Enhancing Data Resilience and Security in Distributed Storage Systems Against Ransomware Attacks

Diana-Ioana Voineag
Faculty of Automatic Control and Computer Engineering
Gheorghe Asachi Technical University of Iasi
Iasi, Romania
diana-ioana.voineag@student.tuiasi.ro

Catalin Mironeanu
Faculty of Automatic Control and Computer Engineering
Gheorghe Asachi Technical University of Iasi
Iasi, Romania
catalin.mironeanu@academic.tuiasi.ro

***Abstract*—This paper aims to present an efficient data security mechanism through the implementation of Transparent Data Encryption (TDE) in the case of crypto-ransomware variant of attacks. The proposed solution relies on a Man-in-the-middle (MITM) proxy, which enables dynamic encryption and decryption of data both in transit and at rest, using state-of-the-art authenticated encryption algorithms AES-GCM and ChaCha20-Poly135, which are dynamically selected according to the characteristics of the files, minimizing the performance impact on the system. For a secure management of cryptographic keys, a vault service was integrated, enabling the development of a Zero Knowledge architecture. Ransomware detection process adds minimal overhead compared to the traditional analysis methods. The proposed architecture provides improved data privacy and resilience against exfiltration attacks or data leaks. The stored data is fully encrypted and lacks identifiable metadata, enhancing protection against unauthorized access. Preliminary results confirm the effectiveness of the proposed solution and demonstrate the benefit of its integration into GlusterFS and HDFS platforms by substantially enhancing data security and protection within distributed systems as storage agnostic encryption security layer.**

***Index Terms*—Data resilience, Information security, Distributed file systems, Ransomware, Data in transit, Data at rest**

I. Introduction

The rapid growth of digitally stored and processed data has heightened concerns over cyber threats - particularly ransomware attacks and privacy breaches - for organizations and individuals relying on distributed storage systems. With increased reliance on distributed storage systems and cloud environments, ensuring the confidentiality, integrity, and availability of data has become a critical feature for modern storage systems. In the given context, distributed file storage solutions, such as GlusterFS or HadoopFS, offer considerable advantages including scalability, redundancy, and ease of deployment. However, this widespread adoption has also attracted important cyber threats, among which ransomware attacks stand out due to their disruptive nature and potentially devastating consequences.

Ransomware attacks primarily compromise data integrity by encrypting critical files and demanding ransom for their decryption, while also often exfiltrating sensitive data for additional leverage. The distributed nature of storage solutions like the ones mentioned above, while beneficial for redundancy and scalability, do not inherently provide robust protection against such targeted threats. Existing security measures often lack comprehensive mechanisms for rapid attack detection and inherent transparent defense against these security threats. Therefore, developing resilient, integrated solutions for securing distributed storage infrastructures remains an ongoing challenge.

This paper addresses this need by proposing a novel approach through the design and implementation of a storage platformagnostic proxy that applies file-aware Transparent Data Encryption (TDE). This proxy dynamically encrypts and decrypts data in transit and at rest, utilizing state-of-the-art encryption algorithms like AES-GCM and ChaCha20-Poly135. These algorithms are dynamically selected based on specific file characteristics and importance, thus maintaining an optimal balance between security and system performance.

Secure cryptographic key management is critical to the effectiveness of encryption strategies. The implementation of a Zero Knowledge security model leverages HashiCorp Vault, a secure key management service, to ensure confidential key management and prevent metadata leakage. Every sensitive information about each file and the related encryption process are stored in vault. This approach ensures that stored data is encrypted without retaining any identifiable metadata, significantly reducing vulnerabilities related to unauthorized access or data compromise.

