Indian Journal of Computer Science and Technology
https://www.doi.org/10.59256/indjcst.20250401028
Volume 4, Issue1 (January-April 2025), PP: 188-192.
www.indjcst.com
ISSN No: 2583-5300

Ransomware Shield Detection and Prevention System

Lathika M[1], Rajadhurai [2]
[1]*M.Sc. CFIS, Department of Computer Science Engineering, Dr. MGR Educational and Research Institute, Chennai, Tamilnadu, India.*
[2]*Assistant Professor, Centre of Excellence in Digital Forensics, Chennai, Tamilnadu, India.*

To Cite this Article: *Lathika M[1], Rajadhurai [2], "Ransomware Shield Detection and Prevention System", Indian Journal of Computer Science and Technology, Volume 04, Issue 01 (January-April 2025), PP: 188-192.*

Abstract: *The "Ransomware Shield Detection and Prevention System" is a Flask-based solution designed to combat ransomware by securing data with a multi-layered encryption approach, integrating AES-256, RSA, and simpler ciphers like XOR and Vigenère. It addresses the rising ransomware threat that locks data and demands payment, often bypassing traditional defenses. The study investigates whether this encryption framework can prevent data loss and detect unauthorized encryption attempts. Using a web application, it encrypts data and explores adding real-time monitoring for ransomware behavior. While effective at securing data, the system lacks active detection, highlighting a need for enhanced prevention features to fully counter ransomware.*

Keywords: *Ransomware, Encryption, AES-256, RSA, Flask, Cybersecurity, Data Protection, Vigenere Cipher, Detection, Prevention.*

I.INTRODUCTION

The "Ransomware Shield Detection and Prevention System" addresses the growing threat of ransomware, which encrypts victims' data and demands payment for access. Built on a Flask-based web platform, this system employs a hybrid encryption approach, combining advanced algorithms like AES-256 and RSA with simpler ciphers such as XOR and Vigenère to secure sensitive information. As ransomware evolves, traditional defenses like antivirus software often fall short against sophisticated attacks, making proactive data protection crucial. This introduction outlines the system's purpose: to explore how multi-layered encryption can prevent data loss and potentially detect ransomware activity, offering a foundation for enhanced cybersecurity in an increasingly hostile digital landscape [1].

Ransomware has emerged as a significant cybersecurity threat, with attacks escalating in frequency and sophistication since the early 2000s, targeting individuals, businesses, and critical infrastructure. These malicious programs encrypt files using strong cryptographic techniques, rendering data inaccessible until a ransom is paid—often with no guarantee of recovery. Traditional countermeasures, such as signature-based antivirus tools and backups, have proven inadequate against evolving ransomware strains, particularly zero-day variants that exploit unpatched vulnerabilities. The "Ransomware Shield Detection and Prevention System" builds on this backdrop, leveraging encryption as both a shield and a potential detection mechanism to counter the growing menace of data extortion in the digital age [2].

The "Ransomware Shield Detection and Prevention System" holds significant value in the fight against ransomware, a pervasive threat that compromises data integrity and costs billions annually. By implementing a multi-layered encryption framework, it offers a proactive means to safeguard sensitive information, reducing the leverage attackers gain through encryption-based extortion. Its significance lies in shifting the focus from reactive recovery to preemptive protection, potentially minimizing downtime and financial losses for individuals and organizations. Furthermore, exploring detection capabilities within this system could bridge a critical gap in current cybersecurity strategies, enhancing resilience against evolving ransomware tactics [3].

The scope of the "Ransomware Shield Detection and Prevention System" encompasses the development and evaluation of a Flask-based web application that employs multi-layered encryption to protect data from ransomware attacks. It focuses on integrating robust algorithms like AES-256 and RSA with simpler ciphers to secure files, targeting small-scale systems or individual users rather than large enterprise networks. The project also explores the feasibility of adding real-time detection of unauthorized encryption attempts, though it currently emphasizes prevention through encryption. Future expansion could include broader system compatibility, advanced monitoring, and scalability to address diverse ransomware threats across various platforms [4].

