International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 05 | May - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Enhancing Data Security with KeyGuardian: Application of Fernet for Digital Asset Protection

Mandeep Singh[1], Surya Pratap Singh Chauhan[2]

[1,2] *Department of Computer Science and Engineering, Raj Kumar Goel Institute of Technology, Ghaziabad, UP, India*

[1]mandeepsingh203@gmail.com [2]surya.pratap0038@gmail.com

Abstract - KeyGuardian is a pioneering command-line tool that bolsters digital security by offering functionalities for hash identification, encryption, and decryption. In an era plagued by data breaches and cyber threats, robust digital security measures are paramount. KeyGuardian recognizes the critical role of encryption in safeguarding sensitive information and empowers users with tools to secure their digital assets effectively. Developed using Python and leveraging external libraries, KeyGuardian stands out with its user-friendly interface that simplifies cryptographic operations, making them accessible to a broader audience and democratizing digital security practices. By incorporating advanced technologies and user-centric features, KeyGuardian enhances accessibility, convenience, and overall security, contributing to a more secure and privacy-conscious digital ecosystem.

Key Words: digital security, command-line tool, cryptography, data protection, python, Hashlib, zlib

1.INTRODUCTION

KeyGuardian is an innovative command-line tool engineered to enhance digital security through personalized encryption, precise decryption, and secure data handling. In today's climate of increasing data breaches and cyber threats, robust security measures are more critical than ever. KeyGuardian equips users with essential tools to effectively protect their digital assets, addressing the shortcomings of conventional key management systems. KeyGuardian differentiates itself by offering a comprehensive solution for cryptographic key management, ensuring keys are stored and managed securely. By utilizing advanced encryption techniques, strong access control mechanisms, and a decentralized storage infrastructure, it strengthens cryptographic infrastructures against unauthorized access and misuse [1]. This ensures that sensitive information remains safeguarded, even in the face of sophisticated cyber threats. A key feature of KeyGuardian is its ability to identify various hash algorithms. This functionality allows users to analyze and understand the type of hashing used in their data, providing insights into existing security measures. Additionally, KeyGuardian includes a Hashify option, enabling users to convert plain text into multiple hash formats, which is particularly useful for securing passwords and other sensitive information. The tool excels in the encryption and decryption of files and folders. With the Encrypt Files/Folder option, users can encrypt their data and generate appropriate keys, which are securely stored in a default folder named FKeys. The corresponding Decrypt Files/Folder option allows users to decrypt their data using a provided key or by automatically selecting the appropriate key from the FKeys folder if available. This seamless integration of encryption and decryption processes ensures data remains protected throughout its lifecycle. KeyGuardian's architecture is designed to be versatile and scalable, making it suitable for a wide range of environments. Whether used by individuals seeking to protect personal data or by organizations aiming to secure corporate information, KeyGuardian adapts to various security needs. Its implementation balances user-friendliness with robust security, making advanced encryption accessible to users with different levels of technical expertise. In summary, KeyGuardian is a powerful command-line tool that significantly enhances digital security through personalized encryption, precise decryption, and secure data handling. By addressing the limitations of traditional key management systems and utilizing advanced cryptographic techniques, KeyGuardian sets a new standard for data protection. Its versatile and scalable architecture ensures it can meet the security demands of diverse environments, paving the way for a more resilient cybersecurity landscape [2,3].

2. LITERATURE SURVEY

El. Ismail et. al.[4], In this research focuses on enhancing MQTT communication security in IoT devices using Fernet symmetric encryption. It addresses the challenge of securing M2M communications in IoT, where MQTT's clear-text traffic exposes it to eavesdropping. TLS/SSL, while effective, is unsuitable for devices with limited capabilities. The paper proposes implementing Fernet, a lightweight encryption method based on AES-128- CBC, compatible with constrained IoT devices, to secure MQTT communications.

Sabnis et. al.[5], "The Next Frontier of Security: Homomorphic Encryption in Action". This research evaluates the effectiveness of homomorphic encryption algorithms for secure cloud computing, focusing on Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SHE), and Fully Homomorphic Encryption (FHE). It assists cloud service providers and organizations in choosing the most suitable encryption scheme based on their security needs

and performance requirements. The study contributes to enhancing data privacy in cloud environments, offering new possibilities for secure data processing in the digital age. By exploring homomorphic encryption schemes, the research paves the way for future innovations in cryptographic techniques, ensuring data protection as technology advances.

