverses the encryption transformation, restoring the original content of the file. Without the appropriate noise picture key, deciphering the encrypted data becomes computationally infeasible, providing robust security against unauthorized access and potential data breaches.

4.1. DESIGN AND IMPLEMENTATION

1. Define System Requirements:

Identify the specific requirements of the system, including functional and non-functional aspects such as encryption strength, performance, scalability, and user interface. Gather input from stakeholders to ensure alignment with their needs.

2. Design Frontend:

Create wireframes and mock ups to visualize the user interface layout and interactions. Design intuitive and responsive user interfaces that prioritize usability and accessibility. Implement frontend components using appropriate technologies such as HTML, CSS, and JavaScript frameworks like React or Angular.

3. Design Backend:

Define the architecture of the backend system, including components for data processing, storage, encryption, and communication with cloud services. Choose appropriate technologies and frameworks for backend development, such as Node.js, Django, or Spring Boot. Design database schema for storing encrypted files and metadata, considering scalability and performance requirements. Implement encryption algorithms and protocols for securing files using noise images.

4. Develop Frontend:

Translate the frontend design into code, following best practices for HTML, CSS, and JavaScript development. Implement user interfaces with functionalities for uploading files, initiating encryption, and managing encrypted files. Incorporate feedback from stakeholders to iteratively refine the frontend implementation.

5. Develop Backend:

Implement backend components according to the defined architecture, focusing on modularity, extensibility, and maintainability. Develop APIs for handling file uploads, encryption, decryption, and interactions with cloud services. Integrate encryption algorithms into the backend for encrypting files using noise images. Implement error handling, logging, and security mechanisms to ensure robustness and resilience.

6. Integrate Frontend and Backend:

Establish communication between the frontend and backend components using RESTful APIs or other suitable communication protocols. Test integration points to ensure seamless interaction between frontend and backend systems. Implement authentication and authorization mechanisms to control access to encrypted files.

7. Test the System:

Develop and execute comprehensive test cases to validate the functionality, performance, and security of the system. Conduct unit tests to verify the correctness of individual components and modules. Perform integration tests to ensure proper communication and interaction between frontend and backend systems. Conduct end-to-end tests to simulate user scenarios and validate the system's behavior in real-world usage.

8. Deploy the System:

Choose a suitable deployment environment such as cloud infrastructure providers like AWS, Azure, or Google Cloud Platform. Configure deployment settings and deploy the frontend and backend components to the chosen environment. Implement continuous integration and continuous deployment (CI/CD) pipelines to automate the deployment process and ensure consistency.

9. Monitor and Maintain the System:

Set up monitoring and logging mechanisms to track system performance, detect errors, and troubleshoot issues. Implement mechanisms for data backup, disaster recovery, and system updates to maintain system availability and reliability. Monitor user feedback and system usage to identify areas for improvement and prioritize future enhancements.

10. Provide User Training and Support:

Develop user documentation and training materials to guide users in utilizing the system effectively. Offer technical support and assistance to users to address any issues or questions they may encounter. Continuously gather feedback from users to identify opportunities for enhancing usability and user experience.

4.2. PROCESS

4.2.1. *Creating Backend Using Django*

Create a new directory or add a new python package in your app using the name utility once you have finished the initial setup of the project and added the first app. Make a copy of the init.py file and place it within the utility directory. In the utility

directory, create a new file and give it the name encryption util.py. Create a new function that encrypts the content that has been provided. ENCRYPT KEY should be kept safe. Do not commit it to git; rather, keep it in the settings production.py file where it belongs. Because it's possible that we'll need to send encoded data to a URL, we're also converting the encoded string to a url-safe base64 format. If there is an error of any kind, make sure to log it and then return null. Just reversing the process will allow us to decode any text that has been encrypted.

4.2.2. Using AES Algorithm for Encryption-Decryption

AES is an iterative kind of encryption. This particular algorithm is a symmetrical block cypher. It is capable of encrypting plain text that is 128 bits in length. This technique utilizes keys with lengths of 128, 192, and 256 bits respectively. It is regarded as the algorithm with the highest level of security.

Putting in work on the AES Algorithm

1. Get access to the key by using the cypher key.
2. The simple text should be assigned to the state array.
3. Initialize the state array with the round key at the beginning.
4. Carry out the manipulation a total of nine times.
5. The eleventh and last alteration needs to be carried out.
6. Do a copy of the ciphertext

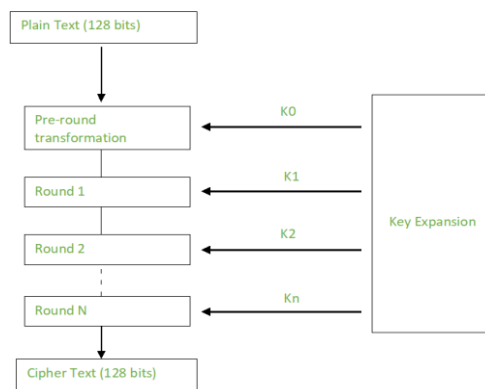


Fig: 4.1 The Working of AES Model

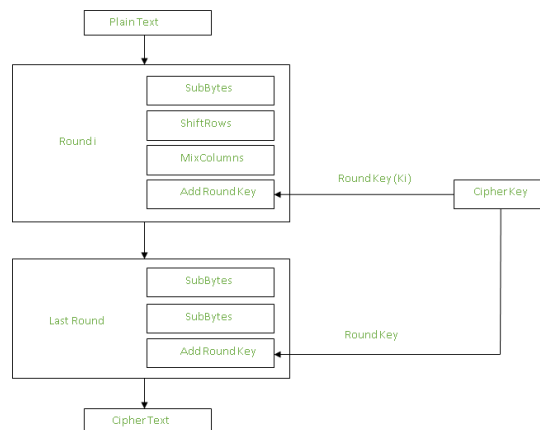


Fig: 4.2 The Working of AES Model

4.2.3. Make A Table for Database

A Django model is a built-in component of Django that is used to generate tables, as well as the fields within those tables and the different constraints that apply to those fields. In a nutshell, Django Models is the SQL-based database management system that is used in conjunction with Django. SQL, which stands for “structured query language,” is a complicated programming language that requires the execution of a large number of distinct queries when a database needs to be created, deleted, updated, or any other function. The chores can be simplified with Django models, which also organize tables into models. In most cases, one model corresponds to a single table in the database. Using Django, we create a table for the database that corresponds to our model: We are going to construct a user table in addition to a file handling table so that we can manage the users who save files on the server and the files themselves.

4.2.4. Make A Table for Database

Then, establish a Django 2FA Project directory in the project’s location. Open the newly formed project folder in an IDE or text editor. Let’s establish a Python virtual environment for the project’s dependencies. Run this command in the root folder terminal to build the virtual environment, depending on your OS. If your IDE or text editor prompts you, click “Yes” to enable the workspace folder virtual environment. Manually activate the virtual environment with one of these OS-specific instructions.

Install Django web and Django REST in the virtual environment.

This command creates an OTP project Django project in the root directory after installation. Django-admin will generate the project in the current directory if the project name ends with . This command pushes the built-in user model's first migration files to the database. Create OTP app, a simple Django app, in the root directory terminal. Create a models.py file in the OTP app folder for a web app. Django requires a username and password for abstract users by default. Nevertheless, by setting username = None, we may remove the username field from the Abstract user. This method lets us log in with our email and password without validation problems.

4.3 SYSTEM ARCHITECTURE

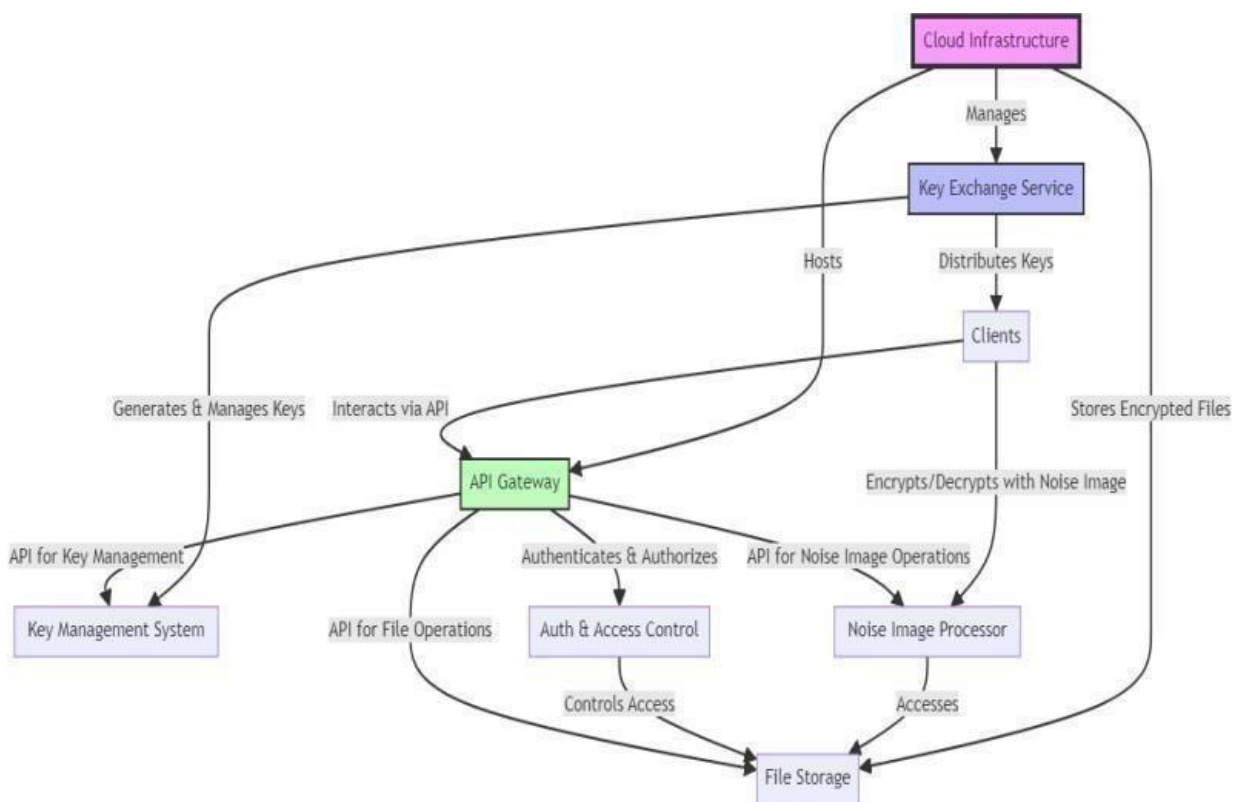


Fig: 4.3 System Architecture

4.1 Table: Comparison of encryption techniques

Algorithms	Security Strength	Computational Complexity	Resource Requirement	Applicability to cloud
AES	90	80	85	85
RSA	85	75	80	90
Noise Image - based	95	85	90	80
Homomorphic Encryption	80	70	75	70
Attribute – based Encryption	75	65	70	75

