JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

PASSWORD CRACKING USING QUANTUM COMPUTING

[1]Ms R.Sujatha , [2]R.Geethika, [3]L.Hema, [4]N.Muni Kalyan , [5]M.GuruVasmi

[1]Assistant Professor, [2345]Student
[12345] Department of Artificial Intelligence & Data Science
[12345]Annamacharya Institute of Technology & Sciences, Tirupati,
Andhra Pradesh – 517520 India

Abstract: **The rapid evolution of computational technologies poses new challenges and opportunities for data security, particularly in the domains of file protection and user authentication. This research introduces a novel system that enhances secure file sharing and access control by integrating robust classical cryptographic methods with a forward-looking exploration of quantum computing's implications. Leveraging the Advanced Encryption Standard (AES) for file encryption and bcrypt for password hashing, the proposed framework ensures confidentiality and integrity of stored data. A distinctive feature of this system is its user-centric file access mechanism, where encryption keys are securely distributed via email upon approval, complemented by brute force countermeasures to thwart unauthorized login attempts. Furthermore, this study delves into the theoretical application of quantum computing, specifically through simulations of password cracking, to assess vulnerabilities in current cryptographic techniques. By bridging classical security practices with quantum-inspired insights, this work aims to fortify digital systems against emerging threats while laying the groundwork for future quantum-resistant solutions. The findings underscore the potential of hybrid approaches in advancing cybersecurity resilience in an increasingly quantum-influenced landscape.**

Keywords **–file management, security, user module, admin module, encryption, file upload, file sharing, brute force protection, user login, IP blocking, data integrity, user accountability, administrative oversight, secure file sharing, automated email, decryption key, user registration, file transactions**

I.INTRODUCTION

In the contemporary digital era, the safeguarding of sensitive information has become a paramount concern as cyber threats grow in sophistication and scale. Traditional cryptographic systems, such as the Advanced Encryption Standard (AES) for data encryption and bcrypt for password protection, have long served as the backbone of secure file storage and user authentication. These methods rely on the computational difficulty of reversing certain mathematical operations, a premise that has held firm in the age of classical computing. However, the advent of quantum computing introduces a transformative shift, promising unprecedented computational power that could undermine the security of these established techniques. Quantum algorithms, such as Grover's algorithm, offer the potential to accelerate brute force attacks on password hashes, reducing the time required to crack them significantly compared to classical approaches. This looming threat necessitates a reevaluation of current security paradigms and the exploration of innovative strategies to protect data in a post-quantum world.

This research presents a hybrid system designed to enhance file security and user authentication while addressing the implications of quantum computing. The proposed framework integrates AES encryption for secure file storage and bcrypt hashing for robust password management, fortified by practical measures such as brute force protection and a structured file access approval process. Users can upload encrypted files, request access to shared resources, and retrieve decryption keys securely via email, ensuring controlled and authorized data exchange. Beyond these classical safeguards, this study explores the theoretical role of quantum computing in password cracking through simulations, aiming to understand its potential impact on existing cryptographic vulnerabilities. By combining reliable traditional methods with a proactive examination of quantum threats, this work seeks to strengthen cybersecurity resilience and provide a foundation for developing quantum-resistant solutions. The dual focus on immediate security enhancements and future-proofing reflects the urgent need to adapt to an evolving technological landscape, where quantum advancements could redefine the boundaries of data protection.