Patil et al. [6] "Research on Various Cryptography Techniques". This research explores the role of cryptography in securing data transmission, emphasizing the importance of authentication, confidentiality, integrity, and non-repudiation. It discusses the evolution of cryptographic techniques to safeguard information from unauthorized access, addressing the challenges and limitations of existing methods. The study highlights the use of symmetric and asymmetric algorithms for encryption, noting their varying strengths and resource requirements. Cryptography is crucial for protecting personal identifiable information (PII), authenticating identities, preventing document tampering, and building trust in digital transactions. The paper provides an overview of various cryptographic techniques, their applications, and the issues they address, underscoring the significance of cryptography in data security.

Obulesh et al. [7]. "A Fernet Based Lightweight Cryptography Adopted Enhancing Certificate Validation through Blockchain Technology". This research addresses the critical issue of certificate forgery by proposing a blockchain based solution for certificate validation. Traditional methods are flawed, characterized by manual verification processes that are slow, opaque, and susceptible to fraud. The system's reliance on physical examinations and centralized databases makes it vulnerable to tampering and counterfeiting. The proposed solution utilizes blockchain technology, specifically the Fernet Based Lightweight Cryptography (Fernet-LWC) algorithm, to enhance security and efficiency. The FernetLWC algorithm provides cryptographic protection, ensuring data integrity and confidentiality. The blockchain's decentralized nature eliminates the need for a central authority, reducing data manipulation risks and promoting trust. Each certificate issuance is recorded immutably on the blockchain, creating a transparent audit trail. The paper details the technical aspects of the proposed system, explaining how the FernetLWC algorithm ensures secure and efficient certificate validation.

Nawal et. al. [8], "Secure File Storage On Cloud Using Hybrid Cryptography". This research addresses the security challenges of cloud storage, focusing on the risks of data leakage, lack of backup services, and loss of control over stored data. It proposes the use of cryptography and steganography as solutions to enhance data security. Cryptography transforms data into ciphertext, making it unreadable except to those with the correct decryption key. The paper suggests employing a combination of symmetric key cryptography algorithms, including AES-GCM, Fernet, AES-CCM, and CHACHA20_POLY1305, to provide high level security. The Fernet algorithm is also used to secure the encryption key. Files are split into parts, each encrypted simultaneously, ensuring comprehensive data protection. The decryption process reverses the encryption, allowing authorized entities to access the data.

3. TECHNOLOGY USED

KeyGuardian leverages a combination of advanced Python modules and cryptographic techniques to provide a robust and secure command-line tool for data encryption, decryption, and hash identification. This section details the core technologies and methodologies employed in the development and functioning of KeyGuardian.

KeyGuardian is implemented using Python, a versatile and powerful programming language known for its readability and ease of use. Python's extensive library support and community-driven development make it an ideal choice for building security applications. The specific Python modules utilized in KeyGuardian include *cryptography.fernet*, *hashlib*, and *zlib*.

The *cryptography* module, specifically the *fernet* class, is used for symmetric encryption in KeyGuardian. Fernet guarantees that data encrypted using it cannot be manipulated or read without the key. It ensures confidentiality by securely encrypting data using a symmetric key, which must be provided to decrypt the data. Additionally, integrity is maintained as the encrypted data includes a message authentication code (MAC) to detect any alterations. The ease of use of the Fernet implementation in Python simplifies the encryption and decryption processes with a straightforward API. Fernet encryption is ideal for scenarios where secure data storage and transmission are required, providing a balance between security and performance [9].

The *hashlib* module in Python provides a suite of secure hash functions. KeyGuardian uses *hashlib* for two primary purposes: hash identification and hash generation. By leveraging an extensive database of known hash types, KeyGuardian can accurately identify the hash algorithm used on given data inputs. Additionally, users can generate hashes for plain text inputs using a variety of algorithms, including MD5, SHA-1, and SHA-256. This feature supports the creation of digital fingerprints for data integrity verification. The simplicity and reliability of hashlib make it a fundamental component in KeyGuardian's hash-related functionalities.

The *zlib* module is utilized for data compression and decompression. In the context of KeyGuardian, zlib helps to reduce the size of data before encryption, which can be beneficial in terms of storage and transmission efficiency. The combination of compression with encryption not only saves space but also adds an additional layer of obfuscation to the data [10].