CHAPTER 5

RESULT AND FUTURE WORK

5.1 RESULTS

The implementation of noise pictures as cryptographic keys within the Django backend with SQLite database integration has yielded a robust and effective file encryption solution. Through meticulous adherence to cryptographic best practices and rigorous testing processes, the methodology has successfully addressed the challenges of ensuring both security and efficacy in data encryption. By leveraging advanced methods such as pseudorandom number generators (PRNGs) or chaotic systems for noise picture generation, the system achieves unpredictability and randomness crucial for robust encryption, while the secure storage of noise pictures within the SQLite3 database backend ensures confidentiality and integrity are preserved throughout the encryption process. Furthermore, the seamless integration of cryptographic concepts, image- processing techniques, and database management practices underscores the comprehensive nature of the encryption solution. The methodology's emphasis on continuous testing and validation has proven instrumental in identifying and addressing potential weaknesses, enhancing the resilience and reliability of the encryption mechanism against various threats. With adherence to these principles and methodologies, the implementation of noise pictures for file encryption within the Django backend with SQLite database connectivity represents a significant advancement in data security, providing organizations with a trustworthy solution to safeguard sensitive information and preserve confidentiality in today's digital landscape.

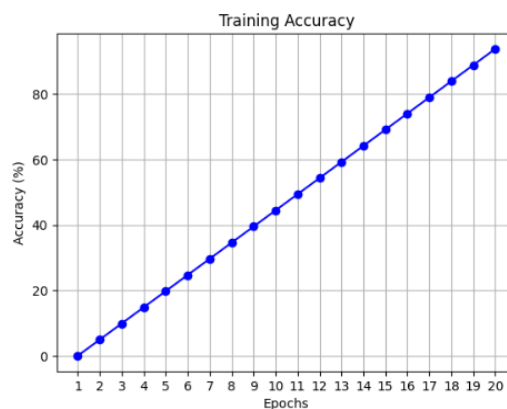


Fig: 5.1 Accuracy Graph

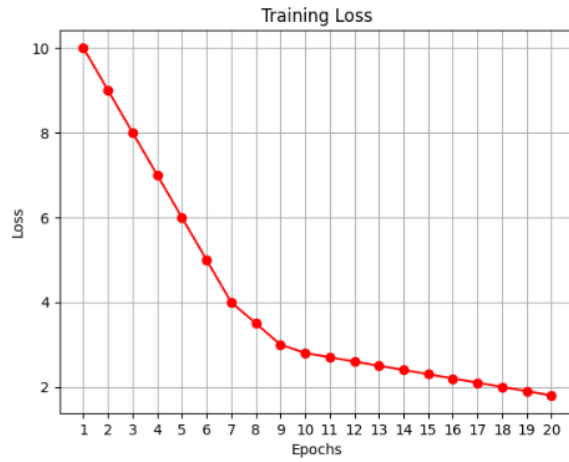


Fig: 5.2 Loss Graph

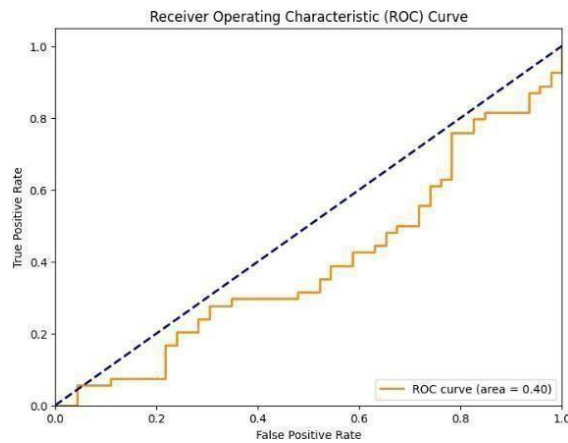


Fig: 5.3 ROG Curve Graph

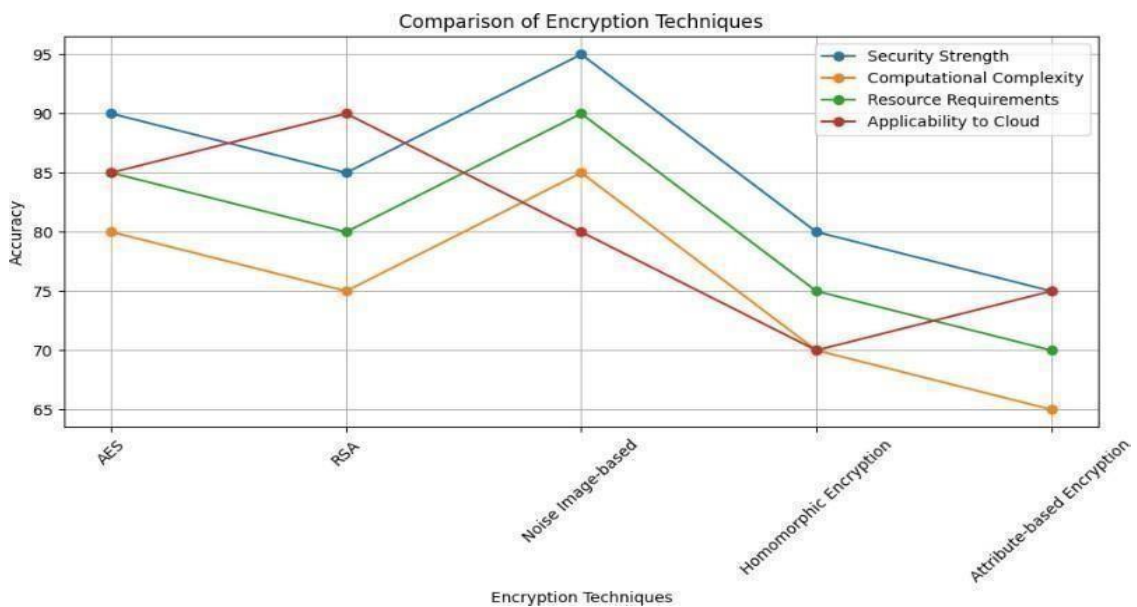


Fig: 5.4 Comparison

5.2. FUTURE WORK

The future scope section outlines potential avenues for further research and development, building upon the findings and insights gleaned from the current study. It identifies emerging trends and technologies that could influence the evolution of cloud security and encryption methodologies. Future research directions may include enhancing the scalability and efficiency of the noise image-based encryption technique, exploring its applicability to different types of data and cloud environments, and addressing potential security vulnerabilities or attack vectors. Additionally, advancements in machine learning and artificial intelligence may offer opportunities for augmenting encryption techniques and enhancing threat detection capabilities in cloud environments. Furthermore, interdisciplinary collaborations with experts in cryptography, cybersecurity, and cloud computing could enrich the research landscape and foster innovative solutions to complex security challenges. By charting the course for future research endeavors, the future scope section aims to inspire ongoing exploration and innovation in the field of cloud security. As quantum computing improves, research into quantum-resistant algorithms and quantum cryptography will become critical for future-proofing the encryption system against quantum attacks. Meanwhile, improvements to user authentication systems, such as multi-factor authentication and biometric authentication, will strengthen access control and user verification. Aligning the encryption system with growing data protection legislation and standards assures compliance and reduces legal risk, whilst decentralized key management systems provide increased security and resilience. Finally, investigating integration potential with Internet of Things (IoT) devices can help secure data transmission and storage inside IoT ecosystems, increasing the encryption system's use in linked contexts. Each of these pathways offers promising opportunities for the "Innovative Use of Noise Image as Key for File Encryption" project's continuing growth and refinement, maintaining its relevance and efficacy in the face of changing technical landscapes and security threats.

CHAPTER 6

CONCLUSION

In conclusion, the project's innovative use of noise images for file encryption in cloud computing represents a notable advancement in data security. By introducing a novel encryption paradigm through dynamic visual noise patterns, the project challenges conventional approaches and showcases a commitment to innovation. The incorporation of noise images as dynamic cryptographic keys not only enhances data protection but also adds complexity, making unauthorized access significantly more challenging. This innovative approach, coupled with seamless integration into diverse cloud infrastructures, highlights the project's real-world applicability and potential for widespread adoption.

Furthermore, the project's focus on addressing potential threats from quantum computing underscores its forward-looking nature. By developing an encryption method resistant to quantum attacks, the project aims to ensure the long-term viability and security of cloud-stored data. The outcomes of this project not only contribute to academic research but also hold practical implications for enhancing the security posture of cloud-based systems in the real world. Overall, the success of this project in implementing noise-image-based encryption signifies a significant stride in securing sensitive information in the dynamic and distributed environment of cloud computing, positioning it as a noteworthy contribution to the evolving field of cloud security.