To complement the encryption strategy, this work introduces a lightweight ransomware detection mechanism based on an Observer-Agent model. The system operates at the file handler level within the operating system, monitoring low-level I/O activity in real time. Each agent locally tracks process behaviour, and upon detecting abnormal file operation patternssuch as a rapid increase in open or modified file descriptors, it immediately notifies the central Observer node. This low-latency approach enables rapid response, such as process termination, while maintaining minimal system overhead. By focusing on kernel-level indicators (eg. number of open file descriptors per process, unusual patterns in syscalls) rather than post-encryption symptoms, the proposed detection framework provides early-stage protection against crypto-ransomware attacks in distributed environments. An empirical evaluation demonstrates the system's efficiency and interoperability within distributed file systems GlusterFS and HDFS.

II. Related work

Ransomware attacks represent one of the greatest threats to data availability and integrity in modern distributed storage systems and cloud environments. As outlined in a comprehensive ransomware overview [1], these attacks have evolved from threats that are misleading and targeting individuals to malicious operations specifically designed to compromise enterprise-level storage systems. The financial impact of such attacks has increased exponentially, with global damages exceeding \$42 billion in 2024 [2].

A. Ransomware Detection Approaches

Ransomware itself has evolved into multiple types crypto-ransomware, locker ransomware, scareware, and leakware, each exhibiting distinct behaviours and attack goals. Among these, crypto-ransomware and leakware have become particularly disruptive due to their combination of file encryption and data exfiltration, as noted in the comprehensive taxonomy by Razaulla et al. [2]. Moreover, the rise of Ransomware-as-a-Service (RaaS) has lowered the barrier to entry for cybercriminals, enabling widespread and scalable attacks through affiliate networks. For our specific study, crypto-ransomware is the most relevant one as its file-encryption-centric nature poses a direct threat to the integrity and availability of data within distributed storage systems.

Most of the current ransomware detection approaches can be mainly categorized into signature-based, behaviour-based, and hybrid detection mechanisms [2]. Although machine learning and artificial intelligence techniques have great results in identifying ransomware patterns [3], these methods often introduce substantial computational overhead and may detect the threat only after malicious-encryption has begun [4]. As highlighted by Scaife et al. [5], by the time many detection systems alert administrators, significant amounts of data may have already been compromised.

Our proposed solution differs by implementing detection at kernel level, intercepting file operations before potential encryption by ransomware can be completed. This strategy aligns with research by Kharaz et al. [6], who demonstrated that monitoring low-level I/O requests can detect ransomware with higher accuracy and lower latency than application-level solutions.

B. Transparent Data Encryption in Distributed Systems

TDE has evolved from being a security enhancement in traditional relational databases like MySQL, PostgreSQL, and Oracle, where it primarily focused on encrypting tablespaces or columns at rest, to becoming a critical security feature in distributed storage environments. This transition reflects the growing need to ensure end-to-end data protection across large-scale, decentralized infrastructures where sensitive data is increasingly exposed during transit, replication, and multi-node access. As analysed in [7], TDE mechanisms in distributed systems offer an optimal balance between security and usability by operating transparently to users, ensuring strong protection and privacy without impacting the user experience. The agnostic encryption layer approach is inspired from this work, but extends it by incorporating a man-in-the-middle (MITM) proxy, enabling full storage platform agnosticism.

The implementation of TDE in Hadoop Distributed File System (HDFS) [8] provides insightful ideas for this work. However, our approach differs by functioning independently of the underlying storage system. Benchmarking studies by Hu and Ji [9] indicate that attentive algorithm selection and optimization can minimize the performance impact of TDE, which motivated the dynamic encryption strategy.

C. Observer-Agent Architecture

The observer pattern has been broadly used for distributed systems security. [10] demonstrated the effectiveness of observer model in large-scale distributed systems, where centralized monitoring can detect patterns invisible to individual nodes. Our implementation builds upon this foundation by enabling a central observer node and additional agents attached to each storage node. The role of the agents is to monitor file system operations, tracking low-level I/O activities such as open, write, and delete. [11] established that effective distributed monitoring systems must find a balance between local autonomy and centralized decision-making. Their work lead to the idea of centralized decision-making mechanism developed for this project. Similarly, [12] explored the performance of various communication protocols and concluded WebSocket as a strong candidate for real-time data transmission due to its low latency and persistent connection model. Therefore, it was chosen to handle communication between the observer and agents, as it offers a good balance between speed and reliability for transmitting security alerts in distributed environments. GRPC was also noted as a compelling alternative, mainly for scenarios requiring stricter type safety or more complex service definitions and will be studied for future improvements.

