

An Industrial Oriented Mini Project

Report on

IMAGE ENCRYPTION USING BLOWFISH ALGORITHM

**Submitted in Partial fulfillment of requirements for the award of the
degree of**

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING (AI & ML)

By

Y SHRIYA SRAVANI 20BD1A66C0

N KARTHIK 20BD1A6694

R CHETNA 20BD1A66A3

T JAHANAVI 20BD1A66B3

Under the guidance of

Mrs. R Priyanka

Assistant Professor, Department of CSE (AI & ML)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI & ML)

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH.

Narayanaguda, Hyderabad, Telangana-29

2023-24



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(AI & ML)**

CERTIFICATE

This is to certify that this is a bonafide record of the project report titled **“Image Encryption using Blowfish algorithm”** which is being presented as the Industrial Oriented Mini Project report by

- | | |
|----------------------------|-------------------|
| 1. Y SHRIYA SRAVANI | 20BD1A66C0 |
| 2. N KARTHIK | 20BD1A6694 |
| 3. R CHETNA | 20BD1A66A3 |
| 4. T JAHANAVI | 20BD1A66B3 |

In partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering (AI & ML) affiliated to the Jawaharlal Nehru Technological University Hyderabad, Hyderabad

**Internal Guide
(Mrs. R Priyanka)**

**Head of Department
(Dr. T.V.G Sridevi)**

Submitted for Viva Voce Examination held on _____

External Examiner

Vision of KMIT

- To be the fountainhead in producing highly skilled, globally competent engineers.
- Producing quality graduates trained in the latest software technologies and related tools and striving to make India a world leader in software products and services.

Mission of KMIT

- To provide a learning environment that inculcates problem solving skills, professional, ethical responsibilities, lifelong learning through multi modal platforms and prepares students to become successful professionals.
- To establish an industry institute Interaction to make students ready for the industry.
- To provide exposure to students on the latest hardware and software tools.
- To promote research-based projects/activities in the emerging areas of technology convergence.
- To encourage and enable students to not merely seek jobs from the industry but also to create new enterprises.
- To induce a spirit of nationalism which will enable the student to develop, understand India's challenges and to encourage them to develop effective solutions.
- To support the faculty to accelerate their learning curve to deliver excellent service to students.

VISION & MISSION OF DEPARTMENT

Vision of the Department

To be among the region's premier teaching and research Computer Science and Engineering departments producing globally competent and socially responsible graduates in the most conducive academic environment.

Mission of the Department

- To provide faculty with state-of-the-art facilities for continuous professional development and research, both in foundational aspects and of relevance to emerging computing trends.
- To impart skills that transform students to develop technical solutions for societal needs and inculcate entrepreneurial talents.
- To inculcate an ability in students to pursue the advancement of knowledge in various specializations of Computer Science and Engineering and make them industry-ready.
- To engage in collaborative research with academia and industry and generate adequate resources for research activities for seamless transfer of knowledge resulting in sponsored projects and consultancy.
- To cultivate responsibility through sharing of knowledge and innovative computing solutions that benefits the society-at-large.
- To collaborate with academia, industry and community to set high standards in academic excellence and in fulfilling societal responsibilities.

PROGRAM OUTCOMES (POs)

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem Analysis:** Identify formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/Development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct Investigations of Complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage:** Create select, and, apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society:** Apply reasoning informed by contextual knowledge to societal, health, safety. Legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-Long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: An ability to analyze the common business functions to design and develop appropriate Computer Science **solutions** for social upliftments.

PSO2: Shall have expertise on the evolving technologies like Mobile Apps, CRM, ERP, Big Data, etc.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO1: Graduates will have successful careers in computer related engineering fields or will be able to successfully pursue advanced higher education degrees.

PEO2: Graduates will try and provide solutions to challenging problems in their profession by applying computer engineering principles.

PEO3: Graduates will engage in life-long learning and professional development by rapidly adapting changing work environment.

PEO4: Graduates will communicate effectively, work collaboratively and exhibit high levels of professionalism and ethical responsibility.

PROJECT OUTCOMES

O1: Efficient image sharing and access between users.

O2: Robust key management for enhanced security.

O3: User-friendly interface for easy sharing.

O4: Authenticated system to maintain user privacy.

MAPPING PROJECT OUTCOMES WITH PROGRAM OUTCOMES

PO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
O1	M	M	H	H	M			H	H	M	M	M
O2	M	M	H	H	H	H	M	H	H	M		M
O3			H	M				H	H	M	H	M
O4		M	M	H	M	H	M	H	H	M	M	H

L – Low

M –Medium

H– High

**PROJECT OUTCOMES MAPPING
PROGRAM SPECIFIC OUTCOMES**

PSO	PSO 1	PSO2
P1	H	M
P2	M	H
P3	H	H
P4	H	H

**PROJECT OUTCOMES MAPPING
PROGRAM EDUCATIONAL OBJECTIVES**

PEO	PEO 1	PEO 2	PEO 3	PEO 4
P1		H	H	M
P2	M	M	M	H
P3	H	H		H
P4	H		M	H

DECLARATION

We hereby declare that the results embodied in the dissertation entitled “Image Encryption using Blowfish algorithm” has been carried out by us together during the academic year 2023-24 as a partial fulfillment of the award of the B.Tech degree in Computer Science and Engineering (AI & ML) from JNTUH. We have not submitted this report to any other university or organization for the award of any other degree.



Student Name	Roll No.
Y SHRIYA SRAVANI	20BD1A66C0
N KARTHIK	20BD1A6694
R CHETNA	20BD1A66A3
T JAHANAVI	20BD1A66B3

ACKNOWLEDGEMENT

We take this opportunity to thank all the people who have rendered their full support to our project work. We render our thanks to **Dr. B L Malleswari**, Principal who encouraged us to do the Project.

We are grateful to **Mr. Neil Gogte**, Founder & Director, **Mr. S. Nitin**, Director for facilitating all the amenities required for carrying out this project.

We express our sincere gratitude to **Ms. Deepa Ganu**, Director Academic for providing an excellent environment in the college.

We are also thankful to **Dr. TVG Sridevi**, Head of the Department for providing us with time to make this project a success within the given schedule.

We are also thankful to our guide **Mrs. R Priyanka**, for her valuable guidance and encouragement given to us throughout the project work.

We would like to thank the entire Computer Science and Engineering (AI & ML) Department faculty, who helped us directly and indirectly in the completion of the project.

We sincerely thank our friends and family for their constant motivation during the project work.

Student Name	Roll No.
Y SHRIYA SRAVANI	20BD1A66C0
N KARTHIK	20BD1A6694
R CHETNA	20BD1A66A3
T JAHANAVI	20BD1A66B3

ABSTRACT

At the heart of today's information age lies a torrent of data—personal, financial, and organizational—that traverses networks, resides in databases, and resides on individual devices. This surge in data traffic has, in turn, amplified the risks associated with unauthorized access, interception, and malicious exploitation. As a response to this pressing concern, encryption algorithms have emerged as the guardians of digital information, ensuring that sensitive data remains confidential, even in the face of sophisticated cyber threats.

Our project on "Secure Image Sharing Using Blowfish Algorithm" is an innovative solution designed to address the growing need for secure image sharing and transmission in various domains, including healthcare, military, and confidential document exchange. The project leverages the robust Blowfish encryption algorithm to provide a secure and efficient method for encrypting and sharing image data. The Blowfish algorithm is utilized to encrypt image files, ensuring the confidentiality and integrity of the data. The system includes features for key management and secure transmission, allowing authorized users to securely share and access encrypted images. The encryption key generation process is designed to be robust and secure, enhancing the overall protection of the images. Additionally, the project promotes user-friendly interactions through a well-designed user interface and secure authentication methods.

"Secure Image Sharing Using Blowfish Algorithm" offers a practical and reliable solution for organizations and individuals seeking to safeguard sensitive image data while facilitating secure image sharing and collaboration. This project contributes to enhancing data security and privacy in an era where secure image sharing is of paramount importance.

LIST OF ABBREVIATIONS

DES	Data Encryption Standard
AES	Advanced Encryption Standard
IDEA	International Data Encryption Algorithm
p-array	Permutation array
s-box	Substitution box
f-function	Feistel Function
IDE	Integrated Development Environment

LIST OF DIAGRAMS

S.No	Name of the Diagram	Page No.
1.	System Architecture Diagram	4
2.	Use Case Diagram	19
3.	Sequence Diagram	20
4.	State Chart Diagram	21
5.	Deployment Diagram	22
6.	Jdbc Architecture	25
7.	Cryptography Architecture	27
8.	Encryption and decryption process	28
9.	Flow of Blowfish Algorithm	28
10.	Feistel Cipher Structure	31
11.	Feistel Cipher Function (F-function)	32
12.	Feistel Structure in Blowfish Algorithm	33
13.	Encryption in Blowfish Algorithm	35
14.	Decryption in Blowfish Algorithm	36
15.	Original Images	49

S.No	Name of the Diagram	Page No.
16.	Decrypted Images	52
17.	Performance of Blowfish on various file size	56



LIST OF TABLES

S.No	Name of Table	PageNo.
1.	Client Requirement	17
2.	Sever Requirements	17
3.	MySQL Database	39
4.	Structure of Cloud Table of Database	40
5.	Structure of Owner Table of Database	40
6.	Structure of Temp Table of Database	41
7.	Structure of User Table of Database	41
8.	Comparison Table of Symmetric Key Algorithms	55

LIST OF SCREENSHOTS

S.No	Name of Screenshot	PageNo.
1.	User Interface (UI) of Data owner registration	48
2.	User Interface (UI) of Data owner login	48
3.	User Interface (UI) of Image Upload	49
4.	User Interface (UI) of File Sharing	50
5.	User Interface (UI) of Data owner registration	50
6.	User Interface (UI) of Data owner login	51
7.	User Interface (UI) of Image download for data user (a)	51
8.	User Interface (UI) of Image download for data user (b)	52
9.	User Interface (UI) of Image download for data user (c)	52

CONTENTS

<u>DESCRIPTION</u>	<u>PAGE</u>
CHAPTER - 1	1
1. INTRODUCTION	2-5
1.1 Purpose of the project	2
1.2 Problem with Existing Systems	2
1.3 Proposed System	3
CHAPTER – 2	6
2. LITERATURE SURVEY	6-11
2.1 Review of relevant research papers	7
2.2 Relevant technologies and tools	10
CHAPTER - 3	12
3. SOFTWARE REQUIREMENT SPECIFICATION	13-18
3.1 Introduction to SRS	13
3.2 Functional Requirements	13
3.3 Non-Functional Requirements	14
3.4 Performance Requirements	15
3.5 Software Requirements	17
3.6 Hardware Requirements	17
CHAPTER – 4	18
4. SYSTEM DESIGN	19-25
4.1 Introduction to UML	19
4.2 UML Diagrams	19
4.2.1 Use Case Diagram	19
4.2.2 Sequence Diagram	20
4.2.3 State Chart Diagram	21
4.2.4 Deployment Diagram	22
4.3 Technologies used	22

CHAPTER – 5	26
5. IMPLEMENTATION	27-52
5.1 Working Principle	27
5.1.1 Cryptography	27
5.1.2 Blowfish Algorithm	28
5.2 Description of Approach	37
5.3 Data Analysis	38
5.4 Project Execution	39
5.4.1 Databases created and used	39
5.4.2 Code for Implementation	41
5.4.3 Functional Modules	48
CHAPTER – 6	53
6. RESULTS	54-60
6.1 Expected results	54
6.2 Comparing Blowfish	55
6.3 Applications	56
6.4 Merits	58
6.5 Demerits	59
CONCLUSION	61
FUTURE ENHANCEMENTS	62
BIBLIOGRAPHY	64

CHAPTER - 1



1. INTRODUCTION

1.1 Purpose of the project

In the ever-evolving landscape of information technology, the importance of securing sensitive data has never been more critical. As our world becomes increasingly interconnected, the need for robust encryption algorithms to protect confidential information has become paramount. The field of encryption is gaining significance in the present era, as image security becomes a paramount concern with the escalating severity of web attacks. Image encryption and decryption find applications in internet communication, multimedia systems, medical imaging, telemedicine, military communication, etc. Numerous image content encryption algorithms have been proposed to secure data from various attacks and ensure data integrity before transmission or storage.

The Blowfish encryption algorithm, conceived by renowned cryptographer Bruce Schneier in 1993, stands as a testament to innovation in cryptographic methodologies. What sets Blowfish apart is its unique combination of strength, speed, and simplicity. Unlike its predecessors, Blowfish employs a variable key length, making it adaptable to different security requirements—a feature that has contributed significantly to its enduring relevance.

The core strength of the Blowfish algorithm lies in its ability to execute rapid encryption and decryption processes without compromising security. This efficiency is particularly vital in contemporary applications where the demand for real-time data processing is incessant. Blowfish's computational speed, coupled with its proven security features, makes it a go-to choice for scenarios where performance is as crucial as safeguarding information.

As we dive into our mini project exploring Blowfish, we're not just reading about it—we're going to see how it actually works. We want to understand why it's good at keeping our info safe and how we can use it in our digital world.

In summary, our mini project on Blowfish encryption is a tribute to the powerful combination of technology and digital security. It marks a significant leap forward in making the protection of our digital information more accessible and effective. By delving into the intricacies of Blowfish, we aim to simplify the complex world of encryption, ensuring that individuals and organizations can safeguard their sensitive data.