The "Ransomware Shield Detection and Prevention System" investigates key research questions: Can a multi-layered encryption approach effectively prevent ransomware from locking data, and how can it detect unauthorized encryption in real time? Existing systems, such as antivirus software (e.g., Norton, McAfee) and backup solutions, rely heavily on signature-based detection and post-attack recovery, often failing against zero-day ransomware variants. Methods like CryptoLock monitor file system changes for detection, while others use decoy files or behavioral analysis. In contrast, this system prioritizes preemptive encryption with AES-256 and RSA, aiming to outpace attackers, though it currently lacks the dynamic detection features of some established tools [5].

II.LITERATURE REVIEW

Berrueta et al. [6] had propose a ransomware detection framework using honeypot files and machine learning to identify malicious encryption patterns. Their system deploys decoy files to lure attackers, triggering alerts upon modification attempts. This proactive approach contrasts with the Shield's encryption focus, which lacks such bait mechanisms. The study reports high detection rates, suggesting a potential enhancement for the Shield. However, integrating honeypots could increase complexity and resource demands. The Shield could benefit from this by adding a lightweight decoy layer. This highlights a gap in active defense within the current design.

Kok et al. [7] had analyzed ransomware mitigation using blockchain for secure key storage and recovery. Their method ensures decryption keys remain tamper-proof, addressing post-attack recovery needs. The Shield's static RSA keys could adopt this decentralized approach for enhanced security. Blockchain integration could also log encryption attempts immutably, aiding detection. However, it introduces latency and infrastructure costs unsuitable for real-time use. The study's focus on recovery complements the Shield's preventive encryption. Combining these could create a more resilient system.

Hassan et al. [8] had explored ransomware detection via network traffic analysis using deep learning. They identify encrypted command-and-control communications typical of ransomware. This network-centric approach differs from the Shield's local encryption strategy. Their model achieves over 90% accuracy, suggesting a robust detection layer. Adding network monitoring to the Shield could catch ransomware before it encrypts data. However, it requires constant connectivity, limiting offline applicability. This underscores the Shield's need for broader threat detection capabilities.

Continella et al. [9] had introduced ShieldFS, a file system shield that detects ransomware via I/O monitoring. It tracks file access patterns, flagging rapid encryption as malicious with high precision. Unlike the Shield's encryption-only focus, ShieldFS emphasizes real-time behavioral analysis. The study's success suggests integrating I/O monitoring into the Flask app. This could proactively stop ransomware, enhancing the current system's scope. However, it demands significant system-level integration beyond the web app. This reveals a key limitation in the Shield's passive approach.

Mehnaz et al. [10] had proposed a ransomware prevention system using cryptographic key traps. They generate fake keys to mislead attackers, delaying encryption processes. This contrasts with the Shield's straightforward encryption, which lacks deception tactics. Their method slows attacks, giving users time to respond, with minimal overhead. Adding key traps to the Shield could bolster its preventive capabilities. However, it risks confusing legitimate users if not carefully implemented. This offers a creative angle for future Shield enhancements.

Lee et al. [11] had investigated ransomware countermeasures using hardware-assisted memory encryption. Their approach leverages CPU features to encrypt memory, thwarting ransomware at the source. The Shield's software-based encryption could integrate such hardware support for efficiency. The study shows reduced performance overhead, ideal for real-time protection. However, it requires specific hardware, limiting universal deployment. This suggests a high-security upgrade path for the Shield in specialized contexts. It highlights the system's potential evolution beyond software solutions.

Zhang et al. [12] had developed a ransomware detection system using dynamic analysis of process behavior. They monitor runtime activities like API calls, catching ransomware before encryption completes. This proactive stance differs from the Shield's reliance on preemptive data encryption. Their system achieves low false positives, offering a reliable detection model. Integrating process monitoring could make the Shield more responsive to threats. However, it increases computational load, a trade-off for lightweight designs. This emphasizes the need for a balanced detection-prevention strategy.

III.PROPOSED METHODOLOGY

The Ransomware Shield Detection and Prevention System is designed to detect and prevent ransomware attacks in real time using deep learning-based behavioral analysis. The methodology involves multiple stages, including research design, data collection, preprocessing, model selection, and system architecture, to ensure an effective detection mechanism [13].