The concept of cooperative intrusion detection through distributed observers, before establishing a final decision, has been further explored by Vasilomanolakis et al. [13], who established that collaborative detection approaches can significantly reduce false positives while maintaining high detection sensitivity. The proposed system implements a simpler version of the one analysed, where each agent communicates only with the observer.

D. Process Behaviour Monitoring and Control

For effective ransomware detection [2], [5], [6], monitoring process behaviour at the system level provides critical insights.

[14] developed a framework for identifying malicious process behaviour through syscall pattern analysis, demonstrating that ransomware exhibits distinctive I/O patterns that can be detected with high accuracy. Their work has informed the agent implementation to monitor file descriptors at kernel level.

The process termination authorization mechanism in this project is inspired by the capability-based distributed authorization system proposed by Li et al. [15]. Their approach uses cryptographically secure tokens to enforce fine-grained and context-aware permissions in distributed environments. Applying this principle, impactful actions can be performed only when authorized by a delegated central node, helping maintain security as well as fast response times.

E. Encryption Algorithm Selection

The choice of encryption algorithms significantly impacts both security and system performance. For this paper, the dynamic selection between AES-GCM and ChaCha20-Poly135 is supported by Bernstein's analysis [16], which demonstrated that ChaCha20 offers comparable security to AES while providing superior performance on systems without dedicated AES hardware acceleration. For larger files where throughput is critical, this feature becomes even more important.

F. Key Management in Distributed Environments

Secure key management represents the basis of effective encryption strategies. Ferretti et al. [17] and Santos et al. [18] analyse various key management services (KMS) approaches, highlighting the importance of secure key storage, rotation policies, and access controls. The integration of HashiCorp Vault aligns with these recommendations, providing a robust foundation for cryptographic operations while minimizing key exposure and enhancing metadata protection.

G. Monitoring and Detection in Distributed File Systems

The Observer-Agent architecture implemented in the proposed system builds upon monitoring frameworks described by [19], which established that comprehensive monitoring in distributed systems requires a hierarchical approach combining local and central perspectives. In the context of ransomware detection, Continella et al. [5] demonstrated that file access patterns and I/O request characteristics provide strong indicators of malicious activity.

The kernel-level monitoring approach is further informed by research on file system activity patterns during ransomware attacks [20], traffic analysis techniques [21], and real-time behavioural analysis [22]. The TransCrypt framework [23] also provides valuable insights regarding the integration of encryption services into file system operations, though this implementation emphasizes more the ransomware detection capabilities.

H. Low-Latency Communication in Security Systems

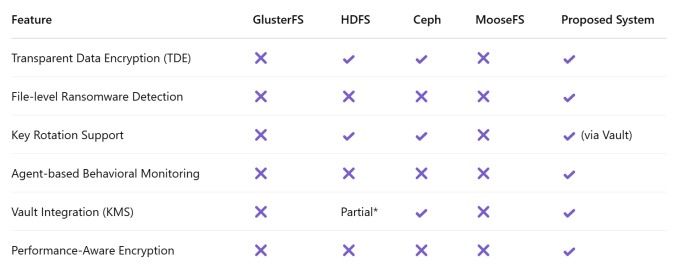
The effectiveness of distributed security monitoring depends entirely on communication efficiency. Vaidyanathan et al. [24] compared multiple communication protocols for security-critical applications, finding that binary protocols with persistent connections offer significant advantages for real-time threat response. Their evaluation and message queue technologies provides valuable insights for the developed observer-agent communication design.

For OpenStack environments specifically, Wang et al. [25] developed a security monitoring framework that integrates both the hypervisor and storage layers, offering improved visibility while maintaining low resource and performance overhead. This architecture is similar to the proposed approach of deploying monitoring agents on each storage node which continuously communicate to a central proxy node, responsible for decision-making in the alert context.