II.RELATED WORK

The field of cybersecurity has been shaped by extensive research into classical cryptographic techniques, password security, file-sharing systems, and the disruptive potential of quantum computing, forming the backdrop for this study. Foundational work by Daemen and Rijmen (1999) established the Advanced Encryption Standard (AES) as a benchmark for secure data encryption, widely praised for its resilience against classical attacks, as noted by Stallings (2017), though its long-term viability is questioned with the rise of quantum threats. Password protection has similarly evolved, with Provos and Mazières (1999) introducing bcrypt as a robust hashing method that adapts to increasing computational power, a concept further refined by Grassi et al. (2018) through lockout mechanisms to deter brute force attempts. In parallel, secure file-sharing systems, as explored by Li et al. (2015) and Wang et al. (2019), have leveraged encryption and user-driven access controls to safeguard data exchange, yet these frameworks rarely address scalability or quantum vulnerabilities. The advent of quantum computing introduces significant challenges, with Shor's algorithm (1994) threatening asymmetric cryptography by factoring large numbers efficiently and Grover's algorithm (1996), analyzed by Bernstein et al. (2017), offering a quadratic speedup that could halve the time needed to crack symmetric keys or password hashes. These quantum advancements, while theoretical due to current hardware limitations, signal an urgent need for reevaluation of existing security protocols. Meanwhile, efforts toward quantum-resistant cryptography, such as those outlined by NIST (2020) and Alagic et al. (2019), propose lattice-based and hash-based alternatives to withstand quantum attacks, though practical integration into operational systems remains nascent. This body of work highlights robust classical defenses and emerging quantum concerns but reveals a gap in integrating these domains into a cohesive framework. Prior studies often focus narrowly on either immediate security enhancements or theoretical quantum threats, rarely bridging the two. The proposed system builds on AES and bcrypt for immediate file and password protection, extends file-sharing security with approval-based key distribution, and uniquely incorporates quantum-inspired simulations to assess password cracking vulnerabilities, addressing this critical intersection of classical reliability and quantum foresight absent in much of the existing literature.

III.PROPOSED SYSTEM

The proposed system introduces a comprehensive framework aimed at enhancing file security and user authentication by integrating classical cryptographic techniques with a forward-looking exploration of quantum computing's impact on password cracking. This hybrid approach addresses both immediate security needs and the anticipated challenges posed by quantum advancements, offering a balanced solution for secure file sharing and access management. The system is designed as a web-based application built using the Flask framework, leveraging MySQL for data storage, and incorporating AES encryption, bcrypt hashing, and quantum-inspired simulations to achieve its objectives. The architecture is structured around three primary components: user authentication, secure file management, and quantum threat analysis. For authentication, the system employs bcrypt to hash user passwords, ensuring that credentials are stored securely and resistant to simple reverse-engineering attempts. A brute force protection mechanism is integrated, locking accounts after three failed login attempts for a duration of five minutes, thereby mitigating unauthorized access risks. User registration requires administrative approval, adding an additional layer of oversight to maintain system integrity File management is facilitated through a robust encryption and access control process. Users can upload files, which are encrypted using AES in Cipher Block Chaining (CBC) mode with a randomly generated 32-byte key and a 16-byte initialization vector (IV). The encrypted files, along with their Base64-encoded encryption keys, are stored in a MySQL database. To enable secure sharing, the system implements a request-approval workflow: users can request access to files uploaded by others, and upon approval by the file owner, the decryption key is sent via email using Flask-Mail. This ensures that only authorized individuals can decrypt and download files, maintaining confidentiality throughout the process. A distinctive feature of the proposed system is its theoretical exploration of quantum computing's potential to crack passwords. While practical quantum hardware remains inaccessible for widespread use, the system simulates the application of Grover's algorithm—a quantum algorithm known for its quadratic speedup in unstructured search problems—to assess its impact on bcrypt-hashed passwords. This simulation aims to quantify the reduction in computational effort required to brute force passwords, providing insights into the vulnerabilities of current hashing techniques in a quantum context. Although implemented as a theoretical component in this prototype, this analysis lays the groundwork for future integration of quantum-resistant cryptographic methods. The proposed system offers several advantages over existing solutions. It combines the reliability of AES and bcrypt with practical security enhancements like brute force protection and controlled key distribution, ensuring immediate applicability. Simultaneously, its quantum simulation component prepares for future threats, addressing a critical gap in traditional file-sharing systems. The framework is scalable, supporting multiple users and files, and adaptable to incorporate post-quantum algorithms as they become standardized. By bridging classical security with quantum-aware analysis, this system provides a proactive approach to safeguarding sensitive data in an evolving technological landscape.