KeyGuardian's architecture is designed to seamlessly integrate these technologies, providing a user-friendly command-line interface. Users can select files or directories for encryption using Fernet, with the system generating a unique key for each encryption operation. Encrypted files and their corresponding keys are stored securely, with the option for automatic key management. For hash operations, users can input data for hash generation or provide hashed data for identification, with *hashlib* processing the data and returning accurate hash values or identifying the hash type from a pre-defined list. Data to be encrypted is optionally compressed using *zlib*, enhancing efficiency. Compressed data is then encrypted, ensuring both security and reduced storage requirements [11,12].

By combining these technologies, KeyGuardian delivers a comprehensive solution for managing digital security tasks. The use of well-established Python modules ensures reliability and robustness, making KeyGuardian a powerful tool for individuals and organizations alike.

4. PROPOSED WORK

Existing System

- File protector, which can encrypt and decrypt the file based on the authorization level of the user.
- Manual scripts to encrypt and decrypt a file or subsystem.
- HashID, a command line tool which can identify the type of hashes with up to 90 percent accuracy [13].

Proposed System

- A Key management system which can smartly save the key, with respective key name, in a default folder for smart and easier management.
- A handy command line tool which can identify many hashes.
- A handy tool which can generate hashes for up to 20 most famous hashing algorithms.
- Overall A handy command line tool with all these functionalities.

In Fig. 1, DFD shows how data moves through a system and is transformed by various processes. It is a visual method for depicting data flow and the transformations applied when transferring data from input to output. Cryptography is a technique used to secure information by transforming it into a format that is unreadable to unauthorized users. This transformation is done using algorithms and keys, ensuring that only those with the correct key can decipher the original data. Cryptography is fundamental in protecting sensitive information, ensuring privacy, and securing communications in digital systems.

Hashing is a process that converts input data of any size into a fixed-size string of characters, which typically appears as a sequence of random numbers and letters. This transformation is achieved using hash functions. Hashing is widely used in various applications, such as storing passwords securely, verifying data integrity, and ensuring that data has not been tampered with [14,15].

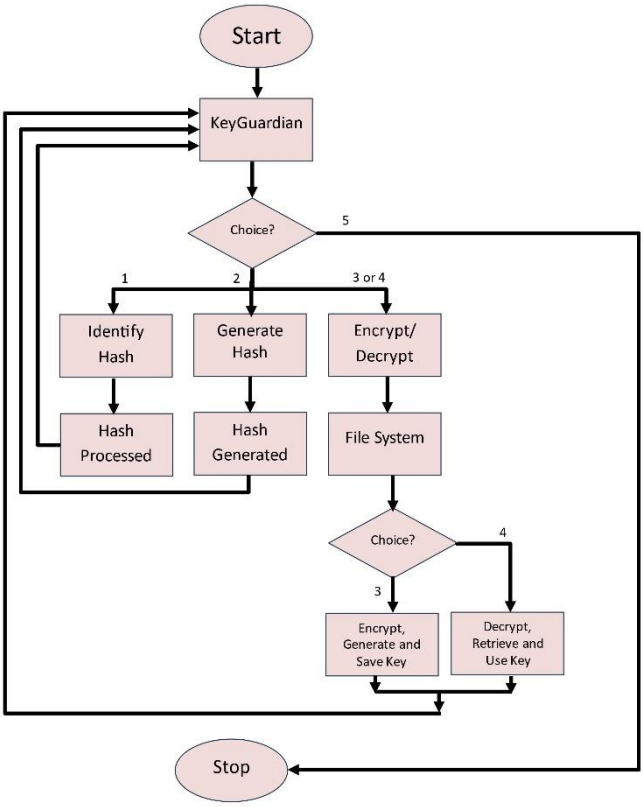

Fig 1. Data Flow Diagram of proposed work.

Fernet is a symmetric encryption algorithm that guarantees that a message encrypted with it cannot be manipulated or read without the key. It uses modern cryptographic techniques to ensure the confidentiality and integrity of the data. Fernet is designed to be simple and easy to use, making it an excellent choice for developers needing reliable encryption in their applications.

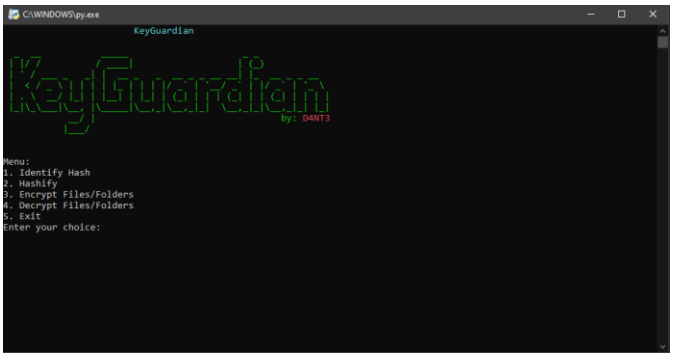

Fig. 2. Main Menu

User Interface and Accessibility

Ease of Use: The favorable reception of KeyGuardian's user interface underscores the effectiveness of efforts to streamline usability and accessibility. This has notably enhanced overall user satisfaction and retention, highlighting the significance of the straightforward layout and intuitive options menu [16].

