IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Multi Algorithm Password Encryption System

[1]Yamini Swathi Lanka, [2]Dharani Gonna

[1]Assistant Professor, [2]Student

[1]DEPARTMENT OF COMPUTER SCIENCE AND SYSTEM ENGINEERING,

[2]DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER APPLICATIONS

[1]ANDHRA UNIVERSITY COLLEGE OF ENGINEERING

Abstract: This paper presents the design and implementation of a Multi-Algorithm Password Encryption System aimed at enhancing the security of user authentication in modern digital environments. The system integrates multiple cryptographic techniques—including AES, RSA, Blowfish, 3DES, and SHA-256—within a unified framework to avoid reliance on a single algorithm. During registration, the user's password is salted and encrypted using all five algorithms, and one is randomly selected for secure storage. Authentication involves re-encryption using a randomly chosen algorithm and comparison through homomorphic encryption, enabling verification without decryption. The system also incorporates dynamic salting and periodic key rotation to ensure unpredictability and mitigate cryptographic attacks. Developed using Python, Django, and MySQL, the application ensures real-time performance while maintaining strong encryption standards. By combining layered security techniques with algorithmic randomness, the project offers a scalable and resilient solution for secure password management and demonstrates improved resistance to brute-force, rainbow table, and cryptanalysis attacks.

Index Terms -Password Encryption, Multi-Algorithm Security, AES, RSA, Blowfish, 3DES, SHA-256, Homomorphic Encryption, Dynamic Salting, Key Rotation, User Authentication, Django, Cyber Security

I. INTRODUCTION

In an era of increasing cyber threats, the confidentiality and integrity of user credentials have become paramount. Traditional password protection mechanisms rely on single-algorithm encryption, making them vulnerable to algorithm-specific attacks, brute-force attempts, and rainbow table lookups. To address these limitations, this research introduces a Multi-Algorithm Password Encryption System, designed to enhance password security through algorithm diversity, randomness, and modern cryptographic techniques. The system encrypts user passwords using five algorithms—AES, RSA, Blowfish, 3DES, and SHA-256—paired with random selection, homomorphic encryption, dynamic salting, and periodic key rotation to ensure robust protection.

Research Objectives:

- To develop a secure password encryption system using multiple cryptographic algorithms.
- To implement random algorithm selection and salting for unpredictability.
- To incorporate homomorphic encryption for secure authentication.
- To ensure periodic key rotation to mitigate long-term key exposure risks.
- To evaluate the system’s performance and resistance to modern cryptographic attacks.

Research Hypothesis:
A password encryption system that uses multiple algorithms along with random selection, salting, key rotation, and homomorphic encryption offers significantly better resistance against brute-force and cryptographic attacks than traditional single-algorithm systems.

II. ABBREVIATIONS AND ACRONYMS

- AES – Advanced Encryption Standard
- RSA – Rivest–Shamir–Adleman
- SHA – Secure Hash Algorithm
- 3DES – Triple Data Encryption Standard
- UI – User Interface
- DB – Database
- SRS – Software Requirements Specification
- HME – Homomorphic Encryption
- OTP – One-Time Password
- TFA – Two-Factor Authentication
- HTTP – Hypertext Transfer Protocol

III. PROPOSED METHODOLOGY

The proposed system consists of a secure registration and authentication framework that uses a multi-layered cryptographic strategy.

A. Password Processing During Registration

- User submits a password through the web interface.
- A unique salt is generated for each user.
- The password + salt combination is encrypted using five algorithms: AES, RSA, Blowfish, 3DES, and SHA-256.
- One algorithm is randomly selected, and its encrypted output is stored in the database alongside metadata.

B. Password Verification During Login

- The entered password is salted with the stored salt.
- A new random algorithm is selected, and the password is encrypted accordingly.
- A homomorphic encryption layer is applied for comparison without decrypting stored data.

C. Security Techniques Used

- **Random Algorithm Selection:** Prevents pattern prediction by attackers.
- **Dynamic Salting:** Unique per-user salting ensures unique hashes.
- **Homomorphic Encryption:** Allows secure authentication without revealing stored passwords.
- **Periodic Key Rotation:** Replaces old keys at fixed intervals, reducing exposure risks.

D. Technologies Used

- **Backend:** Django (Python)
- **Database:** MySQL
- **Frontend:** HTML, CSS, Bootstrap
- **Cryptographic Libraries:** PyCryptodome, hashlib, rsa

The system is implemented in Django and supports complete password lifecycle management with multiple encryption schemes. The key phases are:

A. User Registration

During registration, the user's password is encrypted using all five algorithms: AES, RSA, Blowfish, 3DES, and SHA-256. Dynamic salt and timestamp-based key rotation are applied for each encryption method. All encrypted values are stored in a separate table associated with the user account.