I. Recovery Mechanisms

While the primary focus of this project is early detection and encryption of data in transit and at rest as a way to avoid data compromise, recovery capabilities remain essential for comprehensive protection. Swift RAID approaches [26], popularity-based recovery strategies [27], and Reed-Solomon coding techniques [28] offer promising directions for data recovery. However, as noted in current literature, traditional replication and snapshot-based solutions continue to provide the most reliable recovery options for production environments.

This research tries to integrate abstract security theories with real-world applications by building a system that combines quick threat detection, seamless encryption, and secure key storage in one, platform-agnostic product that can be used as a secure layer for traditional distributed storage system solutions. The performance and security characteristics of the existing approaches compared to the proposed solution, are summarized in Figure 1.



Feature	GlusterFS	HDFS	Ceph	MooseFS	Proposed System
Transparent Data Encryption (TDE)	✗	✓	✓	✗	✓
File-level Ransomware Detection	✗	✗	✗	✗	✓
Key Rotation Support	✗	✓	✓	✗	✓ (via Vault)
Agent-based Behavioral Monitoring	✗	✗	✗	✗	✓
Vault Integration (KMS)	✗	Partial*	✓	✗	✓
Performance-Aware Encryption	✗	✗	✗	✗	✓

Fig. 1. Comparison of the conventional distributed storage systems with the proposed system.

J. Algorithm Selection Likelihood

To determine the most suitable encryption algorithm for each file, the system uses a weighted scoring mechanism that considers three main factors: file size, sensitivity, and value. Each available algorithm (e.g., AES-GCM, ChaCha20-Poly1305) is scored based on how well it performs following these criteria, and the algorithm with the highest probability is selected.

The likelihood score $S(A)$ for a given algorithm A is calculated as:

$$S(A) = \sum_{i=1}^{n} w_i \cdot f_i(A) \tag{1}$$

where:

- w_i represents the weight assigned to factor i
- $f_i(A)$ is the score of algorithm A for that factor

The factors and their respective weights are:

$$S(A) = 0.4 \cdot f_{size}(A) + 0.3 \cdot f_{sens}(A) + 0.3 \cdot f_{value}(A) \tag{2}$$

where:

- $f_{size}(A)$ favors AES-GCM for small files and ChaCha20 for large ones:
  - $f_{size}(\text{AES-GCM}) = 0.9$ for files ≤ 5 MB
  - $f_{size}(\text{ChaCha20}) = 0.9$ for files > 5 MB
- $f_{sens}(A)$ = 1.0 for highly sensitive files, 0.5 otherwise
- $f_{value}(A)$ = 1.0 for high-value files, 0.5 otherwise

Once the scores are computed for all available algorithms, a `softmax` function is applied to convert them into selection probabilities:

$$P(A) = \frac{e^{S(A)}}{\sum_{j=1}^{m} e^{S(A_j)}} \tag{3}$$

where m is the total number of candidate algorithms.

The algorithm selection is included in the Related Work section, as it does not introduce a novel equation to address the problem, and the weight factors were determined empirically.

III. Proposed Solution

A. Overview

The proposed proxy solution focuses on two key aspects: assuring privacy through encryption and enhancing security through ransomware detection. The system is designed to be completely transparent to users, meaning they can utilize the natural, familiar interface built without noticing the security layers operating behind the scenes. The proxy works as an interceptor that enhances existing systems rather than replacing them completely. This approach allows us to both prevent attacks through encryption and detect suspicious ransomware activity through careful monitoring of file operations.

B. System architecture overview

The architecture developed is modular, illustrated in Fig. 2 consisting of multiple layers working together seamlessly as stand-alone APIs. The interception layer captures and routes file requests, while the processing layer analyses files and classifies them based on their characteristics. The security layer handles all encryption operations and key management, as well as the connection with the outside HashiCorp Vault service, with a storage abstraction layer supporting different storage back-ends, making the proxy system platform-agnostic. A Zero-knowledge architecture has been outlined, keeping all metadata, keys and valuable information locked in a HashiCorp Vault service, configured with the highest standards security-wise, which will be detailed in the HashiCorp Vault subsection.