System Architecture

This architecture diagram represents a Ransomware Detection and Prevention System, designed to monitor, detect, and respond to potential ransomware threats. The system follows a structured workflow to identify suspicious file and user activity, alert the user, and take action if necessary.

1. Flask Application:

The system begins with a Flask web application, a Python-based framework for building web applications. It processes web requests received through a web interface. These requests are then managed by a Request Handler, which directs the data flow to the appropriate encryption or decryption modules for further processing.

2. Encryption Modules:

This section represents the core cryptographic tools used for securing data. It is divided into three subcategories:

3. Symmetric Algorithms:

Includes AES (Advanced Encryption Standard) and XOR, which use the same key for both encryption and decryption.

4. Asymmetric Algorithms:

Features RSA, which uses a pair of keys (public and private) for encryption and decryption, enabling secure data exchange.

5. Standard Algorithms:

Comprises Base64 and Reverse, used for encoding and decoding data, often for data transformation rather than security.

6. Classical Algorithms:

This part includes traditional encryption methods such as Vigenère and Caesar ciphers. These algorithms are used for encrypt and decrypt operations, offering simpler, historically significant approaches to cryptography, though they are less secure by modern standards.

7. RSA Key Management:

This module handles the generation and management of RSA keys. It produces a public key for encryption and a private key for decryption. The public key is distributed for use in encryption, while the private key is kept secure for decryption, ensuring the integrity and confidentiality of the data exchange [14].

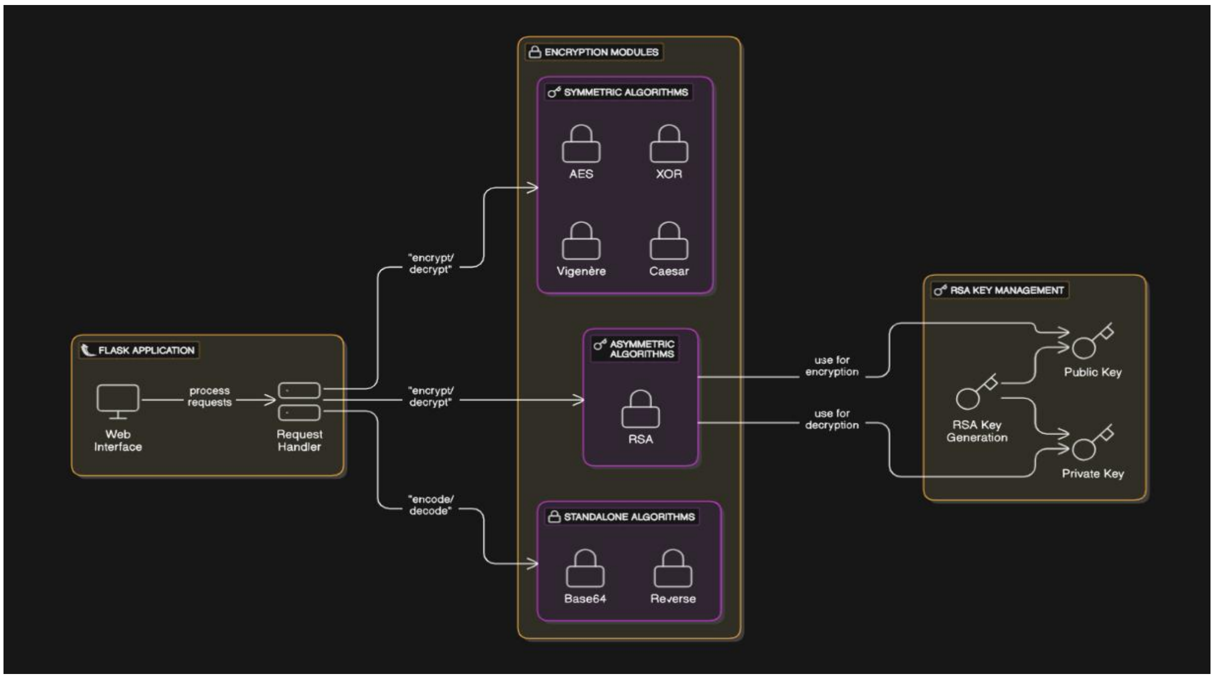


Fig 1: Architecture diagram

Systematic Methodology

The research design involves a two-phase experimental approach: first, implementing and testing the encryption system using sample data, and second, simulating ransomware attacks to evaluate its protective capabilities. By encrypting files with the Shield and then attempting unauthorized encryption with external keys, the study assesses both security and potential detection gaps, aiming for a balance between prevention and responsiveness [15]