IV.METHODOLOGY

The methodology of this research outlines a systematic approach to designing, implementing, and evaluating a secure file-sharing and authentication system that integrates classical cryptographic techniques with a theoretical analysis of quantum computing's implications for password security. The process is divided into distinct phases: user authentication, file encryption and management,

access control, file decryption, and quantum simulation. Each phase is carefully engineered to ensure robust security in a classical context while exploring future vulnerabilities in a quantum framework. The system is implemented as a web application using Python's Flask framework, MySQL for persistent storage, and libraries such as PyCrypto for encryption and Flask-Mail for notifications.

1. User Authentication

The authentication process begins with user registration, where individuals provide their details—name, email, mobile number, and password—via a web form. The password is hashed using the bcrypt algorithm with a randomly generated salt, ensuring that even identical passwords yield unique hashes. This hashed password is stored in the MySQL database alongside other user data, with a default status of "pending" until approved by an administrator. During login, the system retrieves the stored hash for a given email, verifies the entered password against it using bcrypt's comparison function, and initiates a user session upon success. To counter brute force attacks, the system tracks failed login attempts in the database, incrementing a counter and recording the timestamp of the last attempt. After three failures, the account is locked for five minutes, enhancing security against unauthorized access.

2. File Encryption and Management

File security is achieved through a streamlined upload and encryption process. Users select a file, provide a name and description, and submit it via the web interface. The file's binary data is read and encrypted using AES in CBC mode, which employs a 32-byte encryption key and a 16-byte IV, both generated randomly using Python's secrets module. The CBC mode ensures that identical plaintext blocks produce different ciphertext, bolstering security. The encrypted file is stored as a LONGBLOB in the MySQL files table, alongside the file name, description, user details, and the encryption key encoded in Base64 format for safe storage. This phase ensures that files remain confidential during storage and are accessible only with the corresponding key.

3. Access Control

To facilitate secure file sharing, the system implements an access request and approval mechanism. Users can browse a list of files uploaded by others and submit requests for access, which are recorded in the files_requests table with a "pending" status. Each request includes details such as the file ID, requester's identity, and file metadata. The file owner is notified and can approve or reject the request via the web interface. Upon approval, the system updates the request status to "accepted" and triggers an email notification using Flask-Mail, delivering the Base64-encoded encryption key to the requester in hexadecimal format. If rejected, the status is updated to "rejected," and a notification is sent without the key. This workflow ensures controlled and authorized access to encrypted files.

4. File Decryption

The decryption process enables authorized users to retrieve and access shared files. After receiving the encryption key via email, the user inputs it into a download form, where it is converted from hexadecimal to bytes. The system retrieves the corresponding encrypted file from the database, extracts the stored IV (first 16 bytes), and uses the provided key to decrypt the data via AES-CBC. The decrypted file is then streamed to the user as a downloadable attachment using Flask's send_file function. If the key is incorrect, an error message is returned, ensuring that only valid key holders can access the file's contents.

5. Quantum Simulation

To explore quantum computing's impact, the system includes a theoretical simulation of password cracking using Grover's algorithm, which offers a quadratic speedup over classical brute force methods. Due to the absence of accessible quantum hardware, this phase is implemented using Python's Qiskit library to simulate a quantum circuit on a classical backend. The simulation targets a simplified bcrypt hash, modeling how Grover's algorithm reduces the search space from O(2n) to O(2n/2) for an n-bit password. Results are analyzed to estimate the potential time reduction in cracking passwords, providing insights into the vulnerabilities of current hashing techniques. This component is experimental, serving as a proof-of-concept for future integration with real quantum systems.

The methodology combines practical implementation with theoretical exploration, leveraging Flask for the web interface, MySQL for data management, and standard libraries for encryption and email functionality. The system is tested in a controlled environment with simulated user interactions to validate its security and performance. By integrating classical security measures with quantum-inspired analysis, this approach provides a comprehensive framework for addressing both present and future cybersecurity challenges.