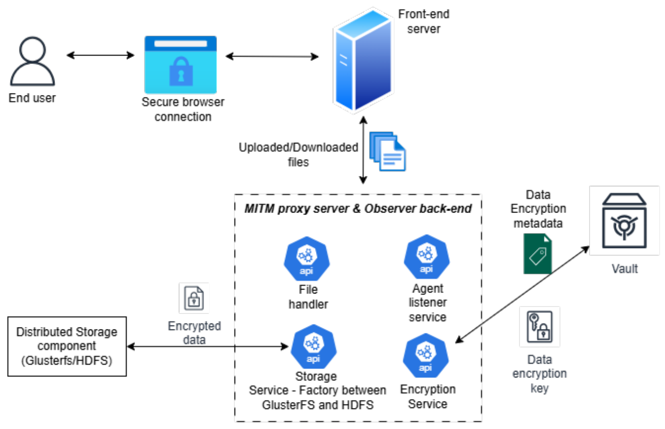

Front-end server
End user
Secure browser connection
Uploaded/Downloaded files
MITM proxy server & Observer back-end
Data Encryption metadata
Vault
api
File handler
api
Agent listener service
Encrypted data
Distributed Storage component (Glusterfs/HDFS)
api
Storage Service - Factory between GlusterFS and HDFS
api
Encryption Service
Data encryption key

Fig. 2. System architecture focused on TDE layer

C. TDE implementation

The encryption service dynamically selects the most appropriate algorithms based on specific file characteristics through a strategy based pattern. Smaller files are handled differently than larger ones to optimize performance, and we adjust encryption strength according to data sensitivity. So for smaller files the system would use AES-GCM for encryption and make subsequent API calls to the external Vault service, while for large files, the algorithm chosen is ChaCha20-Ploy135 and the encryption key used would be one cached locally to avoid frequent data transfers, resulting in high performance overhead. Furthermore, for efficient processing of large files, we use a streaming, chunked approach with dynamically computed size segments that reduces memory requirements when uploading and downloading file from the front-end service. Finally, all encrypted files follow a standardized format (OpaqueStorageFormat) with defined structures for headers, metadata, and the encrypted payload.

D. Hashicorp Integration

As previously mentioned, HashiCorp Vault has been integrated into this system to manage encryption keys securely using role-based access controls that limit who can access which keys. The system automatically rotates keys based on configurable schedules and maintains version history to prevent issues with previously encrypted data. To minimize performance impact, an optimized caching mechanism was created to reduce the number of Vault API calls while maintaining strong data privacy and security. Sensitive credentials automatically expire based on Time-To-Live (TTL) configurations, and all metadata is stored and retrieved using secure methodologies and transit keys to prevent unauthorized access. The Zero-Knowledge architecture of this module is completely reliant on Vault and its unbeatable security mechanism, storing everything needed for decryption or even any remotely sensitive information of the file inside the metadata container. This way, even if the files end up being exfiltrated, the attacker won't be able to obtain enough information in order to actually be able to read the data.

E. Ransomware Detection process

The main goal of this proposed approach would be to keep the response-time as close to real-time as possible. The related work in the Ransomware detection area, presented in the subsection II-A, conclude that the most effective approach would be a low-level, operating-system file handler one, as it adds minimal overhead compared to the traditional analysis methods.