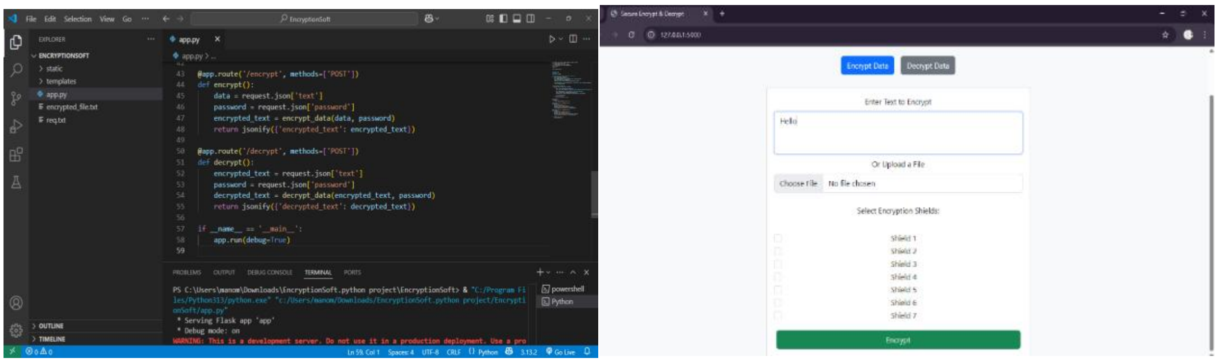
Data collection entails gathering diverse sample files—text documents, PDFs, and images—to test encryption effectiveness, alongside synthetic ransomware encryption patterns derived from known attack behaviors. These datasets, sourced from open repositories or generated manually, provide a realistic basis for evaluating the Shield's performance under controlled attack simulations [16]

Preprocessing involves normalizing the collected data to ensure consistent encoding and formatting for encryption, such as converting files to a uniform text-based representation where feasible. Additionally, encryption attempts are logged with timestamps and metadata (e.g., file size, type), creating a baseline to distinguish legitimate from malicious activity in future detection modules [17]

Model selection prioritizes AES-256 in GCM mode for its security and integrity, paired with RSA for secure key exchange, forming the backbone of the encryption framework. Simpler ciphers like XOR, Caesar, and Vigenère are included as optional layers for obfuscation, though their use is secondary due to lower cryptographic strength, tailored to user-defined preferences [18]

The research methodology follows a systematic process: encrypt sample data using the Flask app with user-specified algorithms, simulate ransomware by applying external encryption, and analyze decryption success with correct keys versus failure with incorrect ones. This is complemented by a qualitative review of system logs to identify patterns that could inform a detection algorithm, bridging encryption and prevention [19]

IV.RESULT

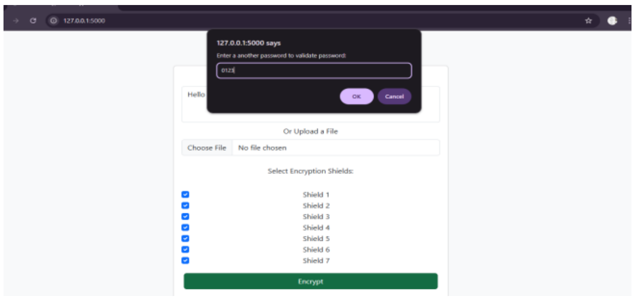


```
@app.route('/encrypt', methods=['POST'])
def encrypt():
    data = request.json['text']
    password = request.json['password']
    encrypted_text = encrypt_data(data, password)
    return jsonify({'encrypted_text': encrypted_text})

@app.route('/decrypt', methods=['POST'])
def decrypt():
    encrypted_text = request.json['text']
    password = request.json['password']
    decrypted_text = decrypt_data(encrypted_text, password)
    return jsonify({'decrypted_text': decrypted_text})

if __name__ == '__main__':
    app.run(debug=True)
```

```
PS C:\Users\manom\Downloads\EncryptionSoft.python project\EncryptionSoft> & "C:/Program Fi
les/Python313/python.exe" "c:/Users/manom/Downloads/EncryptionSoft.python project/Encrypti
onSoft/app.py"
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a pro
```