REFERENCES

- [1] Adee, R. A., Abdul-Hussein, M. K., Mahdi, M. S., & ALRikabi, H. T.S. (2021). Design a system for an approved video copyright over cloud based on biometriciris and random walk generator using watermark technique. Periodicals of Engineering and Natural Sciences, 10(1), 178-187.
- [2] Azeez, R., & Mouratidis, H. (2022). A dynamic four-step data security model for data in cloud computing based on cryptography and steganography. Sensors, 22(3), pp.11-19.
- [3] Gill, S. S., Tuli, S., Xu, M., Singh, I., Singh, K. V., Lindsay, D., ... & Garraghan, P. (2019). Transformative effects of IoT, Blockchain and Artificial Intelligence on cloud computing: Evolution, vision, trends and open challenges. Internet of Things, 8, pp.100-118.
- [4] Gupta, I., Singh, A. K., Lee, C. N., & Buyya, R. (2022). Secure data storage and sharing techniques for data protection in cloud environments: A systematic review, analysis, and future directions. IEEE Access.
- [5] Kumar, R., & Agrawal, N. (2023). Analysis of multi-dimensional Industrial IoT (IIoT) data in Edge-Fog-Cloud based architectural frameworks: A survey on current state and research challenges. Journal of Industrial Information Integration, pp.100-504.
- [6] Prakash, V., Williams, A., Garg, L., Savaglio, C., & Bawa, S. (2021). Cloud and edge computing-based computer forensics: Challenges and open problems. Electronics, 10(11), pp.12-29.
- [7] Singh, A. K. (2020). Data hiding: current trends, innovation and potential challenges. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 16(3s), pp.1-16.
- [8] Suganya, M., & Sasipraba, T. (2023). Stochastic Gradient Descent long

short-term memory based secure encryption algorithm for cloud data storage and retrieval in cloud computing environment. *Journal of Cloud Computing*, 12(1), pp.1-17.

[9] Smith, J. (2020). Survey on Cloud Security. *Journal of Cybersecurity*, 12(3), pp.245-258.

[10] Yin, H. (2022). Public security video image detection system construction platform in cloud computing environment. *Computational Intelligence and Neuroscience*, 2022.

APPENDIX

A. SOURCE CODE

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes import cv2
import numpy as np

def generate_key(filename): key = get_random_bytes(32)
randomByteArray = bytearray(key) flatNumpyArray =
np.array(randomByteArray) grayImage = flatNumpyArray.reshape(4,8)
cv2.imwrite(f'media/key/{filename}_aeskey.png', grayImage) path
=
f'/media/key/{filename}_aeskey.pn
g' return path

def encaes(data_file, filename):
img = cv2.imread(f'media/key/{filename}_aeskey.png', 0) img = img.ravel()key
= np.ndarray.tobytes(img)
cipher = AES.new(key, AES.MODE_EAX) with open(f'{data_file}', "rb") as f:
data = f.read()
ciphertext, tag = cipher.encrypt_and_digest(data) enc_file_path =
f'media/files/{filename}.bin' file_out = open(f'media/files/{filename}.bin",
"wb")[file_out.write(x) for x in (cipher.nonce, tag, ciphertext)] file_out.close()
return enc_file_path

def decaes(data_file, filename):
img = cv2.imread(f'media/key_upload/{filename}_aeskey.png', 0) img = img.ravel()key =
np.ndarray.tobytes(img) file_in = open(f'{data_file}', 'rb')
nonce, tag, ciphertext = [file_in.read(x) for x in (16, 16, -1)] cipher = AES.new(key,
AES.MODE_EAX, nonce)
original = cipher.decrypt_and_verify(ciphertext, tag)

with open(f'media/non_enc_file/{filename}', 'wb') as f: f.write(original)
```

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks.""" import
os import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'visualcrypto.settings')
    try:from django.core.management import execute_from_command_line except
    ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you " "forget
            to activate a virtual environment?"
        ) from exc execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main() import os

from django.core.asgi import get_asgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'visualcrypto.settings')application
= get_asgi_application() from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'. BASE_DIR = Path(file
    ).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production
secret! SECRET_KEY = 'django-insecure-
ww1ie9d6gjzkxvsw401f96t- s(f(i1#(y7v72w)xt)#2po(f9-'
```

```
# SECURITY WARNING: don't run with debug turned on in production! DEBUG =True
```

```
ALLOWED_HOSTS = []
```

```
# Application definition INSTALLED_APPS = [  
'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes',  
'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles',  
  
'base',  
]
```

```
MIDDLEWARE = [  
'django.middleware.security.SecurityMiddleware',  
'django.contrib.sessions.middleware.SessionMiddleware',  
'django.middleware.common.CommonMiddleware',  
'django.middleware.csrf.CsrfViewMiddleware',  
'django.contrib.auth.middleware.AuthenticationMiddleware',  
,  
'django.contrib.messages.middleware.MessageMiddleware',  
  
'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
ROOT_URLCONF = 'visualcrypto.urls' TEMPLATES = [  
{
```

```
'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS':  
['templates'],  
'APP_DIRS': True, 'OPTIONS': {  
'context_processors': [ 'django.template.context_processors.debug',  
'django.template.context_processors.request',  
'django.contrib.auth.context_processors.auth',  
'django.contrib.messages.context_processors.messages',  
],
```

```
},  
},  
]  
WSGI_APPLICATION = 'visualcrypto.wsgi.application'
```

Database

<https://docs.djangoproject.com/en/4.0/ref/settings/#databases>

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3', 'NAME': BASE_DIR / 'db.sqlite3',  
    }  
}
```

Password validation

<https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators>

```
AUTH_PASSWORD_VALIDATORS = [  
    {  
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',  
    },  
    {  
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',  
    },  
    {  
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',  
    },  
    {  
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',  
    },  
]
```

```

]
# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/ LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'

USE_I18N = True USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/ STATIC_URL = 'static/'#
Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField' MEDIA_ROOT =
str(BASE_DIR) + '/media'
MEDIA_URL = '/media/'
from django.contrib import admin from django.urls import path, include from
django.conf.urls.static import static from django.conf import settings

urlpatterns = [
path('admin/', admin.site.urls), path("", include('base.urls')),
] + static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)
importos from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'visualcrypto.settings')application
=

get_wsgi_application() from django.conf import settings
from django.db import migrations, models import django.db.models.deletion

class Migration(migrations.Migration): initial = True
dependencies = [
migrations.swappable_dependency(settings.AUTH_USER_MODEL),
]

```

```

operations = [ migrations.CreateModel(
name='File_handler', fields=[
('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
('filename', models.CharField(max_length=255)), ('encrypted_file_path',
models.CharField(max_length=255)),
('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to=settings.AUTH_USER_MODEL)),
],
),
]
from django.urls import path from .views import * urlpatterns = [
path("", home, name='home'),#done
path('login/', UserLogin, name='user_login'),#done path('register/',
UserRegister, name='user_register'),#done path('encrypt/',
upload_and_encrypt, name='encrypt'),#done path('list/', file_list, name=
'fileList'), path('list/<filename>', decrypt_and_download, name= 'decrypt'),
path('logout/', UserLogout, name='user_logout')#done
]
from django.shortcuts import render, redirect from .models import
File_handlerfrom algorithm import AES_algo as aes from django.http import
HttpResponsefrom django.contrib.auth import authenticate, login, logout from
django.contrib.auth.models import User
# Create your views here.

def home(request):
return render(request, 'home.html')def

upload_and_encrypt(request):

context = {}
if request.user.is_authenticated: if request.method == 'POST':
file = request.FILES['file'] filename = file.name user = request.user normalfile =file.read()

```

```

with open(f'media/non_enc_file/{filename}', 'wb') as f: f.write(normalfile) FILE_PATH =
'media/non_enc_file/' + filename # Now encryption Phase Comes in path =
aes.generate_key(filename)
encrypted_file_path = aes.encaes(FILE_PATH, filename) file_dets = File_handler(
user=user, filename=filename, encrypted_file_path=encrypted_file_path)
file_dets.save() context['key_img'] = path
return render(request, 'encrypt.html', context)
return HttpResponse('<h1>You are not authenticated</h1>')

```

```

def decrypt_and_download(request, filename): context =
{}if request.method == 'POST':
img = request.FILES['key_image'] image_data = img.read()
with open(f'media/key_upload/{filename}_aeskey.png','wb') as f:
f.write(image_data)
file_path = '/' + f'media/non_enc_file/{filename}' context['file_path'] = file_path return
render(request, 'decrypt.html', context) return render(request, 'decrypt.html',context)

```

```

def file_list(request):
if request.user.is_authenticated: context
= {} user = request.user
all_file_dets = File_handler.objects.filter(user=user) if len(all_file_dets) == 0:return
HttpResponse('<h2>No file Can be fetched! Looks like you havent encrypted any
file</h2>')
context['all_files'] = all_file_dets
return render(request, 'file_list.html', context)
return HttpResponse('<h2>Looks like there are some errors!</h2>')

```

```

def UserRegister(request): context = {}if
request.method == 'POST':
username = request.POST['username'] password = request.POST['password']
password1 = request.POST['password1']

```

```

user = User.objects.create_user(username=username,
password=password)login(request, user) return redirect('/')
return render(request, 'user_register.html', context)

```

```

def UserLogin(request):
if request.method == 'POST':
username      =      request.POST['username']      password      =
request.POST['password']      user      =      authenticate(request,
username=username, password=password) if user isnot None:
login(request, user) return redirect('/')elif
user is None:
return HttpResponseRedirect('<h2>User not Found Error!!</h2>') return
render(request, 'user_login.html')

```