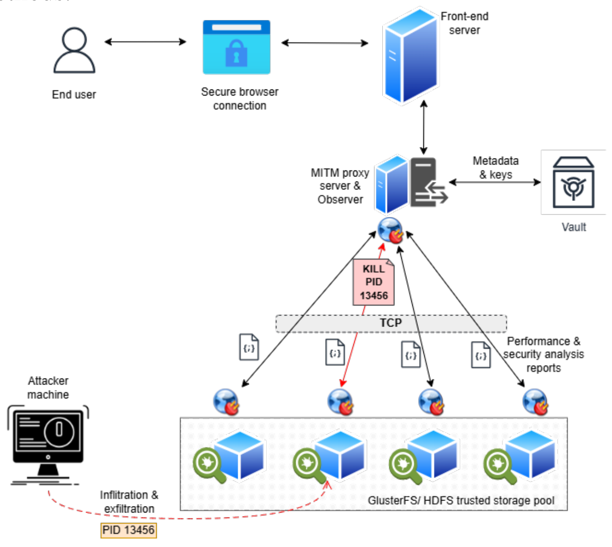

End user
Secure browser connection
Front-end server
MITM proxy server & Observer
Metadata & keys
Vault
KILL PID 13456
TCP
Performance & security analysis reports
Attacker machine
Infiltration & exfiltration
PID 13456
GlusterFS/ HDFS trusted storage pool

Fig. 3. System architecture based on detection processes

The communication between the main agent, also called the Observer, and each agent correspondent to each storage node, is done through sockets using the TCP protocol and the messages are sent in plain JSON format, as illustrated in Figure 3. Each agent analyses the file processes in its own environment and if one exceeds the allowed threshold for file handlers operations, the agent will notify the observer instantly. In this scenario, and for this particular work the observer will almost always tell the agent to kill the suspicious process, but for further work the observer can implement some filtering as well. For demonstration purposes only, we consider that it's best be as cautious as possible.

F. Agent-Observer Communication Flow

The communication flow, illustrated in Fig. 4, is designed to ensure fast and minimal-overhead interactions between each agent correspondent on a storage node and the global observer. Agents perform lightweight, low-level file operation monitoring by counting the number of file handlers being open and report this activity using short JSON messages.

When a process exceeds the predefined operational thresholds - such as abnormal file descriptor usage - an `"Alert"` message is issued to the observer. This message contains very few metadata: the PID, node identifier, and activity count. To ensure speed and resilience, the agent logic is stateless and redundant, including running on the main observer node itself, allowing it to self-monitor as well.

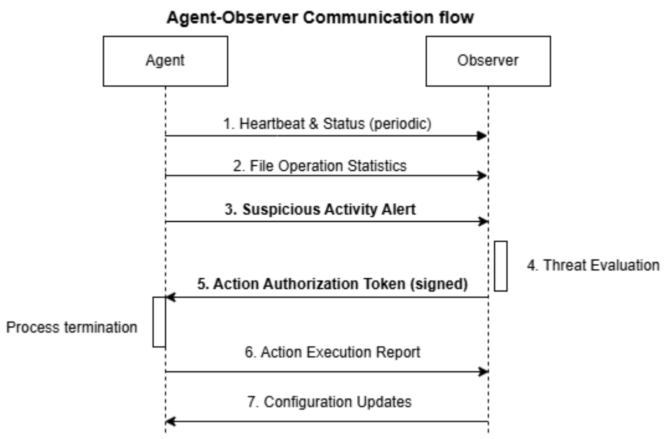

Agent-Observer Communication flow
Agent
Observer
1. Heartbeat & Status (periodic)
2. File Operation Statistics
3. Suspicious Activity Alert
4. Threat Evaluation
5. Action Authorization Token (signed)
Process termination
6. Action Execution Report
7. Configuration Updates

Fig. 4. Communication flow between observer and agent

The flow described begins with periodic heartbeats and file operation statistics (Steps 1 - 2). After detecting suspicious activity (Step 3), the agent raises an alert to the observer, which evaluates the threat (Step 4) and, for this prototype, sends back a signed authorization token to allow process termination (Step 5). Following execution, the agent reports the outcome (Step 6) and updates any relevant configuration parameters (Step 7). This feedback-oriented design ensures rapid response with minimal intrusion on normal operations and overall performance overhead.