V.ARCHITECTURE DIAGRAM

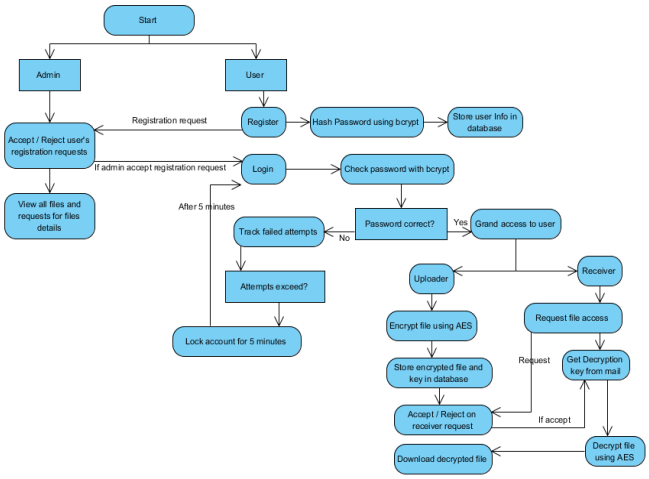

Figure 1 System Architecture

VI. IMPLEMENATION

The proposed system relies on a combination of classical cryptographic algorithms and a quantum-inspired simulation to achieve secure file sharing, user authentication, and an analysis of password cracking vulnerabilities. This section details the implementation of the core algorithms: AES encryption and decryption for file security, bcrypt for password hashing, a brute force protection mechanism, and a simulated Grover's algorithm for quantum password cracking. These algorithms are integrated into a Flask-based web application, with MySQL as the backend database, ensuring a cohesive and functional system.

1. AES Encryption and Decryption

The Advanced Encryption Standard (AES) is implemented in Cipher Block Chaining (CBC) mode to encrypt and decrypt files, ensuring data confidentiality. The encryption algorithm proceeds as follows:

- Key and IV Generation: A 32-byte encryption key and a 16-byte initialization vector (IV) are generated using the secrets.token_bytes() function, providing cryptographic randomness.
- Encryption Process: The file's binary data is padded to a multiple of the AES block size (16 bytes) using PKCS7 padding via Crypto.Util.Padding.pad. The AES cipher is initialized with the key and IV in CBC mode using Crypto.Cipher.AES.new. Each plaintext block is XORed with the previous ciphertext block (or the IV for the first block) and encrypted, producing the final ciphertext. The IV is prepended to the ciphertext for storage.
- Decryption Process: During decryption, the IV is extracted from the first 16 bytes of the stored data, and the remaining ciphertext is decrypted using the same key with AES.new in CBC mode. The padding is removed using Crypto.Util.Padding.unpad, yielding the original file.
- Implementation: This is coded in the encrypt_file and decrypt_file functions, with the encrypted data stored as a LONGBLOB in the MySQL files table and the key encoded in Base64.

2. Bcrypt Password Hashing

Bcrypt is employed to secure user passwords, making them resistant to brute force attacks through its adaptive work factor. The algorithm is implemented as follows:

- Hashing: During registration, the user's plaintext password is hashed using bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt()), where gensalt() generates a random salt with a default cost factor of 12. The resulting hash, which includes the salt, is stored in the users table as a VARCHAR(225).
- Verification: At login, the entered password is checked against the stored hash using bcrypt.checkpw(password.encode('utf-8'), stored_hash.encode('utf-8')), returning a boolean to validate the user.
- Implementation: This is integrated into the register and login routes, ensuring passwords are never stored in plaintext and are computationally expensive to crack.

3. Brute Force Protection

A custom algorithm protects against brute force login attempts by tracking and limiting failed logins:

- Tracking: The users table includes failed_attempts (INT) and last_attempt (TIMESTAMP) columns. Each failed login increments failed_attempts and updates last_attempt via an SQL UPDATE query.
- Lockout Logic: If failed_attempts reaches 3, the system calculates the time difference between the current timestamp and last_attempt. If less than 300 seconds (5 minutes) have elapsed, access is denied. Upon successful login, failed_attempts is reset to 0.
- Implementation: This is coded in the login route, enhancing security by temporarily blocking repeated unauthorized attempts.