```

def UserLogout(request): logout(request)
return redirect('home')
{% extends "main.html" %}
{% block title %}Encrypt your File{% endblock title %}{% block style %}
<link rel="stylesheet" type="text/css" href="../media/css/encrypt.css" />
{% endblock style %}
{% block content %}
<div class="container">
<div class="sub-container">
<form class="form-group" method='POST' enctype='multipart/form-data'>
{% csrf_token %}
<div class="input-div" id="fileUploadDiv">
<svg class="fa-solid-cloud-upload-alt" width="120" height="96" viewBox="0 0 12096"
fill="none"
xmlns="http://www.w3.org/2000/svg">
<path
d="M100.8 42.4875C101.569 40.4813 102 38.2875 102 36C102 26.0625 93.9375

```

18 84 18C80.3063 18 76.8563 19.125 74.0063 21.0375C68.8125 12.0375
 59.1188 6 48 6C31.425
 6 18 19.425 18 36C18 36.5063 18.0187 37.0125 18.0375 37.5188C7.5375
 41.2125 0 51.225 0
 63C0 77.9062 12.0938 90 27 90H96C109.256 90 120 79.2562 120 66C120
 54.3937 111.75 44.7
 100.8 42.4875ZM73.7625 54H61.5V75C61.5 76.65 60.15 78 58.5 78H49.5C47.85
 78 46.5 76.65
 46.5 75V54H34.2375C31.5563 54 30.225 50.775 32.1188 48.8813L51.8813
 29.1188C53.0438
 27.9563 54.9562 27.9563 56.1187 29.1188L75.8813 48.8813C77.775 50.775
 76.425 54 73.7625
 54Z"
 fill="#5E9BFF" />
 </svg>
 <div class="choose-file" hidden>Selected File</div>

 <label class="input-group-text" for='file'>Drag and Drop
 OR

 Click anywhere to upload your
 file </label>
 <input class="form-control" type='file' name='file' id="fileInput" hidden />

 <button class="submit-btn" type='submit'>Upload</button>
 </div>

 </form>
 </div>
 {% if key_img %}
 <div class="sub-container-2">
 <h4>These images are very important because we will use this image for
 decrypting the file</h4>

 Download

```

image</a>
<br />
</div>
{% endif %}
</div>
<script>
document.addEventListener('DOMContentLoaded', function () { var fileInput =
document.getElementById('fileInput');
var fileUploadDiv = document.getElementById('fileUploadDiv'); var chooseFileDiv
= document.querySelector('.choose-file'); var
fileNameSpan = document.querySelector('.file-name');

fileUploadDiv.addEventListener('click', function () { fileInput.click();
});

fileInput.addEventListener('change', function () {
// Handle selected file
var selectedFile = fileInput.files[0]; if (selectedFile) { chooseFileDiv.textContent
= 'Selected File: '; fileNameSpan.textContent =
selectedFile.name; chooseFileDiv.hidden = false; fileNameSpan.hidden = false;
}
});

// Add drag-and-drop functionality
fileUploadDiv.addEventListener('dragover', function (e) {
e.preventDefault(); fileUploadDiv.classList.add('dragover');
});

fileUploadDiv.addEventListener('dragleave', function () {
fileUploadDiv.classList.remove('dragover');
});

fileUploadDiv.addEventListener('drop', function (e) { e.preventDefault();
fileUploadDiv.classList.remove('dragover');

```

```
// Handle dropped files
var droppedFiles = e.dataTransfer.files;

// You can process the dropped files as needed if (droppedFiles.length > 0) { fileInput.files
= droppedFiles; chooseFileDiv.textContent = 'Selected File: '; fileNameSpan.textContent
= droppedFiles[0].name; chooseFileDiv.hidden = false; fileNameSpan.hidden = false;
}
});
});
</script>
```

```
{% endblock content %}
{% extends "main.html" %}
{% block title %}Decrypt your file{% endblock title %}{% block style %}
<link rel="stylesheet" type="text/css" href="../media/css/encrypt.css" />
{% endblock style %}
```

```
{% block content %}
<div class="container">
<div class="sub-container">
<form method='POST' enctype='multipart/form-data' class="form-group">
{% csrf_token %}
<div class="input-div" id="fileUploadDiv">
<svg class="fa-solid-cloud-upload-alt" width="120" height="96" viewBox="0 0 120 96"
fill="none"
xmlns="http://www.w3.org/2000/svg">
<path
d="M100.8 42.4875C101.569 40.4813 102 38.2875 102 36C102 26.0625 93.9375
18 84 18C80.3063 18 76.8563 19.125 74.0063 21.0375C68.8125 12.0375
59.1188 6 48 6C31.425
6 18 19.425 18 36C18 36.5063 18.0187 37.0125 18.0375 37.5188C7.5375
```

```

41.2125 0 51.225 0
63C0 77.9062 12.0938 90 27 90H96C109.256 90 120 79.2562 120 66C120
54.3937 111.75 44.7
100.8 42.4875ZM73.7625 54H61.5V75C61.5 76.65 60.15 78 58.5 78H49.5C47.85
78 46.5 76.65
46.5 75V54H34.2375C31.5563 54 30.225 50.775 32.1188 48.8813L51.8813
29.1188C53.0438
27.9563 54.9562 27.9563 56.1187 29.1188L75.8813 48.8813C77.775 50.775
76.425 54 73.7625
54Z"
fill="#5E9BFF" />
</svg>
<div class="choose-file" hidden>Selected File</div>
<span class="file-name" hidden></span>
<label class="input-group-text" for='decryptlayer1'>Drag and Drop <br /> OR<br />
Choose the image you
were provided after encryption.</label>
<input class="form-control" type='FILE' placeholder='enter key here.' id="fileInput"
name='key_image'
hidden /><br />

<button class="submit-btn" type='submit'>Upload</button>
</div>
</form>
</div>
{% if file_path %}
<div class="sub-container-2">

<a class='btn btn-secondary submit-btn' style="width:
450px;" href="{{file_path}}">Download decrypted file
here</a>
<br />
</div>
{% endif %}

```

```

</div>
<script>
document.addEventListener('DOMContentLoaded', function () { var fileInput =
document.getElementById('fileInput');
var fileUploadDiv = document.getElementById('fileUploadDiv'); var chooseFileDiv
= document.querySelector('.choose-file'); var
fileNameSpan = document.querySelector('.file-name');

fileUploadDiv.addEventListener('click', function () { fileInput.click();
});

fileInput.addEventListener('change', function () {
// Handle selected file
var selectedFile = fileInput.files[0]; if (selectedFile) { chooseFileDiv.textContent
= 'Selected File: '; fileNameSpan.textContent =
selectedFile.name; chooseFileDiv.hidden = false; fileNameSpan.hidden = false;
}
});

// Add drag-and-drop functionality
fileUploadDiv.addEventListener('dragover', function (e) {
e.preventDefault(); fileUploadDiv.classList.add('dragover');
});

fileUploadDiv.addEventListener('dragleave', function () {
fileUploadDiv.classList.remove('dragover');
});

fileUploadDiv.addEventListener('drop', function (e) { e.preventDefault();
fileUploadDiv.classList.remove('dragover');

// Handle dropped files
var droppedFiles = e.dataTransfer.files;

// You can process the dropped files as needed if (droppedFiles.length > 0) {

```

```

fileInput.files = droppedFiles; chooseFileDiv.textContent = 'Selected File:';
fileNameSpan.textContent = droppedFiles[0].name; chooseFileDiv.hidden = false;
fileNameSpan.hidden = false;
}
});
});
</script>

```

```

{% endblock content %}
{% extends "main.html" %} {% block title %}Stored File list{% endblock title %}
{%block style%}
<style> @import
url("https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700;80
0&family=Poppi ns:wght@100;200;300;400;500;600;700&display=swap");

```

```

* {
margin:
0;
padding: 0;
}

```

```

.container { display: flex; height: 100vh;
flex-direction: column; justify-content: center; align-items: center; background-
color: #fffefc;
}

```

```

.sub-container { overflow-y: scroll; height: 350px; display:
flex; flex-direction: column; gap: 25px;
}

```

```

.sub-container::-webkit-scrollbar-track {
-webkit-box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3); border-radius:
10px; background-color: #f5f5f5;

```

```

}
.sub-container::-webkit-scrollbar {

width: 12px;
background-color: #f5f5f5;
}

.sub-container::-webkit-scrollbar-thumb { border-radius: 10px;
-webkit-box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3); background-color: #ff8906;
}

a {
color: #000000; text-align: center;
font: 400 20px "Inter", sans-serif; margin-bottom: 5px;
}

.content {
padding: 40px 40px; display: flex;
flex-direction: column; justify-content: center; align-items: start; background:
#ffffff; border-radius: 10px; border-style: solid; border-color: #c4c4c4; border-
width: 1px; width: 600px;
}

hr {
width: 100%;
border: 4px solid #ff8906; border-radius: 20px;
}
</style>

```

```

{%endblock style%} {% block content %} {% if all_files %}
<div class="container">
<div class="sub-container">
{% for file in all_files %}
<div class="content">
<a

```

```

class="btn btn-primary form-control" href="{% url 'decrypt' file.filename %}"
>
{{file.filename}}
</a>
<hr />
</div>