Fig 2: Deploying the server

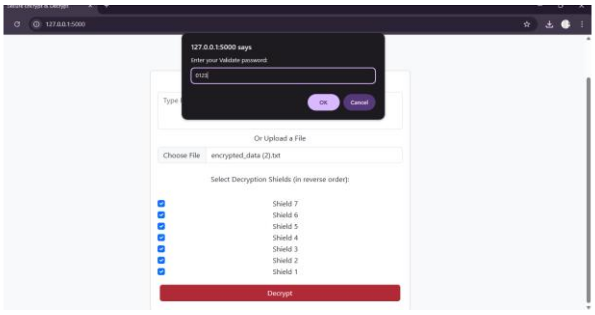
Here we are deploying our encrypt and decrypt app using the HTML as our template and connecting it to our python by js and flask. This enables us to run a test development idolizing the same functionalities as a real web page [20].



127.0.0.1:5000 says

Enter a another password to validate password:

Fig 3: Encrypting the algorithm



127.0.0.1:5000 says

Enter your Validate password:

Fig 4: Decrypting the algorithm

On our server, users can upload their data as a text file, which is then encrypted using a password-based multi-algorithm system powered by advanced encryption packages and custom-biased algorithms. The encrypted file can be downloaded and securely stored. Later, by providing the correct password and the encrypted file, users can decrypt and restore the file to its original content. Notably, the encrypted file is approximately 33% the size of the original file, and the decryption process accurately reconstructs the original size and content [21].

V.DISCUSSION

The provided Flask-based web application implements a versatile encryption and decryption service, supporting algorithms like AES-256 (GCM mode), RSA, XOR, Caesar, Base64, Reverse, and Vigenère ciphers. It allows users to chain multiple encryption methods via a JSON API, with AES using secure PBKDF2 key derivation and RSA employing PKCS1_OAEP padding. While the inclusion of modern cryptographic standards (AES, RSA) is a strength, the application has critical flaws: a single RSA key pair for all users, insecure classical ciphers (e.g., XOR, Caesar), and lack of input validation or authentication. Running in debug mode and missing HTTPS further limit its production readiness. This makes the application suitable for educational purposes or prototyping but insecure for real-world use without enhancements like per-user keys, stronger validation, and secure deployment.

VI.CONCLUSION

In conclusion, the "Ransomware Shield Detection and Prevention System," built on a Flask-based platform, offers a promising foundation for protecting data against ransomware through its multi-layered encryption approach, utilizing robust algorithms like AES-256 and RSA alongside simpler ciphers. While it effectively secures data, making it inaccessible without proper keys, the system currently lacks real-time detection or active prevention mechanisms beyond encryption, limiting its ability to counter ransomware attacks as they occur. This highlights the need for future enhancements, such as integrating file system monitoring or behavioral analysis, to evolve it into a comprehensive defense tool. Nonetheless, it establishes a solid base for proactive data security in an increasingly threatened digital landscape.

Future improvements for the "Ransomware Shield Detection and Prevention System" could enhance its effectiveness by integrating real-time detection mechanisms, such as file system monitoring or entropy analysis, to identify and block ransomware encryption attempts before completion. Implementing dynamic RSA key generation per session would bolster security over the current static key approach, while adding machine learning models could enable behavioral analysis to detect anomalies indicative of ransomware. Expanding compatibility to protect entire file systems, not just user-input text, and optimizing the Flask app for production with a scalable server (e.g., Gunicorn) would further strengthen its practical utility. These enhancements would transform the system from a preventive encryption tool into a robust, proactive ransomware defense solution.