1.2 Problem with the existing system

In the fast-paced digital era, safeguarding information has become a top priority, especially when it comes to sharing images. Whether it's personal photos, sensitive documents, or crucial data, the need for secure transmission is crucial. This is where Blowfish, a robust encryption algorithm, comes into play.

As the digital landscape evolves, so do the risks associated with sharing sensitive content. Images, being a significant part of our online communication, require a reliable shield against unauthorized access. Traditional methods of sharing images may expose them to potential threats, emphasizing the need for a secure encryption solution.

Blowfish, renowned for its strength and efficiency, emerges as a compelling choice for image encryption and decryption. The motivation behind opting for Blowfish is rooted in its proven track record in the realm of data security. Its adaptability, speed, and strong security features make it an ideal candidate for ensuring the confidentiality of shared images.

The motivation is twofold. Firstly, there's a growing awareness of the vulnerabilities associated with image sharing, ranging from personal privacy concerns to the protection of classified information in various sectors. Secondly, the need for a seamless and efficient process of encrypting and decrypting images is essential to strike a balance between security and user convenience.

By implementing Blowfish for image encryption and decryption, we aim to address these concerns head-on. The algorithm's ability to provide a robust defense against unauthorized access aligns with the contemporary need for secure image sharing. Simultaneously, its efficiency ensures that the encryption process doesn't hinder the user experience, making it a practical and user-friendly choice.

In summary, our motivation for implementing Blowfish in image sharing is firmly grounded in the quest for a seamless balance between user expectations and data security. By integrating Blowfish into our image sharing project, we aspire to overcome the shortcomings of conventional methods and elevate the efficiency, security, and satisfaction of users engaging in digital image transmission.

1.3 Proposed System

In the contemporary landscape of digital imagery, the extensive applications in fields like medical imaging, satellite imaging, and surveillance footage have become indispensable. However, the invaluable information encapsulated within these images renders them vulnerable during transmission over public networks. The potential threats of unauthorized access and modification loom large, posing significant challenges to the confidentiality and integrity of these images.

The issue at hand revolves around the secure sharing of these images, considering the sensitive nature of the content they may contain. The conventional transmission methods expose them to risks that demand a robust solution. Image encryption emerges as a pivotal approach to selectively share sensitive images over networks, safeguarding them from unauthorized tampering and access.

The image encryption process involves a complex interplay of encryption and decryption, utilizing a security algorithm and a secret key. This secret key, a unique identifier shared between the sender and receiver, acts as the linchpin in the authentication process. Access to the image is granted only when the secret key is validated by the encryption algorithm, ensuring a secure exchange of sensitive visual information.

While several encryption algorithms are available for image protection, such as DES, AES, RC6, RC2, among others, the focus of this project lies in the implementation of the Blowfish Algorithm. The choice of Blowfish is rooted in its proven track record as the fastest and most secure among its counterparts. This algorithm not only offers swiftness in processing but also provides an unparalleled level of security, making it an optimal solution for the challenges posed by the transmission of sensitive images.

In summary, the problem at hand revolves around the vulnerabilities inherent in transmitting digital images containing confidential information over public networks. The solution lies in implementing a robust image encryption mechanism, and for this project, the chosen path is the proven efficiency and security of the Blowfish Algorithm. The aim is to establish a framework that ensures the secure and seamless sharing of sensitive images, mitigating the risks associated with unauthorized access and modification in the digital realm.

1.4 Architecture Diagram:

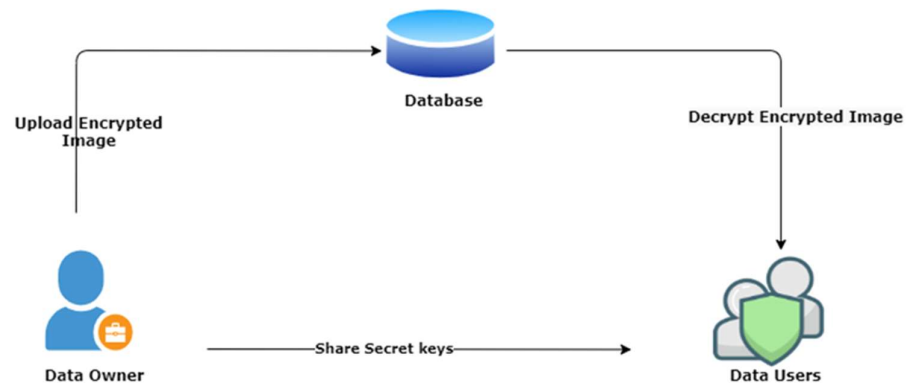


Fig 1.1 System architecture

Data Owner Registration:

In this module, the data owner will be registered with this application by filling out the registration form. Here, the registration form will be having the following fields such as name, user ID, password, email, and mobile number. The submitted registration form details will be stored in the MySQL database.

Data Owner Login:

In this module, the data owner will be authenticated with this application by entering valid login

credentials like user id and password. Thereafter submitting the login credentials will be verified with the database, so if the credentials are valid then the data owner will be redirected to the dashboard, otherwise denied to access their dashboard.

Image Encryption:

In this module, the data owner will upload the required image files into the database server by using blowfish cryptography technique. While image file uploading, the data owner will select an image file which will be encrypted with the Blowfish algorithm with the help of a random secure key. When the image file upload form is submitted then the encrypted image will be stored in the database.

Data User Registration Module:

In this module, the data user will be registered with this application by filling out the registration form. Here, the registration form will have the following fields such as name, user ID, password, email, and mobile number. The submitted registration form details will be stored in the MySQL database.

Data User Login Module:

In this module, the data user will be authenticated with this application by entering valid login credentials like user ID and password. Thereafter submitting the login credentials will be verified with the database, so if the credentials are valid then the data user will be redirected to the dashboard, otherwise denied to access their dashboard.

File Download:

In this module, the data user will be downloaded the required image files from the database server. When the data user clicks on the Image decryption hyperlink then this application will display a list of the uploaded file details. Thereafter, they can choose the required image file for downloading. While the image file downloading, the secure key will get from the database and thereafter the encrypted image will be decrypted with respective secure keys. Finally, the data user will download the image file from a database server.



2. LITERATURE REVIEW

2.1 Review of relevant Research Papers and Projects

1. Survey on Cryptography: A Comparative Analysis for Modern Techniques

Abstract:

Cryptography plays a vital role for ensuring secure communication between multiple entities. It has been identified that existing reviews are either focused only towards symmetric or asymmetric encryption types. Another limitation is found that a criterion for performance comparisons only covers common parameters. In this paper, the performance of different symmetric and asymmetric algorithms by covering multiple parameters such as encryption/decryption time, key generation time and file size have been evaluated. For evaluation purpose, simulations are performed in a sample context in which multiple cryptography algorithms have been compared.

Conclusion:

In this paper, the performance of symmetric and asymmetric cryptography schemes has been analyzed. Encryption time, decryption time and key generation time are used to evaluate the cryptographic schemes. The performance results show that the symmetric schemes are computationally inexpensive when compared with asymmetric schemes. The performance of these security algorithms is evaluated and experimented based on their performance on text file and image. The result showed that all algorithms are slow in performance as compared to Blowfish as it has an increased packet size.

2. A Modified AES Based Algorithm for Image Encryption

Abstract:

With the fast evolution of digital data exchange, security information becomes much important in data storage and transmission. Due to the increasing use of images in industrial process, it is essential to protect the confidential image data from unauthorized access. In this paper, we analyze the Advanced Encryption Standard (AES), and we add a key stream generator (A5/1, W7) to AES to ensure improving the encryption performance; mainly for images characterized by reduced entropy. The implementation of both techniques has been realized for experimental purposes. Detailed results in terms of security analysis and implementation are given. Comparative study with traditional encryption algorithms is shown the superiority of the modified algorithm.

Conclusion:

In this paper a new modified version of AES, to design a secure symmetric image encryption technique, has been proposed. The AES is extended to support a key stream generator for image encryption which can overcome the problem of textured zones existing in other known encryption algorithms. Detailed

analysis has shown that the new scheme offers high security, and can be realized easily in both hardware and software. The key stream generator has an important influence on the encryption performance. We have shown that W7 gives better encryption results in terms of security against statistical analysis attacks.

3. A review on Image Encryption and Decryption

Abstract:

Security information has become the major concern for the fast growth of the digital exchange of data storage and transmission. As there is rapid growth of using images in many fields, so it is important to protect the private image data from the intruders. Image protection has become an imperative issue. To protect an individual privacy has become a crucial task. Different methods have been explored and developed to preserve data and personal information. To protect the important information from unauthorized users, image encryption is used. Encryption is the one of the most used techniques to hidden the data from unauthorized users. The Advanced Encryption Standard (AES) is used for image encryption which uses the key stream generator to increase the performance of image encryption.

Conclusion:

The image is encrypted and decrypted using AES algorithm with 128 bits of key. The original image with key is given where it is converted into a blank form and send to the receiver where receiver will convert back into original image using key. It provides the security form inducer and widely used.

4. A Study of DES and Blowfish Encryption Algorithm

Abstract:

We overviewed the base functions and analyzed the security for both algorithms. We also evaluated performance in execution speed based on different memory sizes and compared them. The experimental results show the relationship between function run speed and memory size. Symmetric keys encryption only uses one key to encrypt and decrypt data. The key should be distributed before transmission between entities. They are both symmetric key encryption algorithms using block cipher. Referencing their encryption process methods, their security is analyzed and experiments are done to evaluate performances of the two encryption algorithms using different memory sizes. From the experimental results, a relationship is found between encryption speed and computer memory utilization. Their advantages and disadvantages are also mentioned.

Conclusion:

In this paper, the two popular encryption algorithms: DES and Blowfish are studied. AN overview of the basic flow of the two algorithms and their security is analyzed. Both algorithms have high security to resist differential cryptanalysis and linear cryptanalysis attacks. It is evaluated that encryption function speed is based on different memory sizes. The experimental results show Blowfish is much faster than

DES but the increase in speed for Blowfish is slower compared to DES because it needs much more memory for sub-key and S-boxes initialization.

5. Cryptography in Image Using Blowfish Algorithm

Abstract:

This paper about encryption and decryption of images uses Blowfish which is an evolutionary improvement over DES, 3DES, etc. designed to increase security and improve performance. It employs Feistel network which iterates a simple function 16 times and uses a combination of 4 S-boxes lookup. It helps in obtaining results by simulating the image processing part in MATLAB and encryption and decryption part in VHDL for better security. In this paper securing the image is executed with a “Blowfish algorithm” from the perspective of cryptology. Blowfish is used for the applications, where the key doesn’t change often and has a larger space to store the data. Encryption and decryption are performed to obtain the original hiding information from the image

Conclusion:

In this paper to transmit encrypting image over the internet we have used the blowfish algorithm. Previously used algorithm like AES, DES and so more has been replaced by the blowfish algorithm, because of producing successful effectiveness on security. Blowfish algorithm can’t be easily broken by the hackers until they find the correct combinations. This is more difficult to form the exact combinations of the lock. To make the algorithm stronger number of rounds has been increased. It takes less time to encrypt and decrypt the image than any other algorithms. For future enhancement advanced algorithms can be invent for better security and helps to encrypt more complicated image.

6. Image Encryption and Decryption using Blowfish & Watermarking

Abstract:

This paper is about encryption and decryption of images using a secret-key block cipher called 64-bits Blowfish designed to increase security and to improve performance. This algorithm will be used as a variable key size up to 448 bits. It employs Feistel network which iterates simple function 16 times. The blowfish algorithm is safe against unauthorized attack and runs faster than the popular existing algorithms. The proposed algorithm is designed and realized using MATLAB.

Conclusion:

Both colour and black & white image of any size saved in tagged image file format (TIF), Bit map (bmp), Portable network graphics (PNG), Joint Photographic Experts group (jpg), etc. can be encrypted & decrypted using blowfish algorithm.

2.2 Relevant Technologies and Tools

The project on “Image Encryption using Blowfish Algorithm” requires a range of technologies and tools to achieve its objectives.

Here’s an overview of the technologies and tools involved:

- a) **Programming Language:** Choose a programming language that supports the Blowfish algorithm and allows for image manipulation. Common choices include Java, C/C++, Python, and others.
- b) **Blowfish Algorithm Implementation:** You will need a library or code that implements the Blowfish algorithm. Depending on your chosen programming language, you can find or develop a suitable implementation.
- c) **Image Processing Libraries:** To work with image data, you may need image processing libraries or tools. For example, Java has the Java Advanced Imaging (JAI) library, while Python has the Python Imaging Library (PIL) or Pillow for image handling.
- d) **Key Management Tools:** Secure key management is essential for image encryption. You might use tools and techniques for generating, storing, and protecting encryption keys, such as secure key storage systems or hardware security modules (HSMs).
- e) **Data Preprocessing Tools:** To prepare the image data for encryption, you may need tools for format conversion, resizing, padding, and other preprocessing operations to ensure the data is compatible with the Blowfish algorithm.
- f) **Cryptography Libraries:** Depending on your chosen programming language, you may need cryptography libraries to assist with key derivation, secure random number generation, and other cryptographic tasks.
- g) **Random Number Generators:** Secure and random initialization vectors (IVs) are important for encryption. Make use of cryptographic random number generators to create IVs.
- h) **Development Environment:** Use integrated development environments (IDEs) or code editors suitable for your chosen programming language to write, test, and debug your encryption code.
- i) **Data Storage:** For saving encrypted images or storing encryption keys, consider secure data storage options, such as databases, file systems, or cloud storage services.