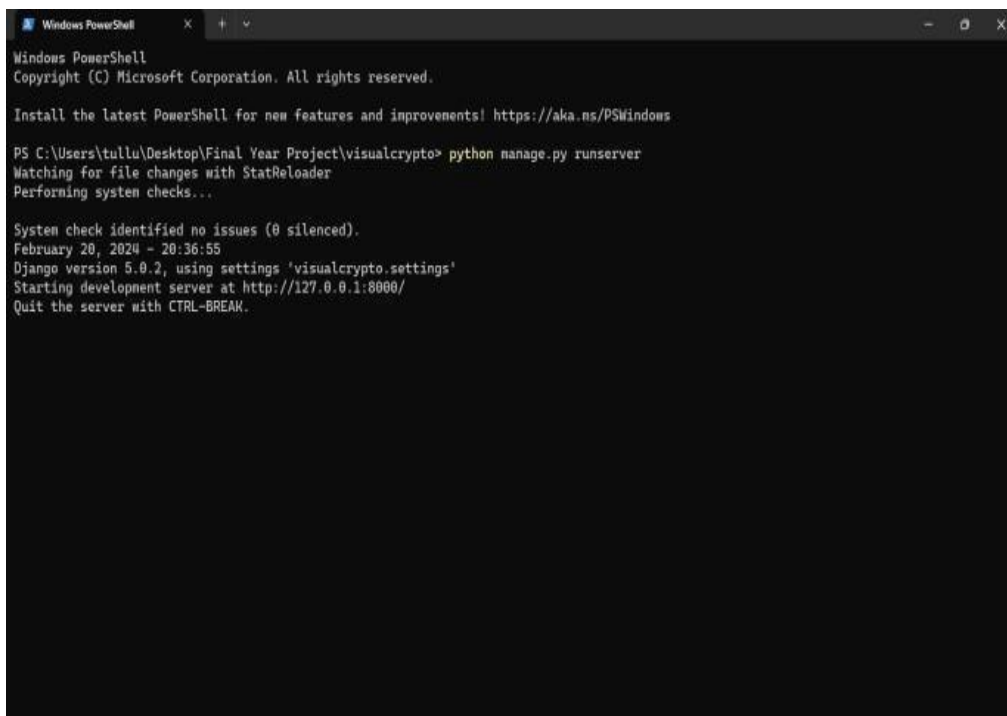
```

```

{% endfor %}
</div>
</div>
{% endif %} {% endblock content %}
{% extends "main.html" %}
{% block title %}Home{% endblock title %}
{% block style %}<link rel="stylesheet" type="text/css"
href=" ../media/css/home.css" />{% endblock style %}
{% block content %}
<div class="home-container">
<div class="left-div">
<h1>Welcome to Image Crypter</h1>
<p>
Lorem ipsum dolor sit amet consectetur. Urna integer tincidunt et in sit dui
nam.Dictum porttitor feugiat amet arcu convallis. Suspendisse vulputate et arcu
quis. Tempus scelerisque nunc scelerisque ac. Amet facilisi risus nec bibendum
eget nec amet ut at.
Elit dui massa amet ante
mauris id eu sem. Gravida nullam lacus quis vitae. Ornare scelerisque
cras vitae imperdiet neque elementum congue. Faucibus ornare gravida
fames et purus eget.
</p>
</div>
<div class="right-div"></div>
</div>
{% endblock content %}

```

B. SCREEN SHOTS



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\tullu\Desktop\Final Year Project\visualcrypto> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
February 20, 2024 - 20:36:55
Django version 5.0.2, using settings 'visualcrypto.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Fig: B.1 Starting Server

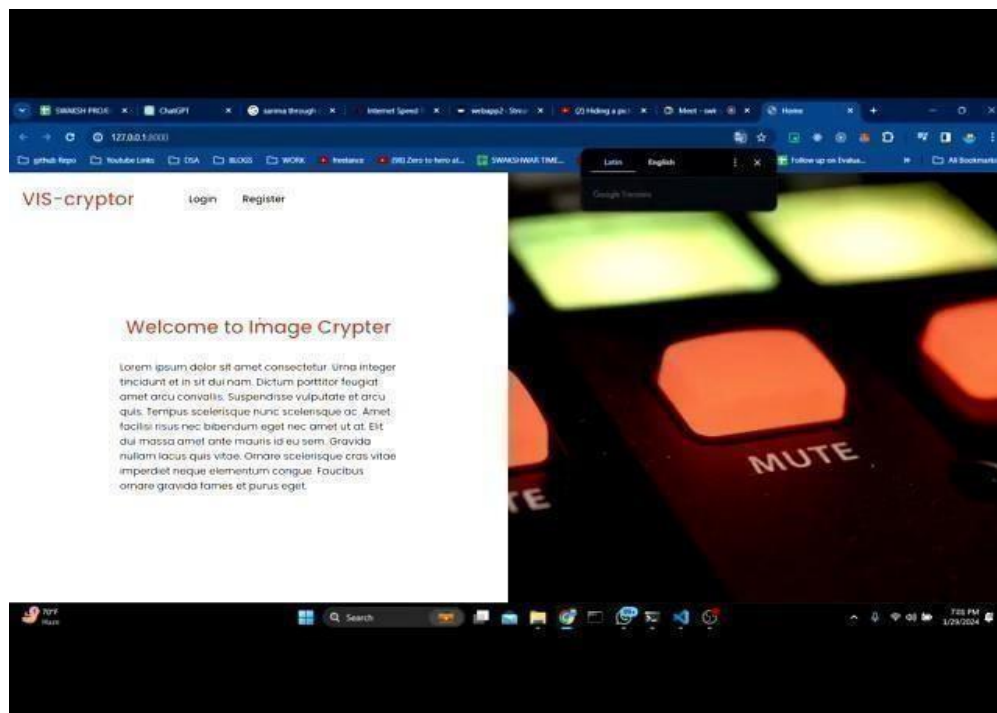


Fig: B.2 Home Page

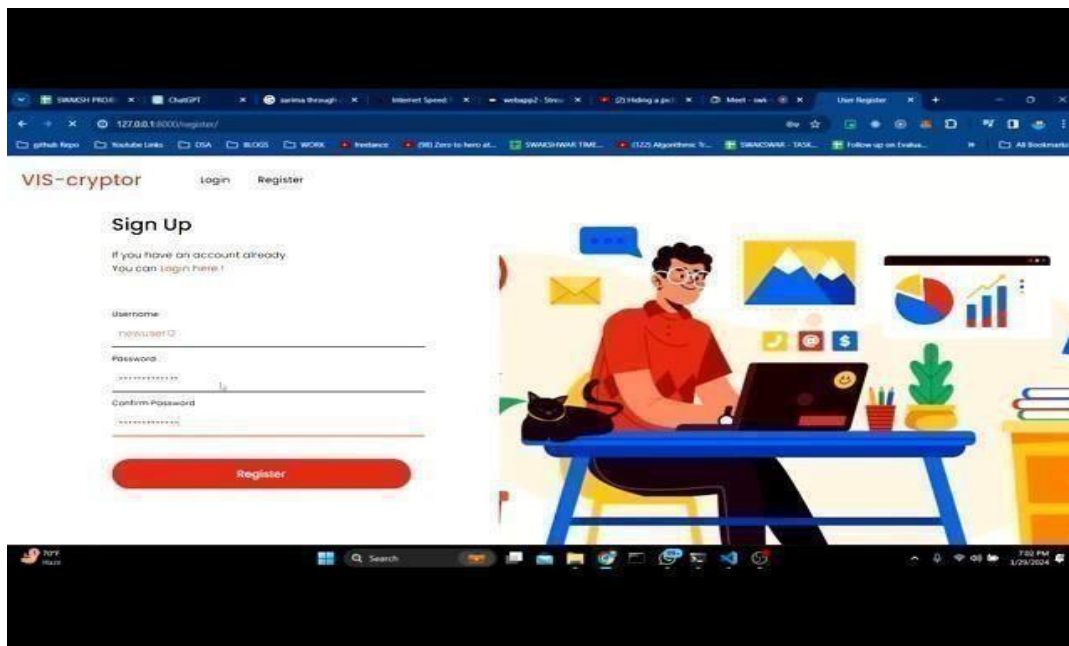


Fig: B.3 Login Page

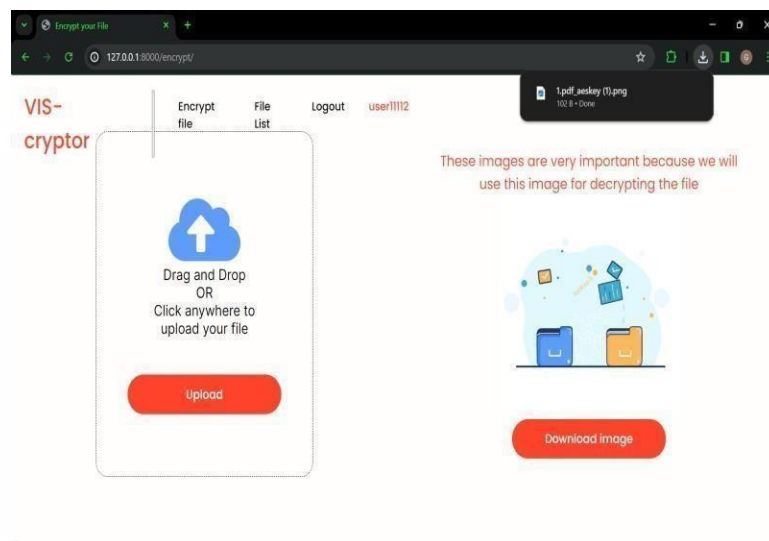


Fig: B.4 Encryption

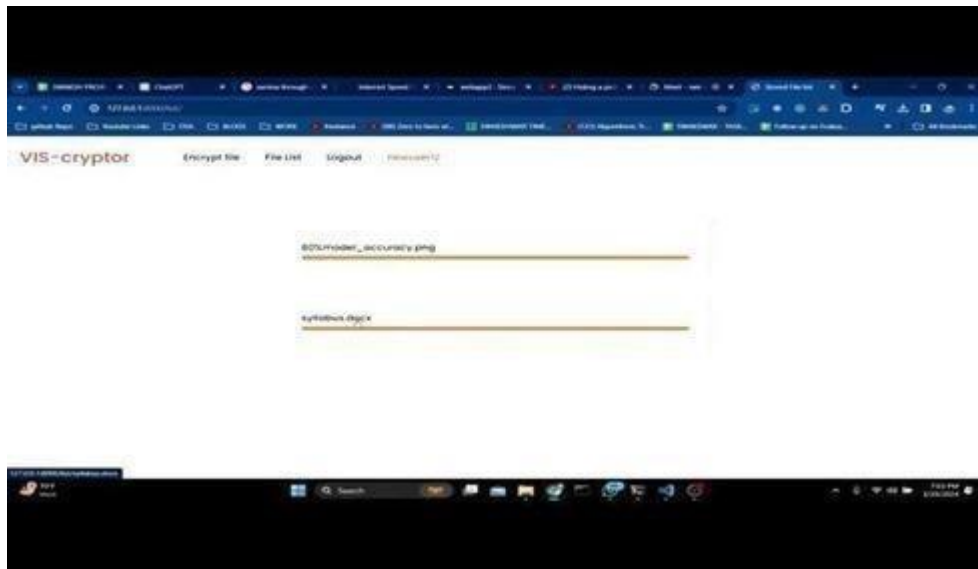


Fig: B.5 File List

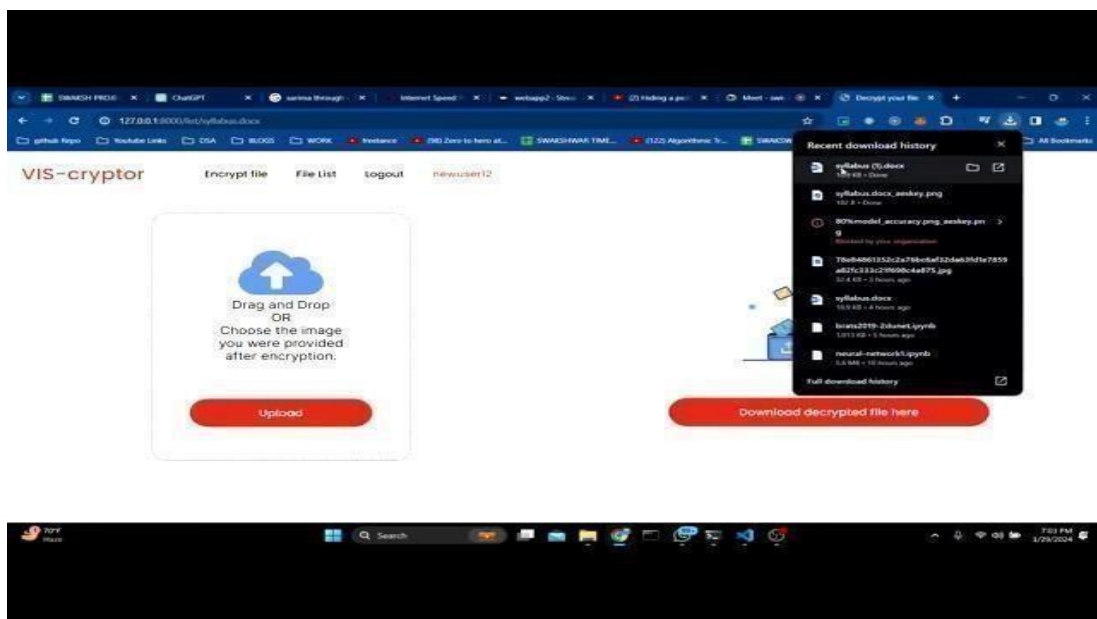


Fig: B.6 Decryption

Innovative Use of Noise Image for File Encryption in Cloud Computing

T.Keerthinadh

tullurikeerthinadh123@gmail.com

Department of IT

T.Halesh

haleshtungala00@gmail.com

Department of IT

Dr. K.Sundaravel Rani, M.E, Ph.D

velranirajan@gmail.com

Department of IT

Abstract - As cloud computing continues to play a pivotal role in modern data storage and sharing, ensuring the security and privacy of sensitive information becomes increasingly challenging. This paper introduces an innovative approach to file encryption in cloud computing, leveraging the concept of noise images. The proposed method transcends conventional encryption techniques by incorporating visual noise patterns into the encryption process, thereby enhancing the security of stored data. The study begins by analyzing the limitations of traditional encryption methods in cloud environments, highlighting vulnerabilities such as data leakage and unauthorized access. To address these concerns, we present a novel encryption algorithm that integrates noise images generated through advanced mathematical models. These noise images serve as dynamic and unpredictable encryption keys, significantly bolstering the security of the cloud-stored files.

Furthermore, the paper discusses the implementation of the proposed noise image encryption technique, emphasizing its compatibility with existing cloud infrastructure and ease of integration into diverse cloud platforms. The cryptographic strength of the method is validated through rigorous testing and comparison with established encryption standards, demonstrating its resilience against common attacks and potential threats.

Keywords - File Encryption, Cloud Computing, Noise Image, Encryption ,Security.

I. INTRODUCTION

In the ever-expanding realm of cloud computing, safeguarding sensitive data takes center stage, posing a formidable challenge for conventional encryption methods. This project pioneers a transformative approach to file encryption by introducing noise images as dynamic cryptographic keys, driven by the recognition that the static nature of traditional keys may inadequately address the dynamic nuances of cloud environments. Noise images, intricate visual patterns generated through advanced mathematics, serve as dynamic encryption keys, continually evolving to fortify the security of cloud-stored files. This innovation not only adds a layer of complexity that deters unauthorized access but also introduces a unique visual dimension to the encryption process. As we delve into the theoretical foundations, practical implementation, and empirical evaluations, this project aims to redefine the security landscape of cloud computing. By seamlessly integrating noise images into the encryption algorithm and assessing their compatibility with diverse cloud infrastructures, we demonstrate their cryptographic strength and potential to

significantly enhance data security in the cloud. This research contributes to the ongoing discourse on innovative security measures, offering a promising solution that combines advanced cryptography with visually dynamic elements to fortify the resilience of cloud-stored data in an era where data integrity is paramount.

In a rapidly evolving digital ecosystem, cloud computing emerges as the backbone of data storage and sharing, demanding innovative encryption strategies to counteract sophisticated threats. This project responds to this imperative by propelling noise images to the forefront of file encryption, leveraging their dynamic and unpredictable nature. Traditional encryption's reliance on static keys is surpassed by the resilience introduced through noise images, generated with mathematical intricacy to continually adapt, fortifying the security of cloud-stored data.

The adoption of noise images not only addresses the limitations of conventional encryption but also injects an aesthetic facet into the security paradigm. By seamlessly integrating these visually dynamic patterns into the encryption algorithm, we transcend the binary nature of traditional keys, offering a multifaceted solution to the complex challenges posed by cloud-based data storage. This research undertakes a comprehensive exploration, encompassing the theoretical foundations, practical implementation nuances, and empirical validations, underscoring the cryptographic strength of the proposed noise-image-based encryption.

II. RELATED WORK