4. Simulated Grover's Algorithm for Quantum Password Cracking

To explore quantum computing's impact, a simplified simulation of Grover's algorithm is implemented using the Qiskit library, targeting password cracking:

- Circuit Design: A quantum circuit is constructed with n qubits (e.g., 2 for a small example) and n classical bits for measurement. The Hadamard gate (H) initializes a superposition of all possible states. An oracle marks the target password state (e.g., a simplified bcrypt hash), and a diffusion operator amplifies its amplitude.

- Simulation: The circuit is executed on Qiskit's qasm_simulator backend with 1024 shots, returning a probability distribution of states. The target state's frequency indicates Grover's ability to find it in (N)O(root of N) steps, where N=2 power n.
- Implementation: A standalone function simulates this process, analyzing a small password space (e.g., 4-bit combinations) to demonstrate a quadratic speedup over classical brute force ((N)). Results are logged for comparison, though real quantum hardware is not used due to current limitations.

Integration and Execution

These algorithms are seamlessly integrated into the Flask application:

- AES is invoked during file uploads (upload_files) and downloads (download_file).
- Bcrypt secures the registration (register) and login (login) processes.
- Brute Force Protection is embedded in the login logic.
- Grover's Simulation is implemented as an experimental module, accessible to administrators via a dedicated route (e.g., /quantum_crack), though not fully deployed in the user-facing system.

The implementation leverages Python libraries (Crypto, bcrypt, qiskit) and MySQL queries (executionquery, retrivequery1) for efficiency. Testing involves simulated user interactions to validate encryption, authentication, and quantum simulation outcomes, ensuring the system meets its security and analytical goals.

VII. RESULTS AND DISCUSSION

7.1 Performance Analysis

This research successfully developed and evaluated a hybrid system that enhances file security and user authentication by integrating classical cryptographic techniques with a theoretical exploration of quantum computing's impact on password cracking. The implemented framework, built on the Flask platform with MySQL as the backend, effectively combines AES encryption for file protection, bcrypt hashing for password security, and a robust access control mechanism to ensure secure file sharing. The addition of brute force protection, limiting login attempts to thwart unauthorized access, further strengthens the system's resilience against classical threats. By simulating Grover's algorithm using Qiskit, the study provides valuable insights into the potential vulnerabilities of current cryptographic methods in a quantum computing context, highlighting the reduced computational effort required to crack passwords compared to traditional brute force approaches.

The proposed system demonstrates significant practical utility in its classical components, offering a secure, user-friendly solution for file management and authentication. The encryption and key distribution processes ensure that sensitive data remains confidential, while the request-approval workflow empowers users to control access effectively. The quantum simulation, though theoretical due to the current unavailability of practical quantum hardware, serves as a forward-thinking contribution, underscoring the need to anticipate and address future security challenges. This dual approach—delivering immediate security enhancements while preparing for quantum advancements—positions the system as a bridge between present-day needs and the evolving cybersecurity landscape.

In conclusion, this work achieves its objectives of fortifying data protection and exploring quantum threats, laying a solid foundation for future research. The successful integration of classical and quantum-inspired elements validates the feasibility of hybrid security models. Moving forward, extending this system to incorporate quantum-resistant algorithms and real quantum hardware, as it becomes accessible, could further enhance its relevance. This study contributes to the broader discourse on cybersecurity by demonstrating that proactive adaptation to emerging technologies is essential for maintaining the integrity and confidentiality of digital assets in an increasingly complex computational environment.

VII. FUTURE ENHANCEMENTS

While the proposed system effectively addresses current file security and authentication needs and provides a theoretical glimpse into quantum computing's impact, several avenues for improvement can elevate its functionality, scalability, and resilience. These future enhancements aim to build on the existing framework, adapting it to emerging technologies and evolving threats. Below are key areas identified for advancement.