- j) **Security Best Practices:** Adhere to security best practices for both code development and deployment. This includes secure coding standards, secure key management, and adherence to data protection regulations.
- k) **Testing Tools:** Utilize testing tools and frameworks for verifying the correctness and security of your image encryption code. This might include unit testing and security testing tools.
- l) **Documentation Tools:** Proper documentation of your image encryption process is important for maintaining and sharing the code, as well as for security audits. Use documentation tools and standards to create clear and comprehensive documentation.
- m) **Version Control System:** Use version control systems like Git to manage and track changes in your code, which helps with collaboration and maintaining a history of your work.
- n) **Operating System:** Ensure that your chosen operating system supports the programming language, libraries, and tools you plan to use for image encryption.
- o) **Security Compliance and Regulations:** Depending on the application and industry, you may need to adhere to specific security compliance standards and regulations, such as HIPAA, GDPR, or FIPS.
- p) **User Interface (if applicable):** If your image encryption system includes a user interface, consider the tools and technologies needed for designing and developing the interface, such as graphical user interface (GUI) libraries or web development frameworks.

The software technologies for our project are as follows :

- a) The project is developed in Java Programming Language by using the Eclipse Juno Integrated Development Environment (IDE). We use the Java Development Kit (JDK) which includes a variety of custom tools that help us to develop web applications on the Apache Platform.
- b) At the Server-side Apache Tomcat Server is used to deploy the program.
- c) For Data Storage Purpose we use MySQL as a database server. And We can run our application in Windows and Linux any version.



3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Introduction to SRS

The project is developed in Java Programming Language by using the Eclipse Juno Integrated Development Environment (IDE). We use the Java Development Kit (JDK) which includes a variety of custom tools that help us to develop web applications on the Apache Platform. At the Server-side Apache Tomcat Server is used. For Data Storage Purpose we use MySQL as a database server. And We can run our application in Windows and Linux any version.

The purpose of this document is to present a detailed description of the “encryption and decryption of images using a secret key of Blowfish designed to increase security and to improve performance” application. It will explain the purpose and features of the system that it will provide, the constraints under which it must operate, and how the system will react. The document also describes the nonfunctional requirements of the system.

3.2 Functional Requirements (Modules)

Data Owner Registration:

In this module, the data owner will be registered with this application by filling out the registration form. Here, the registration form will be having the following fields such as name, user ID, password, email, and mobile number. The submitted registration form details will be stored in the MySQL database.

Data Owner Login:

In this module, the data owner will be authenticated with this application by entering valid login credentials like user id and password. Thereafter submitting the login credentials will be verified with the database, so if the credentials are valid then the data owner will be redirected to the dashboard, otherwise denied to access their dashboard.

Image Encryption:

In this module, the data owner will upload the required image files into the database server by using blowfish cryptography technique. While image file uploading, the data owner will select an image file which will be encrypted with the Blowfish algorithm with the help of a random secure key. When the image file upload form is submitted then the encrypted image will be stored in the database.

Data User Registration Module:

In this module, the data user will be registered with this application by filling out the registration form. Here, the registration form will have the following fields such as name, user ID, password, email, and mobile number. The submitted registration form details will be stored in the MySQL database.

Data User Login Module:

In this module, the data user will be authenticated with this application by entering valid login credentials like user ID and password. Thereafter submitting the login credentials will be verified with the database, so if the credentials are valid then the data user will be redirected to the dashboard, otherwise denied to access their dashboard.

File Download:

In this module, the data user will be downloaded the required image files from the database server. When the data user clicks on the Image decryption hyperlink then this application will display a list of the uploaded file details. Thereafter, they can choose the required image file for downloading. While the image file downloading, the secure key will get from the database and thereafter the encrypted image will be decrypted with respective secure keys. Finally, the data user will download the image file from a database server.

3.3 Non-Functional Requirements**1. Flexibility & Scalability**

Oracle itself has given a set of applications with JDK but the whole developer community can develop their own applications and they have access to the same resources and public API which are accessible to core applications.

2. Robust

The application is fault tolerant with respect to illegal user/receiver inputs. Error checking has been built in the system to prevent system failure.

3. Fragmentation

Java gave the same environment which is open; the entire API's which is open to all the devices which reduces fragmentation. If you develop java application, it will run on all the devices.

4. Open Source:

Java open source is free and easy to download. Java is a platform independent based programming language and The Java virtual machine (JVM) is a software implementation of a computer that executes programs like a real machine.

5. Scalability:

The system can be extended to integrate the modifications done in the present application to improve the quality of the product. This is meant for the future works that is to be done on the application.

6. Reliability:

Since the application is being developed through java, the most famous, efficient and reliable language, so it is reliable in every aspect until and unless there is an error in the programming side. Thus, the

application can be a compatible and reliable one.

7. Portability:

This System must be intuitive enough such that user with average background in using mobile phones can quickly experiment with the system and learn how to use the project. The system has user friendly interface.

8. Feasibility study

A key part of the preliminary investigation that reviews anticipated costs and benefits and recommends a course of action based on operational, technical, economic, and time factors. The purpose of the study is to determine if the systems request should proceed further.

9. Organisational Feasibility

The application would contribute to the overall objectives of the organization. It would provide a quick, error free and cost-effective solution to the current process CRM marketing. It would provide a solution to many issues in the current system. As the new system is flexible and scalable it can also be upgraded and extended to meet other complex requirements which may be raised in the future. However, it is up to the organization to upgrade or extend it.

10. Economic Feasibility

The project is economically feasible as it only requires a system with Windows or Linux operating system. The application is free to get once released into real time host server. The users should be able to connect to internet through machine and this would be the only cost incurred on the project.

11. Technical Feasibility

To develop this application, a high-speed internet connection, a database server, a web server and software are required. The current project is technically feasible as the application was successfully deployed on Lenovo ThinkPad L60 having Windows 7 operating system and also Intel I5 Processor.

12. Behavioural Feasibility

The application is behaviourally feasible since it requires no technical guidance, all the modules are user friendly and execute in a manner they were designed to.

3.4 Performance Requirements

Performance requirements are crucial for ensuring that your image encryption web app runs smoothly and efficiently. Here are some performance considerations:

1. Encryption and Decryption Speed:

Define acceptable time limits for encrypting and decrypting images. The process should be fast enough to provide a seamless user experience. Optimize the encryption algorithms and implementation to achieve the best possible performance without compromising security.

2. User Authentication Response Time:

Set response time expectations for the user authentication system. Users should be able to log in and access their encrypted images quickly and without delays. Implement caching mechanisms and efficient database queries to speed up authentication processes.

3. File Upload and Download Speed:

Specify the maximum acceptable time for uploading and downloading encrypted images. This is particularly important for users with slower internet connections. Optimize file handling processes and consider implementing parallel processing for multiple file uploads or downloads.

4. Mobile App Responsiveness:

Ensure that the mobile version of your app is responsive and provides a smooth user experience on different devices and network conditions. Optimize mobile app performance by minimizing unnecessary data transfers and using efficient algorithms.

5. Decentralized Storage Access Time:

If integrating decentralized storage, evaluate and set expectations for the time it takes to retrieve and store images from decentralized networks. Optimize the integration with decentralized storage solutions to minimize latency.

6. Scalability:

Plan for scalability to accommodate an increasing number of users and growing data storage requirements. Implement load balancing and horizontal scaling strategies to distribute the workload across multiple servers.

7. Resource Utilization:

Monitor and optimize server resource usage, including CPU, memory, and disk space, to ensure efficient operation and minimize server downtime. Implement caching mechanisms to reduce redundant computations and database queries.

8. Security Overhead:

While focusing on performance, ensure that security measures do not introduce excessive overhead. Strive for a balance between robust security and efficient performance.

9. Error Handling and Recovery:

Implement effective error handling mechanisms to promptly identify and address issues, minimizing downtime and ensuring a smooth user experience. Define recovery strategies in case of system failures or unexpected errors.

10. User Interface Responsiveness:

Optimize the user interface to provide a responsive and engaging experience. Consider lazy loading for images and other elements to improve page load times.

Regularly monitor and analyze the performance of your image encryption web app, making adjustments and optimizations as needed. Keep user satisfaction in mind while balancing security and efficiency.

3.5 Software Requirements

The software interface is the operating system, and application programming interface used for the development of the software.

Operating System: Windows XP or higher / Linux

Platform: JDK

Application Server: Apache Tomcat 7 or higher

Database: MySQL

Technologies used: Java, J2EE.

3.6 Hardware Requirements

CLIENT				
OPERATING SYSTEM	SOFTWARE	PROCESSOR	RAM	Hard disk
Windows/Linux	Any Advanced Browser. Chrome/Opera	Intel/AMD processor	256 Mb	160 GB

Table 3.6.1 Client Requirements

SERVER				
OPERATING SYSTEM	SOFTWARE	PROCESSOR	RAM	HARD DISK
Windows/Linux	Any Browser. JDK 1.7 or above Apache 7 or above MySQL 5.0 or above	Intel/AMD processor	256Mb	160 GB

Table 3.6.2 Server Requirements



4. SYSTEM DESIGN

4.1 Introduction to UML

Requirements gathering followed by careful analysis leads to a systematic Object-Oriented Design (OOAD). Various activities have been identified and are represented using Unified Modeling Language (UML) diagrams.

4.2 UML Diagrams

UML diagrams are used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development

4.2.1 Use Case Diagram

In the Unified Modeling Language (UML), the use case diagram is a type of behavioral diagram defined by and created from a use-case analysis. It represents a graphical over view of the functionality of the system in terms of actors, which are persons, organizations or external system that plays a role in one or more interaction with the system. These are drawn as stick figures. The goals of these actors are represented as use cases, which describe a sequence of actions that provide something of measurable value to an actor and any dependencies between those use cases.

Our project has 5 use cases as shown namely, registering, login credentials, image encryption, image decryption and logging out.

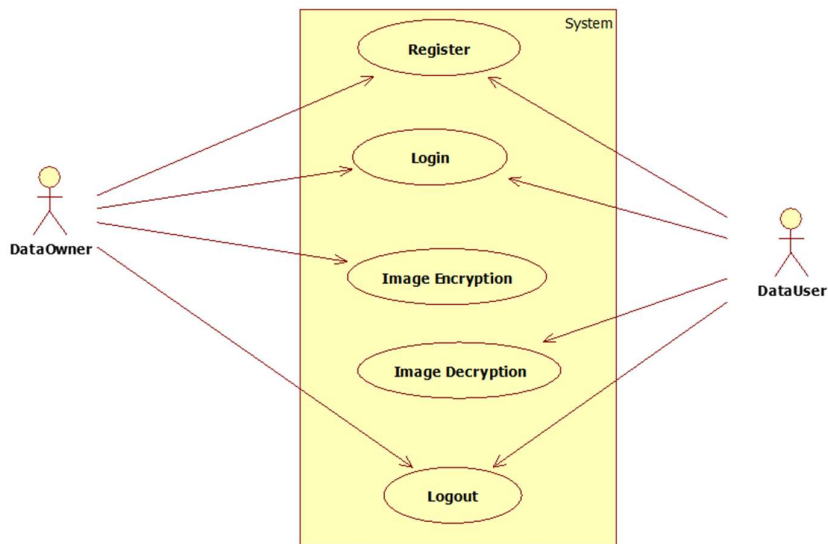


Fig 4.2.1 Use Case Diagram for System

4.2.2 Sequence Diagram

Sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between a number of lifelines.

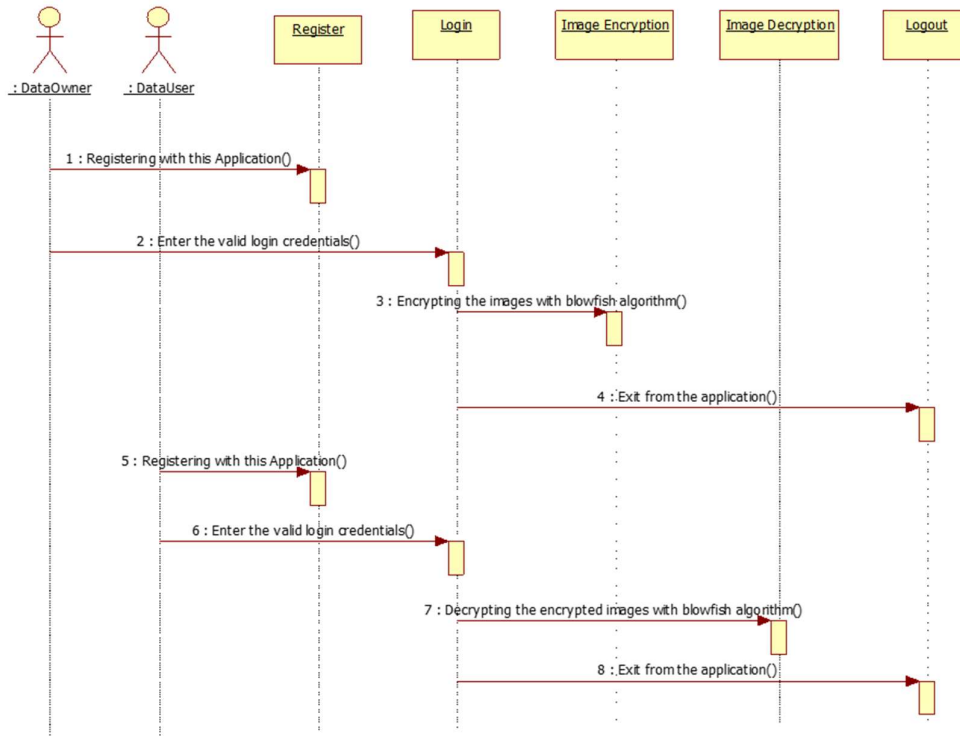


Fig 4.2.2 Sequence Diagram

4.2.3 State Chart Diagram

A state chart, also known as a state machine diagram, is a type of Unified Modeling Language (UML) diagram that represents the dynamic behaviour of a system. It visually depicts the different states that an object or a system can exist in and the transitions between these states in response to events. State charts are particularly useful for modeling the behaviour of entities that undergo discrete changes in their state based on external stimuli.