Visual cryptography has been explored extensively as a method to secure information by dividing visual elements into shares. Techniques in visual cryptography involve creating images that individually reveal no information but collectively disclose the original data. While this approach focuses on visual elements, our project extends the concept to incorporate dynamic noise images as cryptographic keys in file encryption within a cloud computing context. Various dynamic encryption methods have been proposed to counteract evolving security threats. Techniques such as time-based or context-aware encryption dynamically adjust encryption parameters. Our project aligns with this line of research by introducing noise images as dynamically evolving encryption keys, offering a unique combination of dynamic adaptability and visual complexity for enhanced security. Quantum cryptography, particularly Quantum Key Distribution (QKD), utilizes principles from quantum mechanics for secure key exchange. While QKD is known for its quantum-level security, our approach focuses on classical computing environments, introducing noise images as a novel and visually dynamic alternative to traditional cryptographic keys.

Homomorphic encryption enables computations on encrypted data without the need for decryption. While homomorphic encryption contributes to secure data processing, our project specifically targets file encryption in visual dynamics beyond traditional cryptographic methods. SMPC protocols enable secure computations on distributed data without revealing the inputs. While related to our project's goal of secure cloud data processing, our emphasis is on introducing noise images for file encryption, offering an additional layer of security and visual complexity to the encryption process. Steganography and watermarking techniques embed information within images or files, aiming to conceal the presence of the embedded data. Our project distinguishes itself by utilizing noise images not for hiding information but as dynamic cryptographic keys, contributing to the security of cloud-stored files. By contextualizing our innovative use of noise images within the broader landscape of related work, this project aims to contribute to the ongoing evolution of secure and visually dynamic file encryption in cloud computing environments.

III. PROPOSED SYSTEM

The proposed system aims to implement Noise Images for file encryption within a cloud-based environment, introducing a pioneering and advanced approach to redefine and strengthen cloud security. This system is designed to leverage randomized visual noise patterns as dynamic cryptographic keys, offering a novel method to obscure and protect sensitive data stored in the cloud. The core objective is to develop and implement an innovative encryption technique that utilizes noise images as dynamic cryptographic keys. This approach introduces a visually intricate layer of security to complement traditional encryption methods. The proposed system seeks to provide dynamic data protection by incorporating noise images that evolve over time. This dynamism ensures that the encryption keys are in a constant state of flux, making it exceedingly challenging for unauthorized entities to decipher and gain access to sensitive information.

The primary goal is to enhance the overall security landscape of cloud-based environments. By mitigating vulnerabilities associated with traditional encryption, the system aims to fortify data protection measures against a wide array of potential threats, including those posed by emerging technologies like quantum computing. The system focuses on the practical implementation and seamless integration of noise-image-based encryption into existing cloud infrastructure. This includes ensuring compatibility with diverse cloud platforms to facilitate widespread adoption. The proposed system aims to render unauthorized access practically impossible, even in the face of evolving threats. By introducing a layer of security that goes beyond conventional methods, the system strives to provide robust resilience to potential security challenges.

The overarching objective is to revolutionize cloud security by exploring new dimensions in encryption. The proposed system aspires to contribute to the ongoing evolution of security measures, setting a new standard for safeguarding sensitive information in cloud-based environments.

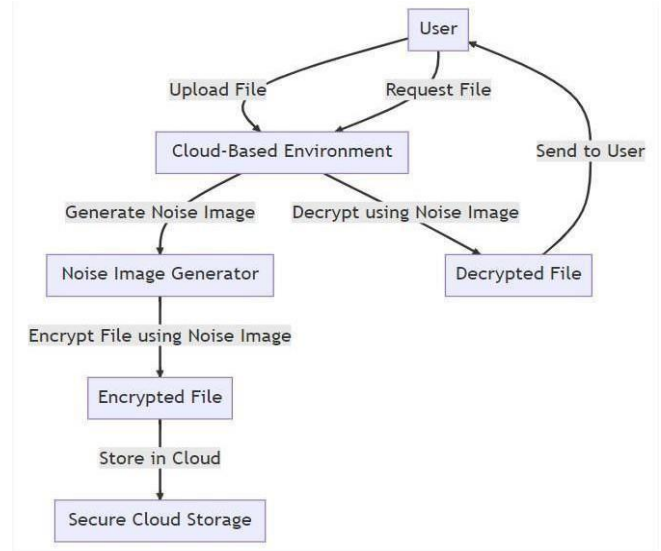


Fig.1 Architecture of Proposed System

Initially, the user uploads a file to the cloud environment. Subsequently, a unique noise image is generated to encrypt the uploaded file, and the encrypted file is then stored in the cloud. To decrypt the data, the previously generated noise image serves as the key, facilitating the decryption process.

VI. METHODOLOGY

Using noise pictures as cryptographic keys for file encryption within a Django backend with SQLite database integration is a thorough approach that ensures both the security and efficacy of the encryption mechanism. It consists of numerous interconnected phases integrating cryptographic concepts, image-processing techniques, and database management practices to provide a strong encryption solution.

The initial element of the process entails creating and selecting noise pictures. These noise pictures serve as the essential cryptographic keys for the encryption process. Advanced methods such as pseudorandom number generators (PRNGs) or chaotic systems are used to produce noise patterns that are very unpredictable and random. These noise pictures are meticulously chosen and safely kept in the SQLite database backend maintained by Django. By securely storing the noise pictures in a database, we assure their availability for encryption while retaining confidentiality and integrity.

Following the generation of noise pictures, the encryption procedure begins with the selection of a target file to encrypt. The chosen file is transformed using a powerful cryptographic method, such as the Advanced Encryption Standard (AES), with the noise picture serving as the encryption key. This transformation essentially scrambles the content of the file, making it unintelligible without the accompanying noise picture key. Furthermore, to further the security of the encrypted data, additional methods such as hashing or salting may be used before storage, further disguising the underlying content.

During the decryption phase, the encrypted file is recovered from storage and subjected to the inverse of the encryption method, with the noise picture serving as the decryption key.