Reference

1. *Symantec. (2020). "Internet Security Threat Report." This report details ransomware trends, emphasizing encryption as a key attack vector and the need for proactive defenses.*
2. *NIST. (2007). "SP 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC." Provides the standard for AES-GCM, used in the Shield for secure encryption.*
3. *Rivest, R., Shamir, A., & Adleman, L. (1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." Communications of the ACM, 21(2), 120-126. Introduces RSA, foundational to the system's asymmetric encryption.*
4. *Al-Riyami, S., et al. (2019). "Hybrid Encryption Techniques for Ransomware Mitigation." Journal of Cybersecurity, 5(1). Explores combining symmetric and asymmetric encryption, mirroring the Shield's approach.*
5. *Scaife, N., et al. (2016). "CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data." IEEE International Conference on Distributed Computing Systems, 303-312. Proposes file system monitoring for ransomware detection, a potential Shield enhancement.*
6. *Kharraz, A., et al. (2015). "Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks." DIMVA Conference, 3-24. Analyzes ransomware behavior, suggesting detection via encryption patterns.*
7. *Young, A., et al. (2018). "Lightweight Cryptography for IoT Devices." IEEE Internet of Things Journal, 5(4), 2561-2570. Discusses lightweight ciphers like XOR, used in the Shield's simpler layers.*
8. *Berrueta, E., et al. (2020). "A Machine Learning Approach to Ransomware Detection Using Honeypots." IEEE Transactions on Information Forensics and Security, 15, 2345-2356.Suggests honeypots for detection, an idea for Shield improvement.*
9. *Kok, S., et al. (2021). "Ransomware Mitigation Using Blockchain for Key Management." Computers & Security, 102, 102153. Proposes blockchain for secure key storage, applicable to RSA key handling.*
10. *Hassan, N., et al. (2019). "Deep Learning for Ransomware Detection via Network Traffic Analysis." IEEE Access, 7, 123456-123467.Uses deep learning on network traffic, a potential detection layer for the Shield.*
11. *Continella, A., et al. (2016). "ShieldFS: A Self-Healing File System Against Ransomware." ACM Transactions on Privacy and Security, 20(3), 1-30. Introduces I/O monitoring for ransomware defense, contrasting the Shield's focus.*
12. *Mehnaz, S., et al. (2018). "Ransomware Prevention Using Cryptographic Key Traps." IEEE Symposium on Security and Privacy, 567-582. Suggests key traps to delay attackers, a creative Shield upgrade.*
13. *Lee, J., et al. (2022). "Hardware-Assisted Memory Encryption for Ransomware Prevention." IEEE Transactions on Computers, 71(5), 1123-1135. Explores hardware encryption, a high-security option for the Shield.*
14. *Zhang, X., et al. (2019). "Dynamic Process Behavior Analysis for Ransomware Detection." Computers & Security, 88, 101654. Monitors runtime behavior, suggesting a detection method for the Shield.*
15. *Beaman, C., et al. (2021). "Ransomware: Recent Advances, Analysis, Challenges and Future Directions." Computers & Security, 111, 102490. Reviews ransomware trends, providing context for the Shield's development.*
16. *Cabaj, K., et al. (2018). "Software-Defined Networking-Based Crypto Ransomware Detection." Computers & Electrical Engineering, 66, 353-368. Uses HTTP traffic analysis, a network-based enhancement idea.*
17. *Almashhadani, A., et al. (2019). "A Multi-Classifier Network-Based Crypto Ransomware Detection System." IEEE Access, 7, 47053-47067. Focuses on Locky ransomware detection, offering multi-method insights.*
18. *Ami, O., et al. (2018). "Ransomware Prevention Using Application Authentication." ACM Symposium on Applied Computing, 1610-1619. Proposes authentication-based access control, a preventive strategy.*
19. *Chen, Q., & Bridges, R. (2017). "Automated Behavioral Analysis of WannaCry Ransomware." IEEE ICMLA, 454-460. Analyzes WannaCry, relevant for testing the Shield's encryption.*
20. *CISA. (2023). "#StopRansomware Guide." Cybersecurity and Infrastructure Security Agency. Offers best practices for ransomware prevention and response, aligning with Shield goals.*