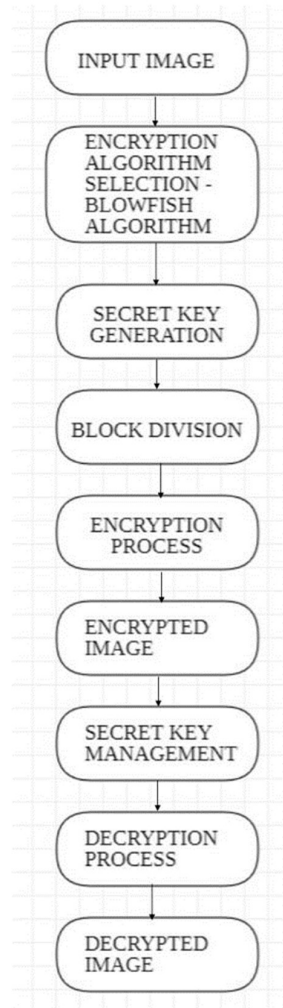


Fig4.2.3 State Chart Diagram

4.2.4 Deployment Diagram

Deployment diagram shows execution architecture of systems that represent the assignment (deployment) of software artifacts to deployment targets (usually nodes). Nodes represent either hardware devices or software execution environments. Artifacts represent concrete elements in the physical world that are the result of a development process and are deployed on nodes.

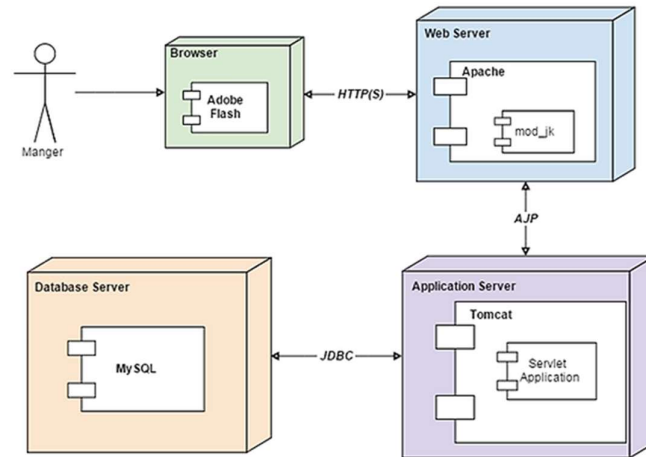


Fig 4.2.4 Deployment Diagram of the System

4.3 Technologies used

Java platform

The programmer writes Java source code in a text editor which supports plain text. Normally the programmer uses an *Integrated Development Environment* (IDE) for programming. An IDE supports the programmer in the task of writing code, e.g., it provides auto-formatting of the source code, highlighting of the important keywords, etc.

At some point the programmer (or the IDE) calls the Java compiler (javac). The Java compiler creates the *byte code* instructions. These instructions are stored in .class files and can be executed by the Java Virtual Machine.

J2SE

J2SE is a collection of Java Programming Language API (Application programming interface) that is very useful to many Java platform programs. It is derived from one of the most dynamic programming languages known as "JAVA".

J2SE is a collection of Java Programming Language API (Application programming interface) that is very useful to many Java platform programs. It is derived from one of the most dynamic programming languages known as "JAVA" and one of its three basic editions of Java known as Java standard edition being used for writing Applets and other web-based applications.

The J2SE platform has been developed under the Java umbrella and is primarily used for writing applets and other Java-based applications. It is mostly used for individual computers. Applet is a type of fast-working subroutine of Java that is platform-independent but works within other frameworks. It is a mini-application that performs a variety of functions, large and small, ordinary and dynamic, within the framework of larger applications.

J2SE provides the facility to users to see Flash movies or hear audio files by clicking on a Web page link. As the user clicks, the page goes into the browser environment and begins the process of launching the application within an application to play the requested video or sound application. So many online games are being developed on J2SE. JavaBeans can also be developed by using J2SE.

J2EE

Java 2 Platform *Enterprise Edition*. J2EE is a platform-independent, Java-centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multitier, Web-based applications.

Key features and services of J2EE:

- At the client tier, J2EE supports pure HTML, as well as Java applets or applications. It relies on Java Server Pages and servlet code to create HTML or other formatted data for the client.
- Enterprise JavaBeans (EJBs) provide another layer where the platform's logic is stored. An EJB server provides functions such as threading, concurrency, security and memory management. These services are transparent to the author.
- Java Database Connectivity (JDBC), which is the Java equivalent to ODBC, is the standard interface for Java databases.
- The Java servlet API enhances consistency for developers without requiring a graphical user interface.

JDBC

JDBC stands for **J**ava **D**atabase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks commonly associated with database usage:

- Making a connection to a database
- Creating SQL or MySQL statements
- Executing that SQL or MySQL queries in the database
- Viewing & modifying the resulting records

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as:

- Java Applications
- Java Applets
- Java Servlets
- Java Server Pages (JSPs)
- Enterprise JavaBeans (EJBs)

All of these different executables are able to use a JDBC driver to access a database and take advantage of the stored data.

JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

JDBC Architecture:

The JDBC API supports both two-tier and three-tier processing models for database access but in general JDBC Architecture consists of two layers:

- **JDBC API:** This provides the application-to-JDBC Manager connection.
- **JDBC Driver API:** This supports the JDBC Manager-to-Driver Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases. The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.

Following is the architectural diagram, which shows the location of the driver manager with respect to the JDBC drivers and the Java application.

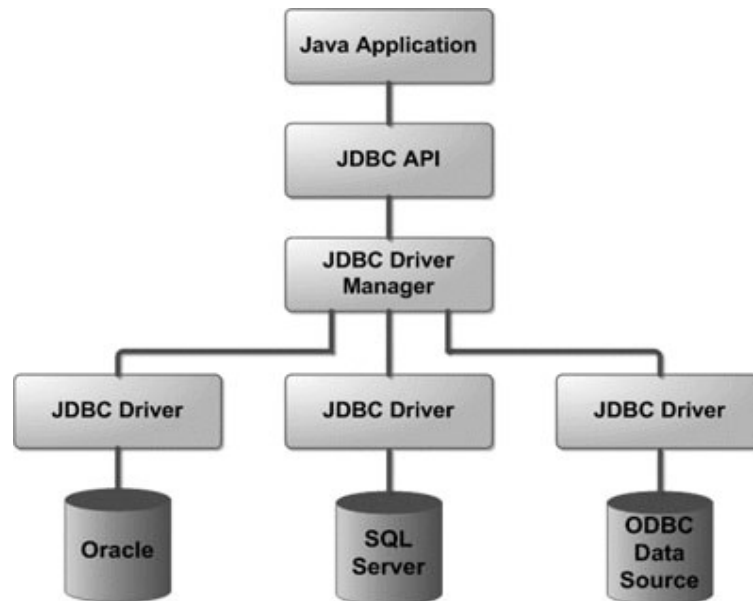


Fig 4.3.1 JDBC Architecture



5. IMPLEMENTATION

5.1 Working Principle

5.1.1 Cryptography

Cryptography plays a vital role for ensuring secure communication between multiple entities. In many contemporary studies, researchers contributed towards identifying best cryptography mechanisms in terms of their performance results. There are many encryption algorithms are developed and widely used for information security. They can be categorized into symmetric (private) and asymmetric (public) keys encryption.

Symmetric key encryption only uses one key to encrypt and decrypt data. The key should be distributed before transmission between entities. Keys play a very important role because if weak key is used in algorithm, then anyone may decrypt the data. Strength of Symmetric key encryption depends on the size of used key. For the same algorithm, encryption using longer key is harder to break than the one done using smaller key. There are many examples of strong and weak keys of cryptography algorithms like RC2, DES, 3DES, RC6, Blowfish, and AES.

Public key is known to the public while private key is known only to the user. There is no need for distributing them prior to transmission. However, public key encryption is based on mathematical functions, computationally intensive and is not very efficient for small mobile devices such as cell phone, PDA, and so on.

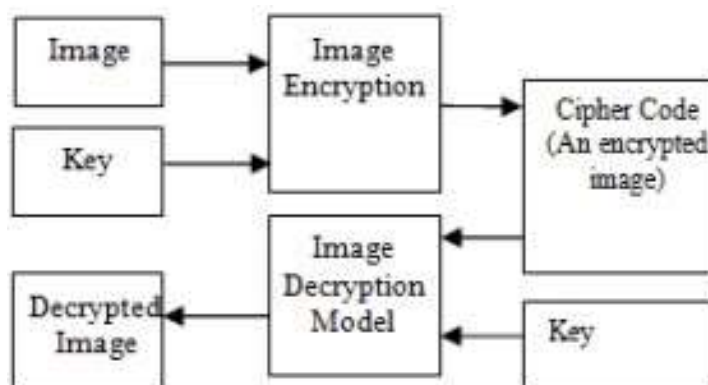


Fig 5.1.1(a) Architecture of Cryptography

When someone wants to send a message to a receiver and wants that message to be confidential, such that no other persons can read the text or the message. There is a possibility of hacking the messages that send through internet and also by other means. To avoid this, encryption and decryption is applied

so that the message will secure, meaning that the third party can't hack or read the message. This is done using encryption. Decryption is the transitive process of encryption.

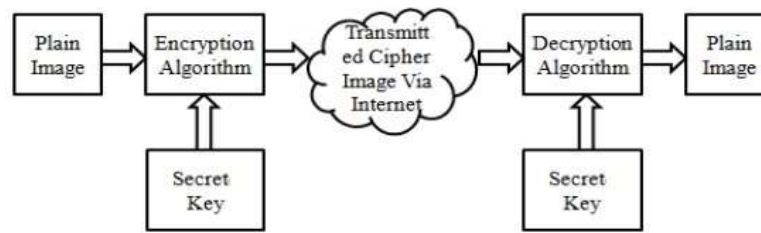


Fig 5.1.1(b) Encryption & Decryption Process

5.1.2 Blowfish Algorithm

In 1993, Bruce Schneier published the Blowfish block cipher. At this time, the current Data Encryption Standard (DES) was known to be vulnerable to crypto analysis and brute-force attacks. Schneier developed Blowfish to be a publicly available cryptographic algorithm with the potential to replace DES. The Blowfish algorithm has many advantages. It is suitable and efficient for hardware implementation and no license is required. The elementary operators of Blowfish algorithm include table lookup, addition and XOR. The table includes four S-boxes and a P-array. Schneier also encouraged others to evaluate the performance and security of Blowfish. To date, the security of Blowfish has not been compromised.

Blowfish Algorithm is a Feistel Network, iterating a simple encryption function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits. Although there is a complex initialization phase required before any encryption can take place, the actual encryption of data is very efficient on large microprocessors. Blowfish is a symmetric block cipher that encrypts data in 8- byte (64-bit) blocks. The following diagram shows a simplified overview of the Blowfish algorithm:

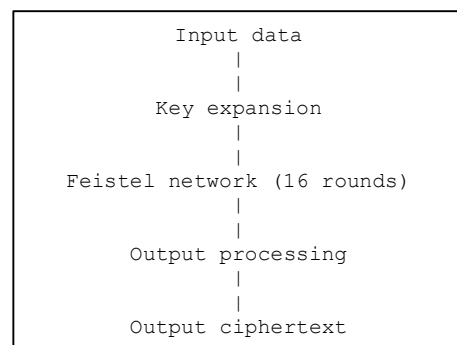


Fig 5.1.2(a) Flow of Blowfish

The algorithm has two parts , key expansion and data
 encryption. Key expansion consists of generating the initial contents of one array (the P-array),
 namely, eighteen 32-bit sub keys, and four arrays (the S boxes), each of size 256 by 32 bits, from a key

of at most 448 bits (56 bytes). All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

Blowfish also incorporated two exclusive-or operations to be performed after the 16 rounds and a swap operation. It can have a key that ranges from 32 to 448-bits. It is suitable and efficient for hardware implementation and no license is required. It has been suggested as a replacement for DES.

It is suitable for applications where the key does not change often, like a communications link or an automatic file encryptor. It is significantly faster than most encryption algorithms when implemented on 32-bit microprocessors with large data caches. Image encryption techniques try to convert an image to another one that is hard to understand. On the other hand, image decryption retrieves the original image from the encrypted one.

1. Key generation:

Blowfish requires a binary key of a specific length. To convert the password into a binary key, a password-based key derivation function (PBKDF) is often used. A common PBKDF used with Blowfish is bcrypt.

The PBKDF takes the password and a randomly generated salt as inputs and produces a binary key of the required length. Once you have a binary key, it is used to initialize the Blowfish algorithm. Blowfish has a fixed key length of 448 bits (56 bytes), so if the derived key is shorter, it should be expanded to meet this length.