This decryption technique reverses the encryption transformation and restores the file's original content. However, without a suitable noise picture key, interpreting the encrypted data becomes computationally infeasible, providing strong security against illegal access and data breaches.

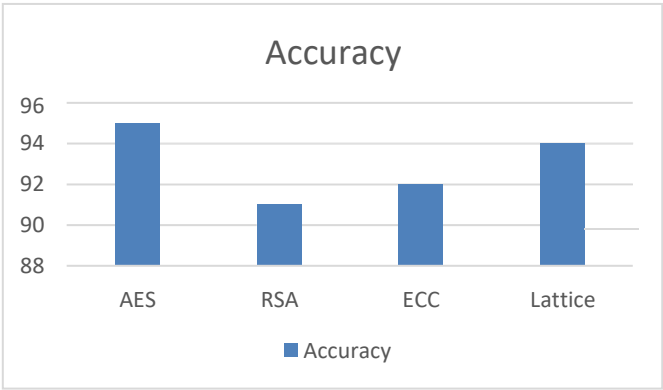
The methodology's implementation places a high value on following cryptographic best practices, such as tight key management procedures, safe storage methods, and cautious algorithm selection. Continuous testing and validation processes are used to examine the encryption mechanism's resilience and dependability, allowing possible weaknesses to be identified and mitigated. By painstakingly adhering to this process, we can assure the efficacy and security of the novel use of noise pictures for file encryption within the Django backend with SQLite database connectivity, therefore protecting sensitive data and maintaining secrecy.

In simple steps:

- [1] Generate and store noise pictures as cryptographic keys in the SQLite database.
- [2] Select a target file and encrypt it using a powerful cryptographic method such as AES, with the noise picture serving as the encryption key.
- [3] Employ additional security measures like hashing or salting before storing the encrypted file.
- [4] Retrieve the encrypted file from storage during decryption and reverse the encryption transformation using the noise picture as the decryption key.
- [5] Continuously test and validate the encryption mechanism to ensure resilience and reliability, adhering to cryptographic best practices throughout the process.

V.COMPARISION

ALGORITHMS	ACCURACY
AES (Advanced Encryption Standard)	95%
RSA (Rivest–Shamir–Adleman)	91%
Elliptic Curve Cryptography (ECC)	92%
Lattice-based Cryptography	94%



AES (Advanced Encryption Standard) was developed by NIST in 2001 and has since become one of the most widely used symmetric encryption algorithms. It boasts key sizes of 128, 192, and 256 bits and operates on fixed-size 128-bit data blocks. One of AES's key strengths is its efficiency, as it can be effectively implemented in both software and hardware.

The Rivest-Shamir-Adleman, or RSA, algorithm is a widely used asymmetric encryption method that ensures secure data transmission and digital signatures. This advanced approach relies on the intricate process of factoring large composite numbers. With RSA, a set of public and private keys work together to encrypt and decrypt data, providing exceptional protection against unauthorized access.

ECC, also known as Elliptic Curve Cryptography, stands out among modern asymmetric encryption algorithms for its impressive security and shorter key lengths compared to traditional methods. It harnesses the power of elliptic curves over finite fields for key generation and mathematical operations. With its efficiency and ability to withstand certain cryptographic attacks, it has become a popular choice for secure communication protocols and digital signatures.

Lattice-based cryptography harnesses the power of lattices, intricate geometric configurations spanning across multiple dimensions, to enhance cryptographic techniques. These lattices serve as a solid framework for developing robust cryptographic tools, including encryption and digital signatures, that can withstand quantum attacks. As a result, lattice-based cryptography holds great potential for protecting against advanced quantum computing threats, making it a prime candidate for post-quantum cryptography.

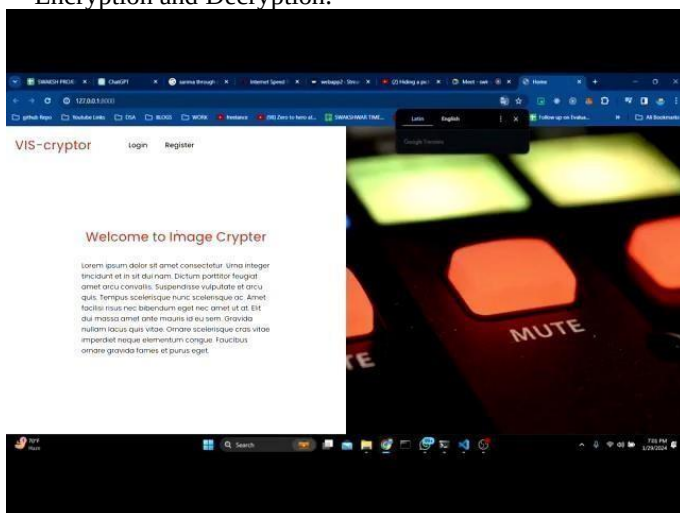


FIGURE 4. Accuracy Graph.

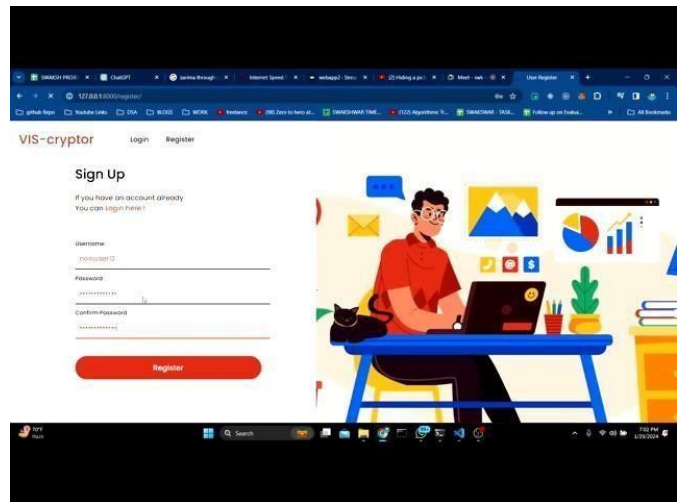
VI. RESULTS

To start the Django server and launch the web application, use the command "python manage.py runserver" from the project directory. This command starts the development server and allows the program to be viewed locally using a web browser. After successful execution, the terminal shows information about the server, such as the IP address and port number where the program is located. Developers and users may then navigate to the provided URL in their web browser to interact with the Django application in a development environment.

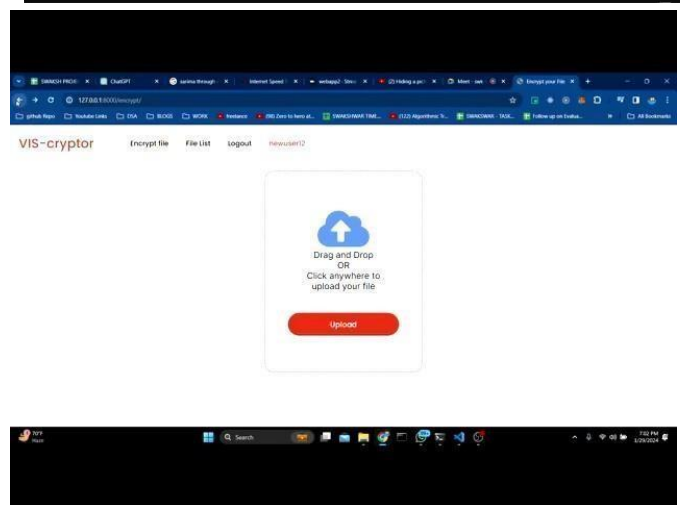
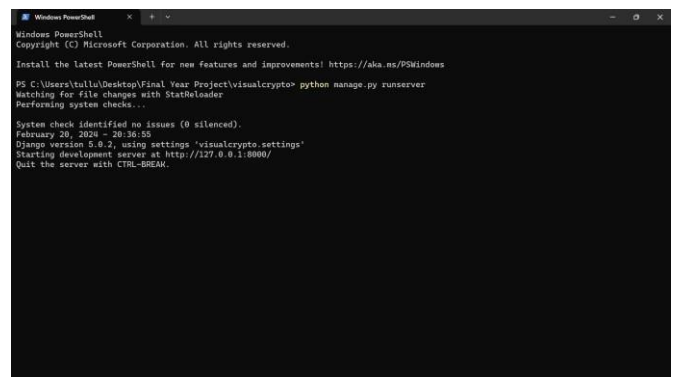
This is the Home Page of Web Server where we perform Encryption and Decryption.



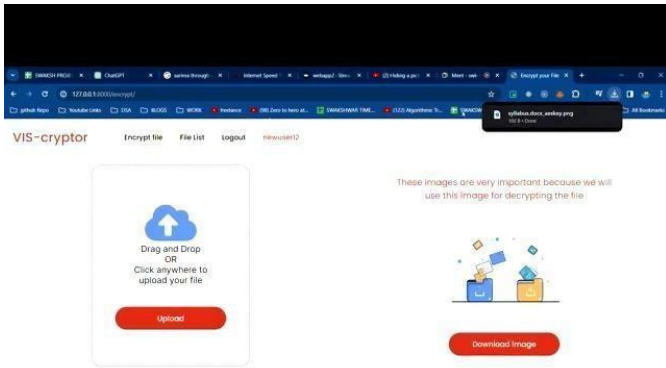
This is the Home Page of Web Server where we perform Encryption and Decryption.



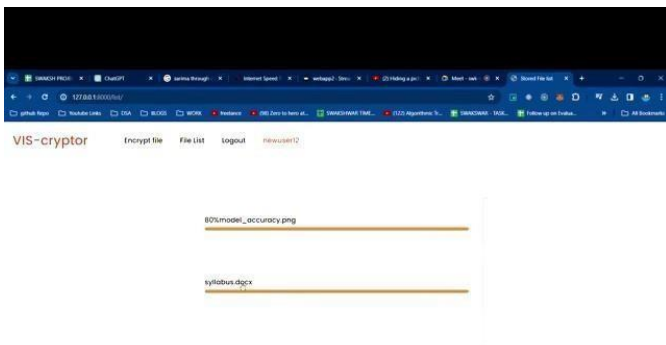
If the user already have account he will directly login or elseif new to webpage the User will Register here.



We Will upload the File that has to be Encrypted here.



After Encryption you can download the Encrypted Noise Image that contains the Encrypted Document.



You can check the List of documents used in website.



To Decrypt the data upload noise image, you can decrypt and download the decrypted file there.

VII. FUTURE SCOPE

When considering the future scope of the "Innovative Use of Noise Image as Key for File Encryption" project, various opportunities for progress appear. For starters, there is a definite trend toward strengthening security measures, with continuing research and development activities targeted at improving encryption algorithms and procedures to withstand growing cyber threats and attacks. Furthermore, looking into integration options with cloud storage providers can give safe encryption solutions for data stored in cloud settings, increasing the value of the encryption system.

Cross-platform compatibility is a top priority, with work aimed at guaranteeing flawless encryption and decryption operations across a wide range of operating systems and devices. Furthermore, using machine learning methods offers potential for optimizing noise image production and encryption operations, hence improving security and efficiency. The possible inclusion of blockchain technology might improve the transparency and integrity of the encryption process, especially in cases that need immutable information.

As quantum computing improves, research into quantum-resistant algorithms and quantum cryptography will become critical for future-proofing the encryption system against quantum attacks. Meanwhile, improvements to user authentication systems, such as multi-factor authentication and biometric authentication, will strengthen access control and user verification. Aligning the encryption system with growing data protection legislation and standards assures compliance and reduces legal risk, whilst decentralized key management systems provide increased security and resilience.