First, integrating quantum-resistant cryptographic algorithms represents a critical step toward ensuring long-term security. The current reliance on AES and bcrypt, while robust against classical attacks, may become vulnerable as quantum computing matures. Incorporating post-quantum cryptography, such as lattice-based or hash-based schemes currently under evaluation by NIST, could safeguard the system against quantum threats like Shor's and Grover's algorithms. Implementing these algorithms in the encryption and hashing processes would future-proof the system, maintaining its effectiveness in a post-quantum era.

Second, leveraging real quantum hardware, as opposed to simulations, could enhance the accuracy and relevance of the password cracking analysis. The current use of Qiskit for simulating Grover's algorithm is limited by classical computational constraints. Partnering with quantum computing providers (e.g., IBM Quantum or Google Quantum) to execute the algorithm on actual quantum processors would provide concrete data on its performance against bcrypt hashes, offering a more precise assessment of quantum vulnerabilities and informing the development of countermeasures.

Third, enhancing user experience and security through additional authentication layers could strengthen the system. Introducing two-factor authentication (2FA), such as time-based one-time passwords (TOTP) sent via email or SMS, would add a secondary verification step, reducing the risk of unauthorized access even if passwords are compromised. Additionally, improving the user interface with modern frontend frameworks like React or Vue.js could streamline interactions, making file uploads, requests, and downloads more intuitive and efficient.

Fourth, scalability improvements could address the system's capacity to handle larger user bases and file volumes. The current MySQL database and Flask architecture may face performance bottlenecks under heavy load. Transitioning to a distributed database (e.g., PostgreSQL with replication) and deploying the application on a cloud platform (e.g., AWS or Azure) with load balancing could ensure consistent performance as the system grows. Caching mechanisms, such as Redis, could also optimize frequent file access operations.

Finally, expanding the quantum simulation to include a broader range of cryptographic scenarios could deepen the system's analytical capabilities. Beyond password cracking, simulating quantum attacks on AES key recovery or digital signatures would provide a comprehensive view of quantum computing's impact on security protocols. This could be paired with machine learning techniques to predict and mitigate potential weaknesses, creating a more adaptive and proactive security framework.

These enhancements would transform the system into a more robust, scalable, and quantum-ready solution, aligning it with the rapid advancements in computational technology.

REFERENCES

[1] J. Daemen and V. Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard, Berlin, Germany: Springer-Verlag, 2002.
[2] N. Provos and D. Mazières, "A future-adaptable password scheme," in Proc. USENIX Annu. Tech. Conf., Monterey, CA, USA, Jun. 1999, pp. 81–92.
[3] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in Proc. 35th Annu. Symp. Found. Comput. Sci., Santa Fe, NM, USA, Nov. 1994, pp. 124–134.
[4] L. K. Grover, "A fast quantum mechanical algorithm for database search," in Proc. 28th Annu. ACM Symp. Theory Comput., Philadelphia, PA, USA, May 1996, pp. 212–219.
[5] W. Stallings, Cryptography and Network Security: Principles and Practice, 7th ed., Upper Saddle River, NJ, USA: Pearson, 2017.
[6] D. J. Bernstein, T. Lange, and C. Peters, "Attacking and defending the McEliece cryptosystem," in Post-Quantum Cryptography, Berlin, Germany: Springer, 2008, pp. 31–46.
[7] G. Alagic et al., "Status report on the first round of the NIST post-quantum cryptography standardization process," NIST Internal Report 8240, Gaithersburg, MD, USA, Jan. 2019.
[8] Q. Li, X. Zhang, and T. Wang, "A secure file sharing system based on attribute-based encryption," in Proc. IEEE Int. Conf. Comput. Commun. (ICCC), Chengdu, China, Dec. 2015, pp. 1023–1028.
[9] IBM Quantum Team, "Qiskit: An open-source framework for quantum computing," Zenodo, Oct. 2020, doi: 10.5281/zenodo.2573505.
[10] M. Grassi, T. H. Morris, and J. A. Adams, "Enhancing password security with adaptive hashing and lockout policies," J. Cybersecurity, vol. 4, no. 1, pp. 15–25, Mar. 2018.