2. Key expansion:

The key expansion algorithm in Blowfish is responsible for converting the encryption key into a set of subkeys that are used by the encryption and decryption algorithms. The key expansion algorithm is as follows:

1. Initialize the p-array with a fixed string of hexadecimal digits of pi.
2. Initialize the s-boxes with a fixed string of hexadecimal digits.
3. For each subkey in the P-array:
 - XOR the subkey with the next 32 bits of the encryption key.
 - Encrypt the XORed subkey using the Blowfish encryption algorithm.
 - Update the subkey with the output of the encryption operation.
4. Repeat steps 3 and 4 until all of the subkeys in the P-array have been updated.

5. Update the S-boxes using the P-array and the encryption algorithm.

Key expansion converts a key of at most 448 bits into several sub-key arrays totalling 4168 bytes. Data encryption occurs via a 16-round (commonly) network. Each round consists of a key-dependent permutation, and a key- and data-dependent substitution.

Blowfish uses a fixed set of initial P-Array (P[0] to P[17]) and S-Boxes (S0, S1, S2, S3). These tables are initialized with the hexadecimal digits of pi (π) for strong diffusion.

The p-array consists of 18, 32-bit sub keys:

P1, P2 ,....., P18

S-Boxes consist of 256 entries each,

S1,0, S1, 1, S1, 255

S2,0, S2,1, S2, 255

S3,0, S3, 1, S3, 255

S4,0, S4, 1, S4, 255

P-array:

The P-array is initialized with a fixed string of hexadecimal digits of pi, but it is also updated during the key expansion algorithm.

It is used in the following ways:

- Encryption: The P-array is used to generate the subkeys that are used by the F-function in the encryption algorithm.
- Decryption: The P-array is used to generate the subkeys that are used by the inverse F-function in the decryption algorithm.

The P-array is key-dependent, which means that it changes with the encryption key. This helps to make the Blowfish algorithm more secure. The use of subkeys helps to make the Blowfish algorithm more resistant to differential and linear cryptanalysis.

S-box:

S-boxes are used to add non-linearity to the encryption process, making it more difficult for attackers to break the cipher.

In Blowfish, there are four S-boxes, each of which is a 32x32 table that maps 8-bit input values to 32-bit output values. The S-boxes are initialized with a fixed set of hexadecimal values, but they are also updated during the key expansion algorithm.

The use of S-boxes in the F-function makes the Blowfish algorithm more resistant to differential and linear cryptanalysis. Differential cryptanalysis is a type of attack that exploits patterns in the differences between plaintext and ciphertext to break a cipher. Linear cryptanalysis is a type of attack that exploits patterns in the linear combinations of plaintext, ciphertext, and key bits to break a cipher.

3. Block Division

Dividing the image data into fixed-size blocks, typically 64 bits (8 bytes) each. Blowfish operates on blocks of data.

4. Feistel Network

A Feistel cipher is a type of symmetric block cipher that encrypts data by dividing it into two halves and performing a series of operations on each half in alternating order. The Feistel cipher structure was invented by Horst Feistel in the 1970s and is used in many popular block ciphers, including the Blowfish encryption algorithm. In this network model the plaintext is XORed with the key and divided into blocks. These blocks are then sent to the s-boxes wherein each output from the s-box is sent into the permutation layer i.e., the permutations are changed and then sent to the next round. The Feistel network consists of N rounds, where N is typically 16 or 32.

Here is a diagram of the Feistel cipher structure :

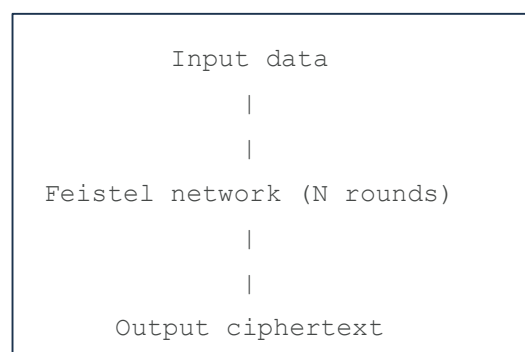


Fig 5.1.2(b) Feistel cipher structure

Each round of the Feistel network consists of the following steps:

1. The input data is divided into two halves, left and right.
2. The right half of the data is passed through a round function, which is a non-linear function that takes the right half of the data and a subkey as input and produces a new output value.

3. The output of the round function is XORed with the left half of the data.
4. The left and right halves of the data are swapped.
5. Steps 2-4 are repeated for N rounds.
6. After the final round, the left and right halves of the data are combined to produce the ciphertext.

The round function is the most important part of the Feistel cipher structure. It is responsible for adding non-linearity and diffusion to the encryption process. The round function typically consists of a combination of substitutions, permutations, and XORs. In the Blowfish encryption algorithm, the round function is called the F-function.

The s-boxes and p-arrays provide more functionality based on properties like confusion and diffusion. Confusion refers to making the relationship between the keys and the cipher text as complex and as involved as possible. Diffusion refers to the property that makes the ‘redundancy’ in the statistics of the plain text as ‘dissipated’ in the statistics of the cipher text. Transposition is a technique of diffusion.

F-function:

The f-function, regarded as the primary source of algorithm security, combines two simple functions: addition modulo two (XOR) and addition modulo 232.

The f-function is the distinguishing feature of blowfish and is applied as follows:

Divide XL (32 bits) into four 8-bit quarters: a, b, c and d. Then:

$$F(XL) = \{(S1[a] + S2[b]) \oplus S3[c] + S[d]\},$$

Where + means addition modulo 232, and \oplus means exclusive OR i.e., XOR S1[a], meaning the content of S-box 1 is at address a. The addresses a, b, c and d are 8-bits wide while the S-box outputs are 32-bits wide.

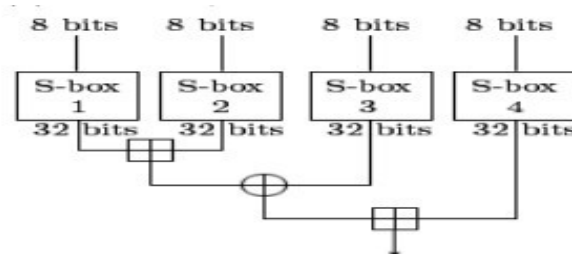


Fig 5.1.2(c) Feistel cipher function (f-function)

The algorithms based on block ciphers as well as stream ciphers use the following design principles :-

- a) **Block Size:** Increasing Size Improves Security, but slows cipher.
- b) **key size:** Increasing size improves security, makes key searching exhaustive but may slow down the

cipher.

c) **No of rounds:** Increasing number of rounds improves security.

d) **Sub Key generation:** greater complexity can make analysis harder.

e) **Round Function:** Greater complexity-based algorithms making breaking into the system harder.

5. Encryption

For encrypting an image, the image is taken as a plaintext. This plaintext and the encryption key are the two inputs of the encryption process. The original image data bit stream is divided into the blocks length of the Blowfish algorithm.

Image header is excluded to encrypt and the start of the bitmap pixel or array begins right after the header of the file. The byte elements of the array are stored in row order from left to right with each row representing one scan line of the image and the rows of the image are encrypted from top to bottom.

During encryption, the plaintext is typically represented as a byte array. The encryption algorithm takes this byte array, along with a cryptographic key, and transforms it into ciphertext. The ciphertext is also represented as a byte array, but it's a different byte array from the original plaintext. It's the byte array that contains the encrypted data.

In our project, the image data shown on the screen is a byte array which gets converted into ciphertext upon encrypting.

The following is the Feistel cipher structure used in Blowfish algorithm:

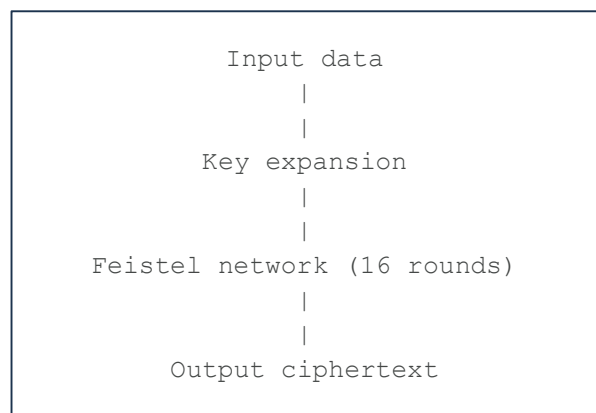


Fig 5.1.2(d) Feistel structure used in Blowfish algorithm

The blowfish algorithm principles are :-

blockSize : 64-bits

keySize: 32-bits to 448-bits variable size

number of subkeys: 18 [P-array]

number of rounds: 16

number of substitution boxes: 4 [each having 512 entries of 32-bits each]

There are 16 rounds in blowfish, here the input is a 64-bit data element, X. X is divided into two 32-bit halves: xL, xR [8]. Then, for i = 1 to 16, a series of rounds of operations are applied to each half. Each round consists of the following steps:

- The left half of the data is XORed with a subkey.

$$xL = xL \oplus P_i$$

- The right half of the data is passed through the F-function.

$$xR = F(xL) \oplus xR$$

- The left and right halves of the data are swapped.

After the sixteenth round, xL and xR are has to swap again to undo the last swap. Then,

$$xR = xR \oplus P_{17},$$

$$xL = xL \oplus P_{18}. \text{ (Here } \oplus \text{ means XOR)}$$

6. Output processing

The left and right halves of the data are XORed with subkeys, and then recombined to produce the output ciphertext. That is, recombining xL and xR to get the cipher text. Output X (64-bit data block: cipher text)

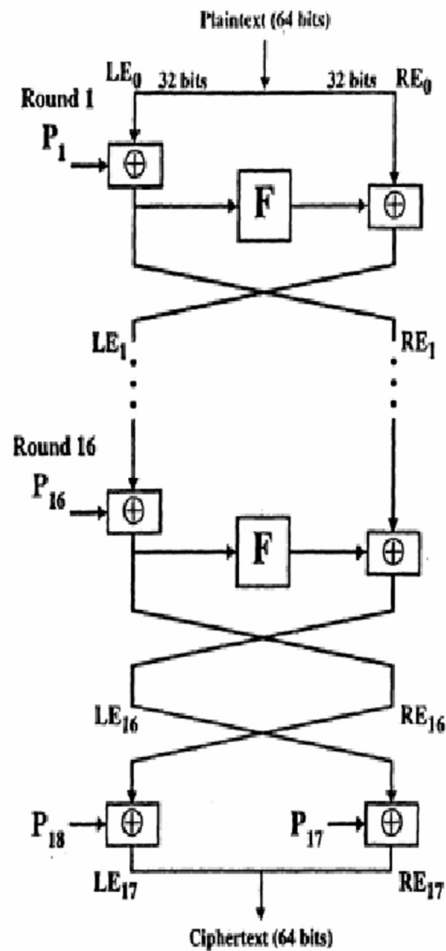


Fig 5.1.2(e) Encryption using Blowfish algorithm

7. Decryption

The first block is entered into the decryption function and the same encryption key is used to decrypt the image but the application of sub keys is reversed. The process of decryption is continued with other

blocks of the image from top to bottom. The nature of the Feistel network ensures that every half is swapped for the next round (except, here, for the last two sub-keys P17 and P18)

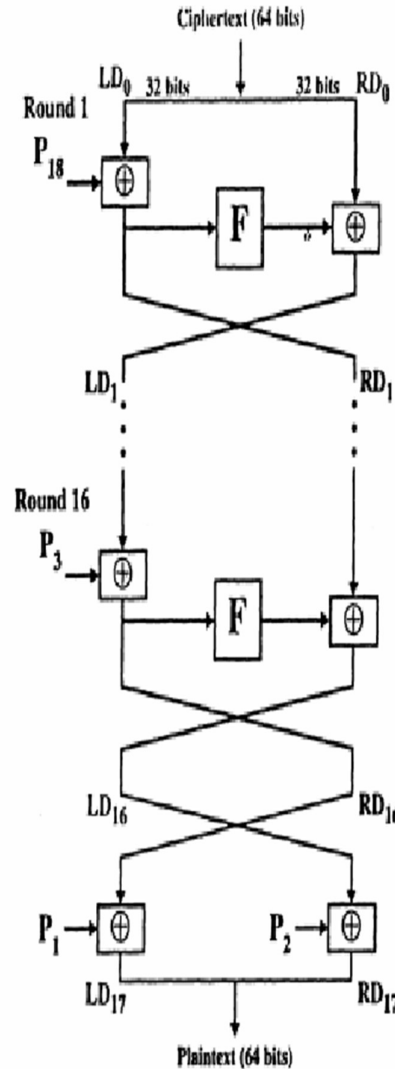


Fig 5.1.2(f) Decryption using Blowfish algorithm

5.2 Description of the approach

For our project, we have integrated this algorithm into an interactive frontend that allows the sender and receiver to share sensitive images with ease. The data owner can send images securely to the intended

receivers without as privacy breach. Designing and analysing an image encryption system using the Blowfish algorithm requires careful planning and evaluation to ensure the security and effectiveness of the process.

1. System Architecture:

Defining the architecture of the image encryption system. This includes the components responsible for encryption, decryption, key management, and any necessary interfaces with external systems or applications.

2. Key Management:

Implementing a robust key management system. Defining procedures for key generation, storage, distribution, and revocation. Considering using a secure key storage solution or hardware security modules for enhanced key protection.

3. Image Preprocessing:

Determining how the input images will be preprocessed. This includes resizing, format conversion, or data extraction to ensure compatibility with the Blowfish algorithm.