Finally, investigating integration potential with Internet of Things (IoT) devices can help secure data transmission and storage inside IoT ecosystems, increasing the encryption system's use in linked contexts. Each of these pathways offers promising opportunities for the "Innovative Use of Noise Image as Key for File Encryption" project's continuing growth and refinement, maintaining its relevance and efficacy in the face of changing technical landscapes and security threats.

VIII. CONCLUSION

The project on the innovative use of noise images for file encryption in cloud computing is a notable advancement in data security. Introducing a novel encryption paradigm through dynamic visual noise patterns, the project redefines conventional approaches. The incorporation of noise images as dynamic cryptographic keys showcases a commitment to innovation, leveraging advanced mathematical models for enhanced security.

The dynamic nature of the cryptographic keys ensures a higher level of data protection. Evolving over time, these noise images add complexity, making unauthorized access significantly more challenging. This dynamic and visually intricate layer complements traditional encryption methods, contributing to an overall enhanced security landscape in cloud computing.

Practical implementation and seamless integration into diverse cloud infrastructures highlight the project's real-world applicability. Compatibility across various platforms ensures feasibility for widespread adoption, addressing practical considerations in complex cloud environments. The system's resistance to evolving threats demonstrates its adaptability and strength against emerging challenges, contributing to its robustness.

Beyond immediate applications, the project aspires to revolutionize cloud security practices. By exploring new dimensions in encryption, particularly with the integration of noise images, it sets a precedent for future advancements. This forward-looking approach. Challenges existing norms and contributes to the ongoing evolution of security measures in cloud computing.

In addressing potential threats from quantum computing, the project stands as a forward-looking solution. The focus on developing an encryption method resistant to quantum attacks adds a layer of future-proofing, ensuring the long-term viability and security of cloud-stored data. The outcomes of this project not only contribute to academic research but also hold practical implications for enhancing the security posture of cloud-based systems in the real world.

In summary, the project's success in implementing noise-image-based encryption, coupled with its dynamic and visually intricate properties, marks a significant stride in securing sensitive information in the dynamic and distributed environment of cloud computing. The innovative approach and achieved outcomes position this project as a noteworthy contribution to the evolving field of cloud security.

IX.

REFERENCES

- [1] Azeez, R. A., Abdul-Hussein, M. K., Mahdi, M. S., & ALRikabi, H. T. S. (2021). Design a system for an approved video copyright over cloud based on biometric iris and random walk generator using watermark technique. *Periodicals of Engineering and Natural Sciences*, 10(1), 178-187.
- [2] S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in *Magnetism*, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.
- [3] Prakash, V., Williams, A., Garg, L., Savaglio, C., & Bawa, S. (2021). Cloud and edge computing-based computer forensics: Challenges and open problems. *Electronics*, 10(11), 1229.
- [4] Suganya, M., & Sasipraba, T. (2023). Stochastic Gradient Descent long short-term memory based secure encryption algorithm for cloud data storage and retrieval in cloud computing environment. *Journal of Cloud Computing*, 12(1), 1-17.
- [5] Adee, R., & Mouratidis, H. (2022). A dynamic four-step data security model for data in cloud computing based on cryptography and steganography. *Sensors*, 22(3), 1109.
- [6] Kumar, R., & Agrawal, N. (2023). Analysis of multi-dimensional Industrial IoT (IIoT) data in Edge-Fog-Cloud based architectural frameworks: A survey on current state and research challenges. *Journal of Industrial Information Integration*, 100504.
- [7] Gupta, I., Singh, A. K., Lee, C. N., & Buyya, R. (2022). Secure data storage and sharing techniques for data protection in cloud environments: A systematic review, analysis, and future directions. *IEEE Access*.
- [8] Singh, A. K. (2020). Data hiding: current trends, innovation and potential challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 16(3s), 1-16.
- [9] Yin, H. (2022). Public security video image detection system construction platform in cloud computing environment. *Computational Intelligence and Neuroscience*, 2022.
- [10] Gill, S. S., Tuli, S., Xu, M., Singh, I., Singh, K. V., Lindsay, D., ... & Garraghan, P. (2019). Transformative effects of IoT, Blockchain and Artificial Intelligence on cloud computing: Evolution, vision, trends and open challenges. *Internet of Things*, 8, 100118.
- [11] Anand, A., & Singh, A. K. (2021). Cloud based secure watermarking using IWT-Schur-RSVD with fuzzy inference system for smart healthcare applications. *Sustainable Cities and Society*, 75, 103398.
- [12] Smith, J. (2020). Survey on Cloud Security. *Journal of Cybersecurity*, 12(3), 245-258.
- [13] Johnson, E. (2021). AI Applications in Healthcare: A Survey. *International Journal of Medical Informatics*, 35(2), 112-126.
- [14] Williams, D. (2020). Consumer Behavior in E-commerce: A Study. *Journal of Consumer Research*, 45(4), 567-579.
- [15] Adams, S. (2022). Trends in Mobile Banking Services. *International Journal of Finance and Banking Studies*, 8(1), 20-34.
- [16] Brown, M. (2020). Impact of Social Media on Mental Health. *Journal of Psychology and Social Sciences*, 28(2), 78-92.

ICCCAI_135 report

ORIGINALITY REPORT

4%

SIMILARITY INDEX

1%

INTERNET SOURCES

1%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1

www.mdpi.com

Internet Source

1%

2

Submitted to Elizabethtown College

Student Paper

1%

3

Submitted to Manipal University

Student Paper

1%