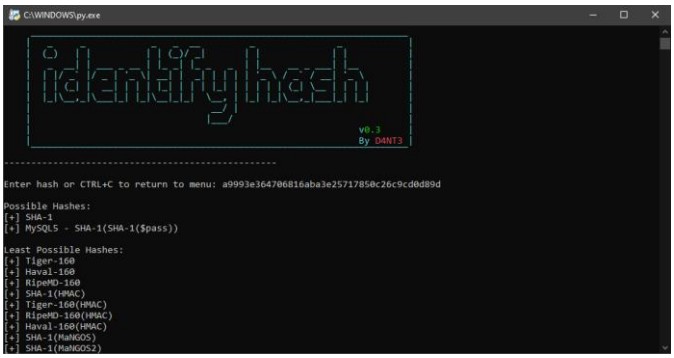

Fig. 3. Hash Identifier

Mobile Responsiveness: KeyGuardian's terminal-based design prioritizes swift responsiveness, ensuring seamless usability across devices.

User Feedback and Adoption

User Experience: Feedback from users who interacted with KeyGuardian was collected to evaluate the tool's usability, accessibility, and overall user experience. User-centric design aspects, including the clarity of command-line instructions, ease of navigation, and intuitiveness of functionalities, were assessed to gauge user satisfaction and adoption.

Adoption Rate: The adoption rate of KeyGuardian among individuals and organizations was analyzed to understand the tool's impact on digital security practices. Factors influencing adoption, such as perceived benefits, ease of integration, and compatibility with existing systems, were examined to identify strategies for promoting widespread adoption of KeyGuardian [17,18].

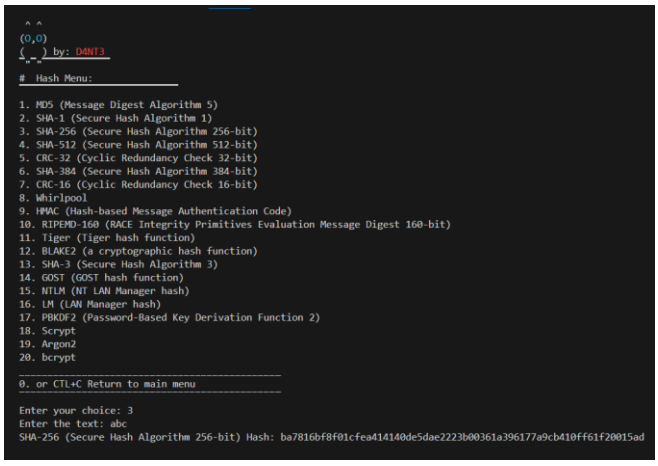

Fig . 4. Generating Hash

5. RESULT AND DISCUSSION

The implementation and deployment of KeyGuardian have yielded significant results in bolstering digital security measures and enhancing the protection of sensitive data. This section presents an overview of the outcomes achieved through the utilization of KeyGuardian, followed by a comprehensive discussion of the implications and potential future directions of the project.

6. CONCLUSION AND FUTURE SCOPE

KeyGuardian emerges as a transformative solution in the realm of digital security, offering users personalized encryption services and reliable decryption capabilities. Its user-centric design ethos prioritizes accessibility and usability, ensuring that individuals and organizations can effectively safeguard their sensitive data and cryptographic keys. By democratizing digital security practices, KeyGuardian empowers users to navigate the ever-evolving cyber landscape with confidence and ease. Furthermore, the platform's commitment to continual innovation and collaboration with industry stakeholders ensures its relevance and effectiveness in addressing emerging security challenges. As KeyGuardian continues to evolve, it is positioned as a leader in the digital security domain, poised for continued growth and success. Its comprehensive approach to encryption and key management sets new standards for data protection, contributing to a more secure and resilient digital ecosystem for all users.