4. Encryption Process:

Implementing the Blowfish encryption algorithm, including key expansion, block division, and the Feistel network rounds and ensuring that the encryption process is correctly integrated into the system architecture.

5. Ciphertext Handling:

Specifying how encrypted images will be stored, transmitted, and accessed. Making sure that the ciphertext maintains the same dimensions as the original image, allowing for proper decryption.

6. Decryption Process:

Blowfish decryption algorithm should be capable of correctly reversing the encryption process using the same secret key.

7. Performance Optimization:

Fine-tuning the system for efficiency and performance by using method like parallel processing, caching, or hardware acceleration to speed up encryption and decryption.

8. Error Handling:

Developing error-handling mechanisms to deal with issues such as incorrect keys, corrupted ciphertext, or unexpected system failures. Proper error handling helps maintain system integrity.

9. User Authentication and Access Control:

Implementing user authentication mechanisms to ensure that only authorized users can access and decrypt images. Access control policies should be defined to protect sensitive image data.

5.3 Data Analysis

Analysing the data before using it effectively helps remove any anomalies and maintains data integrity.

1. Security Analysis:

Conducting a thorough security analysis to identify potential vulnerabilities and threats to the encryption system. Evaluating the strength of the Blowfish algorithm against known attacks and assessing the robustness of key management.

2. Cryptographic Evaluation:

Analysing the cryptographic properties of the system, including the avalanche effect, diffusion, and resistance to various cryptographic attacks and verifying that the encryption process provides a high level of security.

3. Performance Evaluation:

Measuring the system's performance, including encryption and decryption speed, memory usage, and CPU utilization, also identifying any performance bottlenecks and assessing the system's efficiency.

4. Error Rate Analysis:

Calculating the error rate by comparing the original image with the decrypted image. A low error rate indicates the quality of the encryption process.

5. Key Management Assessment:

Evaluating the key management system's effectiveness and security. Assessing key distribution, storage, and revocation procedures to ensure they meet security standards.

6. Usability Testing:

Conduct usability testing to ensure that the system is user-friendly and intuitive. Gather feedback from users to make any necessary improvements to the interface and overall user experience.

7. Documentation and Reporting:

Document all aspects of the design and data analysis process, including findings, recommendations, and any necessary improvements or modifications. A comprehensive report is essential for transparency and future reference.

By following this design and data analysis methodology, you can create a robust and secure image encryption system using the Blowfish algorithm while ensuring it meets both security and performance requirements.

5.4 Project Execution

5.4.1 Databases created and used

MySQL RDBMS

Name
Image_encryption

Fig 5.4.1(a) MySQL Database

Table: cloud

Columns

	Field Name	Datatype	Len
*	fid	varchar	100
	filenm	varchar	100
	enc_image	longblob	
	enckey	blob	
	owner	varchar	100

Fig 5.4.1(b) Structure of cloud table of MySQL Database

Definition

```
CREATE TABLE `cloud` (
  `fid` varchar(100) DEFAULT NULL,
  `filenm` varchar(100) DEFAULT NULL,
  `enc_image` longblob,
  `enckey` blob,
  `owner` varchar(100) DEFAULT NULL
)
```

Table: **owner**

Columns

	Field Name	Datatype	Len
*	name	varchar	100
	uid	varchar	500
	pwd	varchar	500
	email	varchar	500
	mno	varchar	500

Fig 5.4.1(c) Structure of owner table of MySQL Database

Definition

```
CREATE TABLE `owner` (
  `name` varchar(100) DEFAULT NULL,
  `uid` varchar(500) DEFAULT NULL,
  `pwd` varchar(500) DEFAULT NULL,
  `email` varchar(500) DEFAULT NULL,
  `mno` varchar(500) DEFAULT NULL
)
```

Table: **temp**

Columns

	Field Name	Datatype	Len
*	filenm	varchar	500
	filedata	longblob	

Fig 5.4.1(d) Structure of temp table of MySQL Database

Definition:

```
CREATE TABLE `temp` (  
    `filenm` varchar(500) DEFAULT NULL,  
    `filedata` longblob  
)
```

Table: **users**

Columns

	Field Name	Datatype	Len
*	filenm	varchar	500
	filedata	longblob	

Fig 5.4.1(e) Structure of users table of MySQL Database

Definition:

```
CREATE TABLE `users` (  
    `name` varchar(100) DEFAULT NULL,  
    `uid` varchar(100) DEFAULT NULL,  
    `pwd` varchar(100) DEFAULT NULL,  
    `email` varchar(100) DEFAULT NULL,  
    `mno` varchar(100) DEFAULT NULL )
```

5.4.2 Code for implementation

1) Database Connection

```
package databaseconnection;  
import java.sql.*;  
  
public class databasecon  
{  
    static Connection co;  
    public static Connection getconnection()
```

```

        {
            try
            {
                Class.forName("com.mysql.jdbc.Driver");
                co
                DriverManager.getConnection("jdbc:mysql://localhost:3306/image_encryption","root","root");
            }
            catch(Exception e)
            {
                System.out.println("Database Error"+e);
            }
            return co;
        }
    }
}

```

2) Blowfish Algorithm

```

package cryptography;

import java.security.SecureRandom;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

public class BlowFish {

    public static byte[] getRawKey(byte[] seed) throws Exception {
        KeyGenerator kgen = KeyGenerator.getInstance("Blowfish");
        SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
        sr.setSeed(seed);
        kgen.init(128, sr);
        SecretKey skey = kgen.generateKey();
        byte[] raw = skey.getEncoded();
        return raw;
    }
}

```

```

    }

    public static byte[] encrypt(byte[] sk, byte[] fildata) throws Exception {

        byte[] rawKey = getRawKey(sk);
        SecretKeySpec skeySpec = new SecretKeySpec(rawKey, "Blowfish");
        Cipher cipher = Cipher.getInstance("Blowfish");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(fildata);
        return encrypted;
    }

    public static byte[] decrypt(byte[] sk, byte[] encrypted)
        throws Exception {
        byte[] rawKey = getRawKey(sk);
        SecretKeySpec skeySpec = new SecretKeySpec(rawKey, "Blowfish");
        Cipher cipher = Cipher.getInstance("Blowfish");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec);
        byte[] decrypted = cipher.doFinal(encrypted);
        return decrypted;
    }

    public static String toHex(String txt) {
        return toHex(txt.getBytes());
    }

    public static String fromHex(String hex) {
        return new String(toByte(hex));
    }

    public static byte[] toByte(String hexString) {
        int len = hexString.length() / 2;
        byte[] result = new byte[len];
        for (int i = 0; i < len; i++)
            result[i] = Integer.valueOf(hexString.substring(2 * i, 2 * i + 2),
                16).byteValue();
        return result;
    }

```

```

    }

    public static String toHex(byte[] buf) {
        if (buf == null)
            return "";
        StringBuffer result = new StringBuffer(2 * buf.length);
        for (int i = 0; i < buf.length; i++) {
            appendHex(result, buf[i]);
        }
        return result.toString();
    }

    public final static String HEX = "0123456789ABCDEF";

    public static void appendHex(StringBuffer sb, byte b) {
        sb.append(HEX.charAt((b >> 4) & 0x0f)).append(HEX.charAt(b & 0x0f));
    }

    public static void main(String a[]){
        String seedValue = "123456";
        String MESSAGE = "hi";

        byte[] encryptedData;
        byte[] decryptedData;
        try {
            encryptedData=BlowFish.encrypt(seedValue.getBytes(),MESSAGE.getBytes());
            decryptedData= BlowFish.decrypt(seedValue.getBytes(), encryptedData);
            System.out.println(new String(encryptedData));
            System.out.println(new String(decryptedData));
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

3) encrypt.jsp

```

<%@ include file="oheader.jsp"%>
<%@ page import="java.io.*"%>
<%@ page import="databaseconnection.*,java.util.*"%>
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*"%>
<%@ page import="databaseconnection.*,cryptography.*"%>
<%@ page import="java.sql.*"%>
<div class="page-header">
    <div class="overlay">
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    <h1><center>Image Encryption</center></h1>
                </div>
            </div>
        </div>
    </div>
</div>

<%! int rno=0;
String s=null,unm;StringBuffer filedata=null;
byte[] data;
%>
<%
    unm=(String)session.getAttribute("unm");
    String fid=request.getParameter("fid");
    String fnm=request.getParameter("fname");
    String symkey=request.getParameter("key");
    session.setAttribute("symkey",symkey);

    Connection con1=databasecon.getconnection();
    Statement st=con1.createStatement();
    ResultSet r=st.executeQuery("select *from temp ");
    if(r.next())
    {
        byte[] encryptedImage = BlowFish.encrypt(symkey.getBytes(), r.getBytes(2));
        PreparedStatement p=con1.prepareStatement("insert into cloud(fid,filenm,enc_image,enckey,owner)
        values(?,?,?,?,"+unm+"");
        p.setInt(1,Integer.parseInt(fid));

```



```

p.setString(2,fnm);
p.setBytes(3,encryptedImage);
p.setString(4,symkey);
p.executeUpdate();
}
%>
<%

Connection con=databasecon.getConnection();
Statement st1=con.createStatement();
Statement st1=con.createStatement();
ResultSet r1=st1.executeQuery("select enc_image from cloud where fid="+fid+" ");
if(r1.next())
{
data=r1.getBytes(1);
}
%>
<br>

<section id="contact" class="contact">
<div class="container">
<br>
<h2>File Sharing</h2>
<br>
<div class="row">
<div class="col-lg-12">
<form method="post" action="share.jsp" id="contactForm" >
<div class="row">
<div class="col-md-6 wow fadeInLeft" data-wow-duration="2s" data-wow-
delay="600ms">
<div class="form-group">
<label>File Id</label>
<input type="text" class="form-control" required name="fid" value=<%=fid%>
>

<p class="help-block text-danger"></p>
</div>
<div class="form-group">
<label>File Name</label>
<input type="text" class="form-control" required name="fname"

```