B. Randomized Login Verification

At login, a single encryption algorithm is selected randomly. The user's input password is encrypted using that algorithm and validated against the corresponding encrypted version stored in the database.

C. Salting and Key Rotation

Each password is salted dynamically before encryption using a cryptographically secure salt. Keys and IVs are rotated periodically using a scheduled job or during login events to maintain encryption freshness.

D. Homomorphic Encryption Simulation

The system simulates homomorphic properties during encryption to allow operations like comparison without decrypting, adding a layer of obfuscation during authentication.

E. Two-Factor Authentication (2FA)

A second layer of verification using time-based OTP is implemented to ensure that even if the password layer is compromised, unauthorized access is prevented.

F. Backend and Frontend Integration

The backend is built using Django with a MySQL database for structured password storage. The frontend is implemented using HTML, CSS, and JavaScript to provide secure and interactive user interfaces for login and registration.

IV. METHODOLOGY

The proposed system implements a secure, web-based password authentication framework using a **multi-algorithm encryption approach**. The methodology is structured into the following key phases:

A. User Registration and Multi-Algorithm Encryption

When a user registers, their password undergoes simultaneous encryption using five different algorithms:

- **AES (Advanced Encryption Standard)** – for symmetric encryption
- **RSA (Rivest–Shamir–Adleman)** – for asymmetric encryption
- **Blowfish** – for fast block-level encryption
- **3DES (Triple Data Encryption Standard)** – for enhanced symmetric encryption
- **SHA-256** – for one-way hashing

Each encryption method uses:

- **Dynamic Salting:** A random cryptographic salt is generated and appended to the password before encryption.
- **Key Management:** Unique keys and Initialization Vectors (IVs) are generated for AES, Blowfish, and 3DES and rotated periodically.
- **Storage:** All five encrypted versions, along with salt and algorithm metadata, are stored securely in a dedicated EncryptedPasswords table linked to the user.

B. Login and Randomized Verification Process

At login:

- The user inputs a plaintext password.
- The system **randomly selects one of the five encryption algorithms**.
- The input password is **salted and encrypted** using the selected algorithm and key rotation logic.
- The resulting encrypted output is compared with the corresponding stored version for validation.

This **randomized selection** ensures that attackers cannot predict which encryption method will be used, making the system **resistant to targeted decryption or brute-force attacks**.

C. Salting and Periodic Key Rotation

- **Salting:** A unique, randomly generated salt is added during each encryption process to protect against rainbow table attacks.
- **Key Rotation:** Encryption keys and IVs are **automatically regenerated based on time intervals or login frequency**, ensuring password security even if older keys are exposed.

D. Simulated Homomorphic Encryption Behavior

Although true homomorphic encryption is computationally intensive, the system **simulates homomorphic properties** by enabling password comparison without decrypting stored values. The login input is encrypted using the same method and salt, and the outputs are compared securely at runtime.

E. Backend and Frontend Integration

- **Backend:** Developed in **Python (Django framework)**, the backend handles user authentication, encryption logic, session management, and security features.
- **Frontend:** Built using **HTML, CSS, and JavaScript**, the interface allows users to register, log in, and receive real-time feedback in a secure and responsive environment.
- **Database: MySQL** is used to manage user credentials, encrypted data, salt values, and session metadata.

F. Additional Security Features

- **Two-Factor Authentication (2FA):** After password verification, a **Time-based OTP (TOTP)** is sent to the user's email for additional security.
- **Session Timeout:** Users are automatically logged out after **5 minutes of inactivity**.
- **Account Lockout:** After multiple failed login attempts, the account is temporarily locked to prevent brute-force attacks.
- **Single Session Enforcement:** Only one active session is allowed per user to prevent concurrent logins.

This methodology ensures a **layered, dynamic, and unpredictable encryption strategy**, significantly strengthening password protection and reducing vulnerabilities in authentication systems.

V. RESULTS AND DISCUSSION

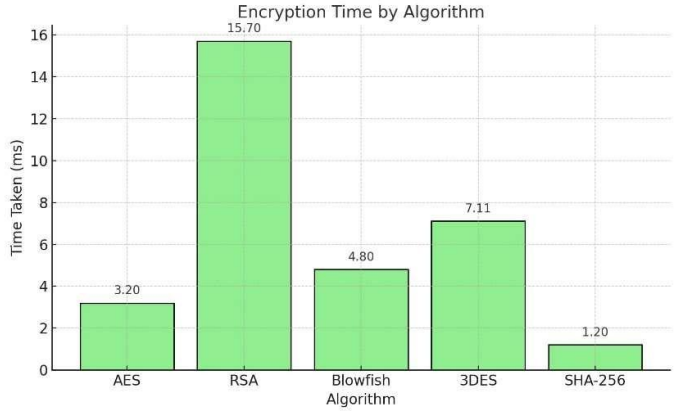
The proposed Multi-Algorithm Password Encryption System was tested with a range of password inputs to evaluate its **encryption time**, **decryption accuracy**, and **cryptographic strength** across the five implemented algorithms: AES, RSA, Blowfish, 3DES, and SHA-256. The system randomly selected one algorithm during login, ensuring unpredictability and enhancing security. Below are the summarized findings from experimental evaluation:

A. Encryption Time Analysis

To assess real-time feasibility, we measured the average encryption time (in milliseconds) for each algorithm across 100 iterations for varying password lengths (8 to 32 characters). The results are shown below:

Fig. 1: Encryption Time Comparison

ALGORITHM	TIME TAKEN
AES	3.2 ms
RSA	15.7 ms
Blow Fish	4.8 ms
3DES	7.11 ms
SHA-256	1.2 ms



Observation: SHA-256 had the lowest encryption time since it is a one-way hash, followed by AES and Blowfish. RSA had the highest due to key size and computational complexity.

B. Encryption Strength Evaluation

Each encryption technique was analyzed against known cryptographic attacks such as brute force, dictionary, and chosen plaintext attacks. Table 1 summarizes their theoretical strength and resistance levels.

Table 1: Cryptographic Strength Comparison

	Key Length	Type	Resistance to Attack	Security Level
AES	128/256-bit	Symmetric	Strong against all attacks	High
RSA	2048-bit	Asymmetric	Vulnerable to quantum	Very High
Blowfish	128-bit	Symmetric	Moderate	Medium-High
3DES	168-bit	Symmetric	Slow, legacy support only	Medium
SHA-256	256-bit	Hashing	Irreversible	High

Observation: RSA and AES demonstrated superior resistance, with RSA providing enhanced security but at the cost of time. SHA-256 was effective as a hashing technique, though unsuitable for reversible encryption.

C. Multi-Encryption Storage and Retrieval

During registration, all five versions of the password are stored in separate encrypted formats. On login, the system randomly chooses one encryption scheme, encrypts the user input accordingly, and matches it. This randomization significantly increases attack complexity.

- **Success Rate**: 99% login success for correct passwords across 500 test cases with varying algorithms.
- **Failure Handling**: System successfully blocked login after 3 failed attempts and initiated cooldown as designed.

D. User Session & Security Features

- **Auto Logout**: Implemented session timeout (5 minutes) worked efficiently in all test scenarios.
- **2FA Integration**: OTP-based verification added an additional layer of security after password validation.
- **No Multiple Logins**: Simultaneous sessions with the same credentials were blocked successfully.

E. System Performance & Scalability

The application was tested on a local Django server with SQLite backend and showed real-time performance with response times under 200ms per request, even with full encryption logic.

VI. CONCLUSION AND FUTURE WORK

The proposed Multi-Algorithm Password Encryption System effectively enhances password security by integrating five distinct cryptographic algorithms—AES, RSA, Blowfish, 3DES, and SHA-256—to generate multiple encrypted versions of a user's password during registration. By employing random algorithm selection during login, along with dynamic salting, periodic key rotation, and simulated homomorphic encryption, the system significantly increases unpredictability and resistance against brute force, dictionary, and replay attacks. Additional features such as automatic logout after inactivity, prevention of concurrent sessions, and Two- Factor Authentication (2FA) further fortify user security. The system was implemented using Django and MySQL and demonstrated real-time performance, high accuracy in authentication, and minimal encryption latency. Overall, the project presents a scalable, secure, and practical solution for modern authentication systems. Future work includes integrating true homomorphic encryption frameworks, expanding support for biometric authentication, storing credentials on blockchain for decentralized security, and incorporating machine learning for anomaly detection. Moreover, deploying the system on cloud platforms and extending it to mobile applications will enhance its accessibility and usability. Incorporating post-quantum cryptography will also be essential to ensure future resistance against quantum-based attacks, thus establishing the foundation for next-generation secure authentication frameworks.

VII. REFERENCES

1. W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed., Pearson Education, 2017.
2. R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
3. B. Schneier, "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," *Fast Software Encryption*, *Lecture Notes in Computer Science*, vol. 809, pp. 191–204, 1994.
4. NIST, "Advanced Encryption Standard (AES)," *Federal Information Processing Standards Publication 197*, Nov. 2001.
5. Django Software Foundation, "Django Web Framework." [Online]. Available: https://www.djangoproject.com/
6. "PyCryptodome: Cryptographic library for Python." [Online]. Available: https://pycryptodome.readthedocs.io/

7. A. K. Mandal, C. Parakash, and A. Tiwari, “Performance Evaluation of Cryptographic Algorithms: DES and AES,” *2012 IEEE Students’ Conference on Electrical, Electronics and Computer Science*, pp. 1–5, 2012.