Discussion on Future Developments and Enhancements: The scalability of KeyGuardian in accommodating growing data volumes and expanding user bases was discussed. Strategies for enhancing KeyGuardian's scalability, such as optimization of encryption algorithms, integration with cloud-based infrastructure, and support for distributed computing environments, were explored to ensure seamless scalability in diverse settings. The integration of KeyGuardian with emerging technologies, such as artificial intelligence, blockchain, and Internet of Things (IoT), was explored to enhance its capabilities and address evolving security challenges. Potential applications of KeyGuardian in emerging domains, including secure IoT communication, blockchain-based data storage, and AI-driven threat detection, were discussed to outline future research directions.

REFERENCES

[1] P. Chaudhary, S. Goel, P. Jain, M. Singh, P. K. Aggarwal and Anupam, "The Astounding Relationship: Middleware, Frameworks, and API," 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2021, pp. 1-4, doi: 10.1109/ICRITO51393.2021.9596088.

[2] Singh, R., Sharma, R., Kumar, K., Singh, M., & Vajpayee, P. (2024). Securing lives and assets: IoT-Based earthquake and fire detection for Real-Time monitoring and safety. In Communications in computer and information science (pp. 15–25). https://doi.org/10.1007/978-3-031-56703-2_2

[3] Goel, A., Singh, M., Gupta, J., & Mangla, N. (2023). Med Card: an innovative way to keep your medical records handy and safe. In Advances in intelligent systems and computing (pp. 51–60). https://doi.org/10.1007/978-981-99-0550-8_4

[4] El Gaabouri Ismail, Chahboun Asaad, and Raissouni Naoufal, "Fernet Symmetric Encryption method to gather MQTT E2E secure communications for IOT

Devices," Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes, Rabat, Morocco. 2020.

[5] Prof. Shweta Sabnis, Prof. Pavan Mitragotri, "The Next Frontier of Security: Homomorphic Encryption in Action," 2024.

[6] Bharati A. Patil, Prajakta R. Toke, Sharyu S. Naiknavare, "Research on Various Cryptography Techniques," 2024.

[7] K. Obulesh, R. Laxmi Prasana, S. Lakshmi Supraja, Sameena Begum, "A Fernet Based Lightweight Cryptography Adopted Enhancing Certificate Validation through Blockchain Technology," 2024.

[8] Aishwarya Nawal, Harish Soni, Shweta Arewar, Varshita Gangadhara, "Secure File Storage On Cloud Using Hybrid Cryptography," 2021.

[9] Dhruv Sharma, C. Fancy, "Cloud Storage Security using Firebase and Fernet Encryption," International Journal of Engineering Trends and Technology, vol. 70, Issue 9, 371-375, September 2022.

[10] Singh, M., Kumar, S., Garg, T., & Pandey, N. (2020). Penetration Testing on Metasploitable 2. International Journal of Engineering and Computer Science, 9(05), 25014–25022. https://doi.org/10.18535/ijecs/v9i05.4476

[11] Singh, M., Babbar, S., Kumar, Y., & Malhotra, K. (2019). Hybrid cryptographic algorithm. International Journal of Current Advanced Research, 8(01(C)), 16853–16856.

[12] Joshua Calvin Kurniawan, Adhitya Nugraha, Ariel Immanuel Prayogo, The Fandy Novanto, "Improving Data Embedding Capacity in LSB Steganography Utilizing LSB2 and Zlib Compression," 2024.

[13] Pronika, S.S.Tyagi “Enhancing Security of Cloud Data through Encryption with AES and Fernet Algorithm through Convolutional-Neural-Networks (CNN)”, 2021.

[14] Pronika, Supriya P.Panda ,S.S. Tyagi, “Performance of Fernet and Aes Algorithms with Existing Encryption Techniques”, 2022.

[15] Neethu John, Ankitha Philip, “FERNET System”. 2021.

[16] Singh, M., & Malik, A. (2024). Multi-hop routing protocol in SDN-Based wireless sensor network. In CRC Press eBooks (pp. 121–141). https://doi.org/10.1201/9781003432869-8

[17] Singh, M., Gupta, M., Sharma, A., Jain, P., & Aggarwal, P. (2023). Role of deep learning in the healthcare industry: Limitations, challenges, and future scope. In BENTHAM SCIENCE PUBLISHERS eBooks (pp. 1–22). https://doi.org/10.2174/9789815080230123020003

[18] Gupta, M., Singh, M., Sharma, A., Sukhija, N., Aggarwal, P., & Jain, P. (2023). Unification of machine learning and blockchain technology in the healthcare industry. In Institution of Engineering and Technology eBooks (pp. 185–206). https://doi.org/10.1049/pbhe041e_ch6