```

value="<%=fnm%>" >
    <p class="help-block text-danger"></p>
</div>
    <div class="form-group">
        <label>Encrypted ImageData</label>
<textareaplaceholder=" " class="form-control" name="file" required type="text" cols="60" rows="5"
required=""><%=new String(data)%></textarea>
        <p class="help-block text-danger"></p>
    </div>
    <div class="form-group">
        <label>Select User</label>
<%=
Statement st2 = con.createStatement();
String sss1 = "select *from users";%>
<select name="ruid" class="form-control" required multiple>
    <option value="">Select User Id</option>
    <%=ResultSet rs2=st2.executeQuery(sss1);
while(rs2.next())
{
%>
    <option value=<%=rs2.getString("uid")%>><%=rs2.getString("uid")%></option>
    <%=}%>
</select>
        <p class="help-block text-danger"></p>
    </div>
    <div class="form-group">
        <button type="submit" class="btn btn-primary">Upload </button>
    </div>
</div>
</div>
</form>
</div>
</div>
</div>
</section>
    <br>
    <%=@ include file="footer.jsp"%>

```

5.4.3 Functional modules

Data Owner Registration:

In this module, the data owner will be registered with this application by filling out the registration form. Here, the registration form will be having the following fields such as name, user ID, password, email, and mobile number. The submitted registration form details will be stored in the MySQL database.

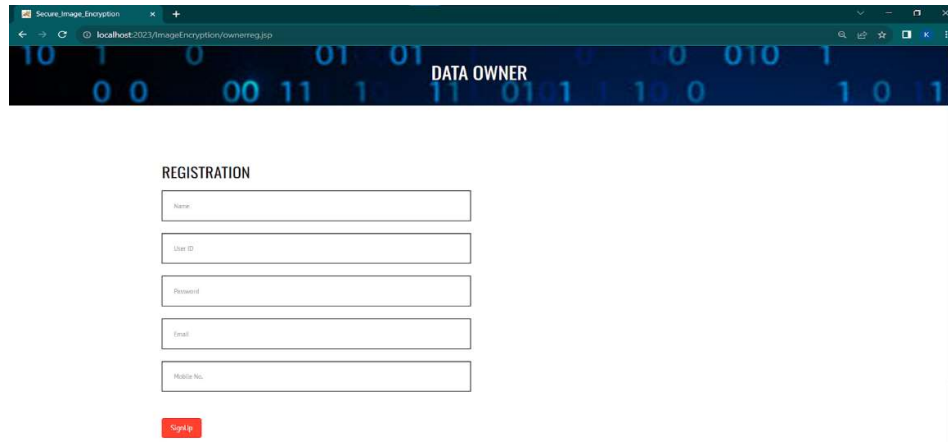
A screenshot of a web browser displaying a registration form titled "DATA OWNER". The browser's address bar shows "localhost:2023/imageEncryption/ownerreg.jsp". The form is titled "REGISTRATION" and contains five input fields: "Name", "User ID", "Password", "Email", and "Mobile No.". Below the fields is a red "SignUp" button. The background of the page features a dark blue header with the text "DATA OWNER" and a pattern of binary code (0s and 1s).

Fig 5.4.3(a) User Interface(UI) of Data owner registration

Data Owner Login:

In this module, the data owner will be authenticated with this application by entering valid login credentials like user id and password. Thereafter submitting the login credentials will be verified with the database, so if the credentials are valid then the data owner will be redirected to the dashboard, otherwise denied access to their dashboard.

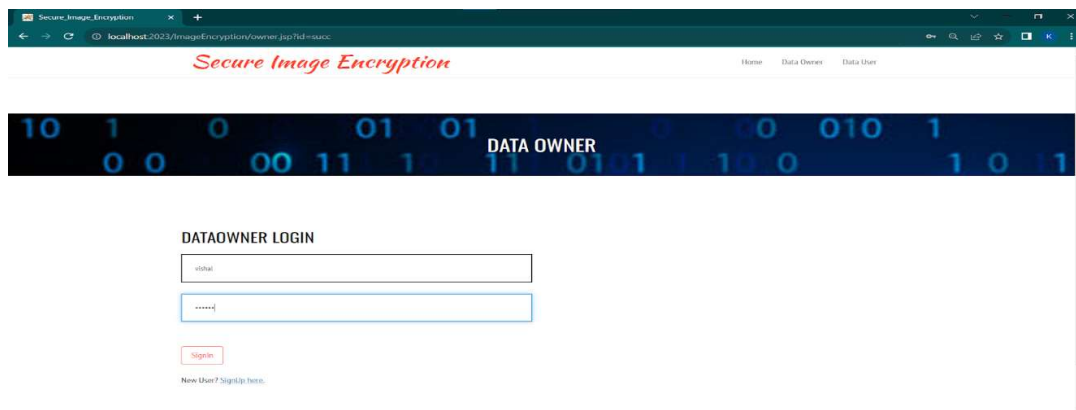
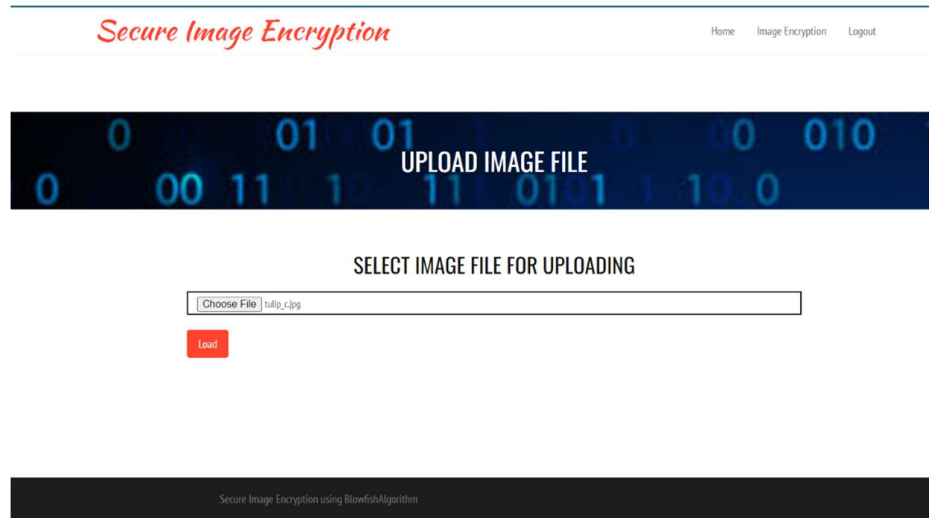
A screenshot of a web browser displaying a login form titled "DATAOWNER LOGIN". The browser's address bar shows "localhost:2023/imageEncryption/owner.jsp?id=msuc". The page has a header with the text "Secure Image Encryption" and navigation links for "Home", "Data Owner", and "Data User". The background features a dark blue header with the text "DATA OWNER" and a pattern of binary code. The login form contains two input fields: "Email" and "Password". Below the fields is a red "Login" button. At the bottom, there is a link that says "New User? SignUp here".

Fig 5.4.3(b) User Interface(UI) of Data owner login

Image Encryption:

In this module, the data owner will upload the required image files into the database server by using blowfish cryptography technique. While image file uploading, the data owner will select an image file which will be encrypted with the Blowfish algorithm with the help of a random secure key. When the image file upload form is submitted then the encrypted image will be stored in the database.



The screenshot shows a web application titled "Secure Image Encryption" in red cursive font. The navigation bar includes links for "Home", "Image Encryption", and "Logout". The main content area features a dark blue header with binary code and the text "UPLOAD IMAGE FILE". Below this is a section titled "SELECT IMAGE FILE FOR UPLOADING" containing a file selection button labeled "Choose File" with the filename "tulip.jpg" and a red "Load" button. At the bottom, a black bar displays the text "Secure Image Encryption using Blowfish Algorithm".

Fig 5.4.3(c) User Interface(UI) of image file upload



(Without image compression; 34kb, 240dpi)



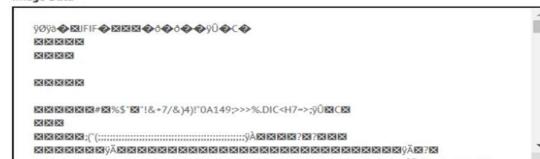
(With image compression; 32kb, 200dpi)

Fig 5.4.3(d) Original images

FILE SHARING

File Id

File Name

Image Data


SymmetricKey

Encrypt

Fig 5.4.3(e) User Interface(UI) of file sharing

Data User Registration Module:

In this module, the data user will be registered with this application by filling out the registration form. Here, the registration form will have the following fields such as name, user ID, password, email, and mobile number. The submitted registration form details will be stored in the MySQL database.

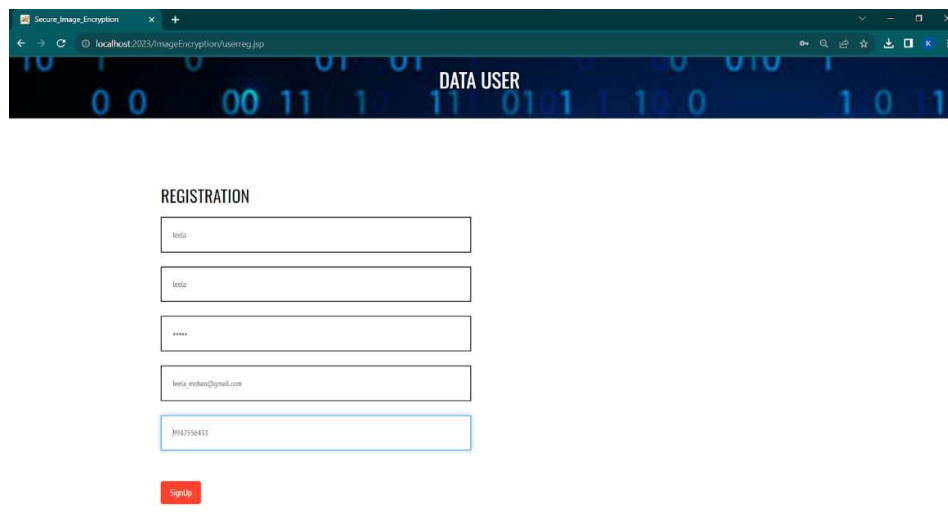


Fig 5.4.3(f) User Interface(UI) of Data user registration

Data User Login Module:

In this module, the data user will be authenticated with this application by entering valid login credentials like user ID and password. Thereafter submitting the login credentials will be verified with the database, so if the credentials are valid then the data user will be redirected to the dashboard, otherwise denied access to their dashboard.

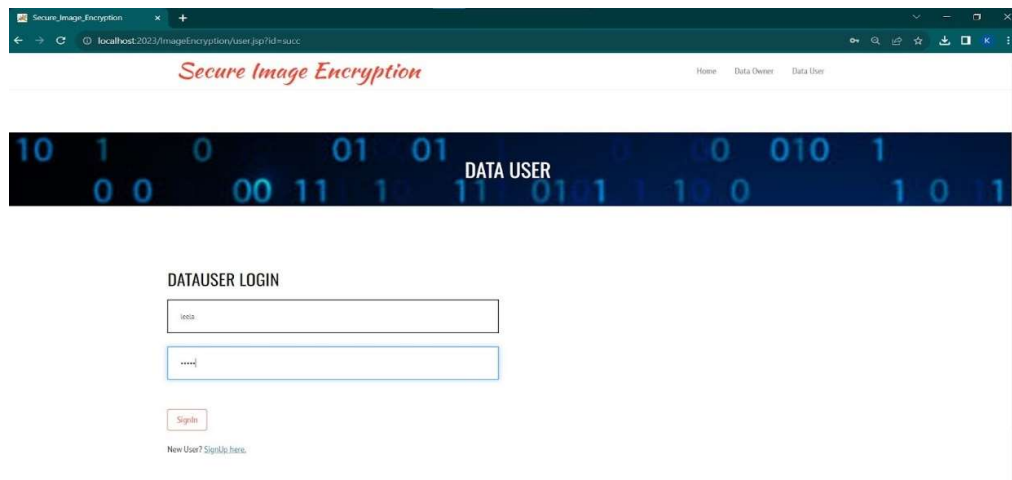


Fig 5.4.3(g) User Interface(UI) of Data user login

File Download:

In this module, the data user will be able to download the required image files from the database server. When the data user clicks on the Image decryption hyperlink then this application will display a list of the uploaded file details. Thereafter, they can choose the required image file for downloading. While the image file is downloading, the secure key will get from the database and thereafter the encrypted image will be decrypted with respective secure keys. Finally, the data user will download the image file from a database server.





6. RESULTS

6.1 Expected Results

Given below are the results found from this project.

1. Variable Key Length:

Blowfish supports variable key lengths, ranging from 32 to 448 bits. This flexibility allows for adaptation to specific security requirements, offering a level of customization not found in fixed-key algorithms like DES.

2. Performance:

Blowfish is known for its efficient and relatively fast performance. Its algorithmic simplicity contributes to quick encryption and decryption processes, making it suitable for applications where speed is crucial.

3. No Known Vulnerabilities:

Blowfish has not been found to have any significant vulnerabilities. Its resilience against known cryptographic attacks adds to its appeal as a secure encryption algorithm.

4. Open Source and Publicly Vetted:

Blowfish is an open-source algorithm, and its code has been widely scrutinized by the cryptographic community. The transparency of its design allows for public evaluation, contributing to increased confidence in its security.

5. Compact Code Size:

Blowfish's implementation typically requires less code than some other encryption algorithms. This can be advantageous in resource-constrained environments, such as embedded systems or mobile applications, where minimizing code size is essential.

6. Ease of Implementation:

Blowfish's straightforward algorithm makes it relatively easy to implement and integrate into various systems. This simplicity can be an advantage for developers seeking a practical and effective encryption solution.

7. Adoption in Certain Applications:

While not as widely adopted as AES in mainstream applications, Blowfish has found use in specific contexts, demonstrating its applicability and reliability in certain niches.

In conclusion, Blowfish offers a balance of security, flexibility, and performance, making it a viable choice in scenarios where its unique features align with project requirements. As always, the suitability

of an encryption algorithm depends on the specific needs and context of the application.

6.2 Comparing Blowfish with other algorithms

Our project based on image encryption using blowfish is based on the requirement of secure image transmission over the internet. This algorithm overcomes the challenges faced by other symmetric algorithms such as DES and AES. It has been designed to ensure privacy as well as smooth transfer of data between users.

The encryption and decryption speed of Blowfish can be affected by the processing power of the CPU. Faster CPUs can generally perform the encryption and decryption operations more quickly. Some implementations of Blowfish may take advantage of parallel processing to enhance performance. This can be particularly helpful when dealing with larger images or a large number of images.

The performance of Blowfish can be influenced by the efficiency of its implementation. Optimized code can significantly improve the speed of encryption and decryption operations. Blowfish is known for having the least decrypting time and hence often chosen as the algorithm to decrypt large size data files.

Parameters	Symmetric Key algorithms				
	AES	DES	3DES	BLOWFISH	IDEA
Created by	Joan Daemen, Vincent Rijmen	IBM	IBM	Bruce Schneier	Xuejia Lai, James Massey
Year	2000	1997	1978	1993	1991
Block Size	128 bits	64 bits	64 bits	64 bits	64 bits
Key Size	128,192 or 256 bits	56 bits	122 or 168 bits	32-448 bits	128 bits
Structure of algorithm	Substitution-permutation	Balanced Feistel network	Feistel network	Feistel Network	Lai-Massey Scheme
Rounds	10 (128 bits), 12 (192 bits), 14 (256 bits)	16	48	16	8.5
Encryption	Fast	Medium	Fast	Medium	Fast
Decryption	Fast	Medium	Slow	Fast	Slow

Fig 6.1(a) comparison of different symmetric key algorithms

As shown in the graph below, blowfish has the least encrypting and decrypting time when it comes to images of small size like 1kb and 10kb. As the size increases, its performance can be influenced by the key length chosen, CPU processing speed and data transfer overhead.

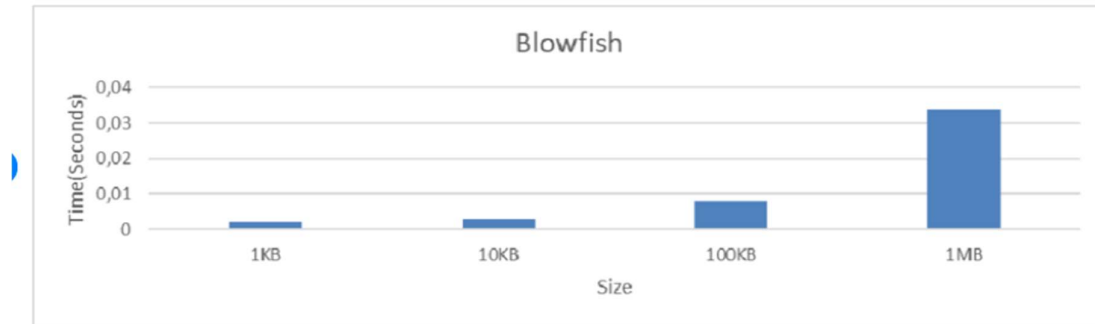


Fig 6.2(b) comparison of working of blowfish with various file sizes

6.3 Applications

The project holds significant promise in various domains.

1. Medical Imaging:

The project's application in the medical field is of paramount importance. It can be used to secure patient records, medical images (X-rays, MRIs, CT scans), and telemedicine data. Blowfish encryption ensures the confidentiality and integrity of sensitive medical information, helping healthcare providers comply with privacy regulations like HIPAA.

2. Secure Communication:

Ensuring the security of images transmitted over networks is crucial for secure communication. This project can be employed in messaging apps, email services, and video conferencing platforms to protect user privacy. It can be part of end-to-end encryption measures, safeguarding images from interception and tampering.

3. Multimedia Data Protection:

In the media and entertainment industry, the protection of multimedia content, such as movies, images, and audio, is essential to prevent piracy and unauthorized distribution. Blowfish encryption can be used to secure multimedia files, especially during the production and distribution phases, thus safeguarding intellectual property.

4. Data Archiving and Storage:

Organizations often store large amounts of image data for various purposes, including historical records, research, and documentation. This project can be applied to ensure the long-term security and confidentiality of archived image data, especially in industries like museums, libraries, and research institutions.

5. Government and Defense:

Government agencies and defense organizations deal with highly sensitive imagery, including satellite images, surveillance footage, and classified documents. Blowfish encryption can help protect these assets from espionage and cyberattacks, ensuring national security and data integrity.

6. Forensics and Law Enforcement:

In criminal investigations, law enforcement agencies use digital images as evidence. Ensuring the security of these images is crucial for maintaining the chain of custody. This project can aid in protecting the integrity of digital evidence, preventing tampering, and ensuring admissibility in court.

7. Digital Watermarking:

Digital watermarking is used to protect the copyright and ownership of images. This project can be integrated with watermarking techniques to secure images from unauthorized use while maintaining their visual quality.

8. Cloud Storage and Backup:

Cloud service providers store vast amounts of user data, including images. Blowfish encryption can be implemented for secure cloud storage and backup solutions, providing users with confidence in the privacy and security of their digital assets.

9. Research and Academia:

Academic institutions and research organizations generate and store a wealth of image data for research purposes. This project can assist in securing research findings and preventing data breaches, ensuring the confidentiality and integrity of scholarly work.

10. E-commerce and Product Imagery:

E-commerce platforms rely heavily on product images. Protecting these images from theft, alteration, or unauthorized distribution is essential for the success of online businesses. Blowfish encryption can be used to secure product imagery and maintain brand integrity.

6.4 Merits

Image encryption using the Blowfish algorithm offers several merits and advantages for a project or application. Here are some of the key benefits:

1. Strong Security:

The Blowfish algorithm is a well-established symmetric-key encryption algorithm known for its robust security features. It provides strong encryption and helps protect image data from unauthorized access and tampering.

2. Confidentiality:

Image encryption with Blowfish ensures the confidentiality of sensitive image content. Only authorized parties with the encryption key can decrypt and view the original image.

3. Integrity:

Blowfish encryption helps maintain the integrity of image data. Any unauthorized changes or modifications to the encrypted image will result in decryption failure, providing a way to detect tampering.

4. Efficiency:

Blowfish is a fast encryption algorithm, making it suitable for real-time or near-real-time image encryption and decryption. This efficiency is essential for applications that require quick access to secured image data.

5. Compatibility:

Blowfish encryption can be implemented in various programming languages and platforms, making it versatile and compatible with a wide range of systems and applications.

6. Scalability:

Image encryption with Blowfish can be scaled to handle large image datasets and high-throughput operations, making it suitable for both small-scale and large-scale projects.

7. Cross-Platform Support:

Blowfish encryption is platform-independent, allowing encrypted images to be shared and accessed across different operating systems and devices.

8. User-Friendly Encryption:

When implemented with a user-friendly interface, image encryption using Blowfish can be accessible and easy to use for non-technical users, which is beneficial for projects with a broad user base.

9. Versatility:

The same Blowfish encryption process can be applied to a wide range of image formats, making it versatile for encrypting different types of image data.

10. Reduced Risk of Data Breaches:

By using strong encryption, projects that involve sensitive image data can significantly reduce the risk of data breaches, ensuring that confidential images remain protected.

11. Secure Sharing:

Blowfish encryption allows for secure sharing of image data. Authorized users can safely transmit encrypted images, knowing that the content remains confidential during transmission.

12. Legal Compliance:

Projects that handle sensitive or regulated image data can benefit from Blowfish encryption to meet legal and compliance requirements, such as HIPAA or GDPR.

13. Data Protection:

Blowfish encryption contributes to data protection by ensuring that image data is stored and transmitted in a secure and protected state.

In summary, image encryption using the Blowfish algorithm provides a strong, efficient, and versatile solution for projects that require secure image data handling and sharing. Its security features, cross-platform compatibility, and scalability make it a valuable tool for safeguarding sensitive image content.

6.5 Demerits

While the project holds considerable promise, it's essential to acknowledge potential drawbacks and limitations:

1. Security Concerns:

Algorithm Aging: Blowfish is an aging encryption algorithm that was designed in 1993. While it has been widely used and studied over the years, it may not be as secure as more modern encryption algorithms due to advancements in cryptanalysis techniques. There's a risk that vulnerabilities in Blowfish may be discovered in the future, potentially compromising the security of the encrypted images.

2. Performance and Efficiency:

Computational Overhead: Blowfish encryption and decryption can be computationally intensive, particularly when dealing with large image files. This can lead to slower processing times. Blowfish may consume significant system resources, including CPU and memory, which can be problematic in

resource-constrained environments like mobile devices or embedded systems.

3.Key Management:

Key Generation and Storage: Proper key management is crucial for the security of any encryption system. Generating and securely storing encryption keys can be a challenging task. Poor key management practices can undermine the overall security of the image encryption process, potentially leading to key leakage and unauthorized access to sensitive images.

4.Image Quality:

Loss of Image Quality: Encryption algorithms, including Blowfish, can introduce noise or artifacts into the encrypted images. Depending on the specific implementation and settings, this may result in a noticeable degradation of image quality. In applications where image fidelity is essential, this could be a significant drawback.

5.Compatibility and Interoperability:

Limited Support: Not all software or platforms may support Blowfish encryption. This can create compatibility issues when trying to exchange or work with encrypted images across different environments or applications, potentially limiting the usability of the encrypted images.

6.Resistance to Attacks:

Dependent on Implementation: The resistance of Blowfish to common attacks relies heavily on the specific implementation and the cryptographic parameters used. If not configured correctly, the system could remain vulnerable to attacks, potentially compromising image security.

7.Lack of Forward Secrecy:

Key Compromise: Blowfish is a symmetric key encryption algorithm. If an encryption key is compromised, all past and future communications or images encrypted with that key can be decrypted. This lack of forward secrecy can be a significant disadvantage in scenarios where ongoing security is critical.

8.Key Length Limitation:

Fixed Key Size: Blowfish has a fixed block size and key size, which may limit the strength of encryption provided, especially when compared to modern encryption algorithms like AES, which support variable key lengths.

9.Continuous Research and Development:

Evolving Cryptography: The field of cryptography is continually evolving, with new encryption algorithms and security standards emerging regularly. By opting for Blowfish, you might miss out on the latest advancements and best practices in encryption technology, potentially leaving your image security measures less robust.

10. Regulatory Compliance:

Compliance Challenges: Depending on the specific application and industry, there may be regulatory or compliance requirements for image encryption. The choice of encryption algorithm could impact your ability to meet these requirements, potentially leading to legal or regulatory issues.

CONCLUSION

In conclusion, the implementation of the Blowfish algorithm in the image-sharing app marks a significant achievement in the realm of secure image encryption. The ability to encrypt both colour and black & white images of any size and in various formats, such as BMP, PNG, and JPG, showcases the versatility of Blowfish in protecting a wide range of visual data. The unique feature of Blowfish allowing variable key lengths and its efficiency in handling encryption and decryption processes contribute to the success of the project.

The observed transformation of the histogram in encrypted images, characterized by reduced dynamism and distinct differences from the original image histograms, serves as a visual testament to the security and efficacy of the Blowfish algorithm in safeguarding image content.

The formidable resistance of Blowfish to brute-force attacks, requiring an attacker to attempt an astronomical $2^{(28r+1)}$ combinations, emphasizes its robust security. Furthermore, the recognition that increasing the number of rounds enhances the algorithm's strength adds an extra layer of assurance.

This project contributes significantly to the field of image encryption by demonstrating the practical application of Blowfish across diverse image formats. The adaptability of Blowfish to handle images of varying sizes and types, coupled with its resilience against known attacks, positions it as a reliable standard for image encryption.

The open-source nature of Blowfish, its ease of implementation, and the absence of known security weak points make it a noteworthy contender for broader adoption in image security applications. This project, by showcasing the effectiveness of Blowfish in a real-world scenario, encourages further exploration and consideration of Blowfish in image encryption research.

Exploring the performance of Blowfish in real-time image encryption scenarios, especially in applications with stringent speed requirements, could be a valuable direction.

Additionally, conducting a comparative analysis with other contemporary encryption algorithms could provide a more comprehensive understanding of Blowfish's strengths and weaknesses in the context of evolving security standards.

Furthermore, considering the rapidly advancing landscape of encryption and cybersecurity, regular updates and evaluations of Blowfish's resistance to emerging threats would be prudent. Collaborative efforts with the cryptographic community could contribute to refining and evolving Blowfish as a reliable encryption standard.

FUTURE ENHANCEMENTS

1. Multi-Algorithm Support:

- Research and select additional encryption algorithms that are well-regarded for security. Some options include AES, RSA, and ECC.
- Implement a modular encryption system that allows users to choose their preferred algorithm during the encryption process.
- Provide clear documentation and explanations for each encryption algorithm to help users make informed decisions.

2. User Authentication:

- Develop a robust user authentication system with strong password hashing using algorithms like bcrypt or Argon2.
- Implement two-factor authentication (2FA) to add an extra layer of security.
- Consider biometric authentication options for mobile users, such as fingerprint or face recognition.

3. File Format Compatibility:

- Research and support a variety of image formats, ensuring that your encryption process is compatible with popular file types like JPEG, PNG, GIF, etc.
- Explore the possibility of extending compatibility to other file types by developing or integrating encryption methods tailored to those formats.

4. Mobile App Integration:

- Choose a cross-platform development framework like React Native or Flutter to streamline the development process for both iOS and Android.
- Ensure a seamless user experience on mobile devices, optimizing the app for different screen sizes and resolutions.
- Implement secure APIs for communication between the mobile app and the server.

5. Decentralized Storage:

- Research decentralized storage solutions such as InterPlanetary File System (IPFS) or blockchain-based storage platforms.
- Develop a strategy for integrating decentralized storage into your application architecture.
- Provide users with options to choose between centralized and decentralized storage based on their preferences.

6. Advanced Key Management:

- Implement key rotation policies to regularly update encryption keys, reducing the risk of compromised keys.
- Develop a key recovery mechanism to help users regain access to their encrypted images in case they forget their passwords or lose access to their keys.
- Explore the use of hardware security modules (HSMs) for secure key storage and management.

These suggestions should help you get started on enhancing your image encryption web app. Keep in mind that security and user experience should be at the forefront of any new implementation or feature.

BIBLIOGRAPHY

- Schneier, B. (1994). "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)." Fast Software Encryption, 191-204.
<https://www.schneier.com/academic/blowfish>
- Ishwaq T. Hashim, Ammar H. Jassem, Suhad A. Ali "A Novel Design of Blowfish Algorithm for Image Security "
- Viswanath Matukumalli, Hariharan Seelam, Sai Gowtham Inturi, Dr. Vijay Babu Burra "Execution Analysis of Different Cryptographic Encryption Algorithms on Different File Sizes on the Cloud and Off the Cloud"
- Shraphalya B. Nalawade,Dhanashri H. Gawali "Design and Implementation of Blowfish Algorithm using Reconfigurable Platform"
- Pia Singh, Prof. Karamjeet Singh "IMAGE ENCRYPTION AND DECRYPTION USING BLOWFISH ALGORITHM IN MATLAB "
- Irfan Landge,Burhanuddin Contractor , Aamna Patel and Rozina Choudhary "Image encryption using blowfish algorithm"
- Ismet Öztürk and Ibrahim Sogukpınar "Analysis and Comparison of Image Encryption Algorithms"
- Ammar H. Jassem ,Ashwaq T. Hashim and Suhad A. Ali "Enhanced Blowfish Algorithm for Image Encryption Based on Chaotic Map "
- Comparison between different algorithms:
<https://www.baeldung.com/cs/des-vs-3des-vs-blowfish-vs-aes>
- Blowfish algorithm: <https://www.geeksforgeeks.org/blowfish-algorithm-with-examples/>
- Blowfish Algorithm in Cryptography with Cybersecurity:
<https://www.javatpoint.com/blowfish-algorithm-in-cryptography>
- Encrypting data with the Blowfish algorithm:
<https://www.design-reuse.com/articles/5922/encrypting-data-with-the-blowfish-algorithm.html>
- What is data encryption and decryption in blowfish algorithm:
<https://www.tutorialspoint.com/what-is-data-encryption-and-decryption-in-blowfish-algorithm>