IV. Initial Results

The experimental evaluation focused on measuring the effectiveness of the Transparent Data Encryption (TDE) implementation and its impact on system performance. The tests were conducted across both GlusterFS and HDFS OpenStack environments to validate the platform-agnostic nature of the solution.

A. Encryption Performance and Overhead

The dynamic encryption strategy demonstrated efficient performance characteristics across different file sizes and types. The streaming chunked approach, using 1MB segments, proved particularly effective in minimizing memory overhead while maintaining security:

- Small files (< 5 MB) AES-GCM encryption with 450 MB/s throughput
- Large files (> 5 MB): ChaCha20-Poly135 encryption with 520 MB/s throughput
- Average encryption latency: 20ms per operation

The proxy layer introduced minimal overhead to the system:

- Average request processing time: 15-30ms
- Proxy overhead: 4.2% of total operation time

B. Integration Performance

The integration with HashiCorp Vault demonstrated efficient key management with minimal impact on system performance for small and medium sized files. For larger files, a caching mechanism has been implemented to reduce some of the overhead of Vault API calls.

The storage abstraction layer maintained consistent performance across both platforms:

- GlusterFS integration:
  - Upload throughput: 510 MB/s
  - Download latency: 42ms
  - Operation success rate: 100% (in testing environment)
- HDFS integration:
  - Upload throughput: 480 MB/s
  - Download latency: 45ms
  - Operation success rate: 100% (in testing environment)

C. *Metrics and Monitoring*

The metrics service provided comprehensive performance monitoring with minimal overhead:

- Metrics collection interval: 5 seconds
- Average metrics processing time: 3ms
- Memory usage for metrics storage: $< 5\%$ of total system memory

V. Conclusion and Future work

The experimental results demonstrate that the proposed TDE implementation successfully balances security requirements with system performance. The proxy-based approach introduces minimal overhead while providing robust encryption capabilities across different storage platforms. Key findings include:

- The dynamic encryption strategy effectively handles different file characteristics
- The streaming chunked approach with dynamically computed sizing reduces memory requirements
- The proxy layer adds minimal latency to operations
- The platform-agnostic design maintains consistent performance across different storage systems
- The metrics service provides insights with low impact on system performance

Future improvement could focus on:

- Optimization of the encryption pipeline for more specific file types and larger files as well
- Enhancement of the caching mechanisms for key management and metadata retrieval
- Extension to additional storage platforms
- Development of a more robust decision-making strategy for the observer component

The successful initial implementation and testing of this solution provide a strong foundation for secure data storage in distributed environments, demonstrating that robust encryption can be achieved without significant performance penalties and the importance of each method in achieving the proposed resilient system.

References

[1] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the gordian knot: A look under the hood of ransomware attacks," in *Proceedings of the 12th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment - Volume 9148*, ser. DIMVA 2015, 07 2015, pp. 3–24.

[2] S. Razaulla, C. Fachkha, C. Markarian, A. Gawanmeh, W. Mansoor, B. C. M. Fung, and C. Assi, "The Age of Ransomware: A Survey on the Evolution, Taxonomy, and Research Directions," *IEEE Access*, vol. 11, pp. 40 698–40 723, 2023.

[3] U. Zahoora, A. Khan, M. Rajarajan, S. Khan, D. Asam, and T. Jamal, "Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive Pareto Ensemble classifier," *Scientific Reports*, vol. 12, 09 2022, DOI: 10.1038/s41598-022-19443-7.

[4] M. Cen, F. Jiang, X. Qin, Q. Jiang, and R. Doss, "Ransomware early detection: A survey," *Computer Networks*, vol. 239, pp. 110–138, 2024.

[5] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barenghi, S. Zanero, and F. Maggi, "ShieldFS: a self-healing, ransomware-aware filesystem," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, ser. ACSAC '16. New York, NY, USA: ACM, 2016, pp. 336–347.

[6] A. Kharraz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "Unveil: a large-scale, automated approach to detecting ransomware," in *Proceedings of the 25th USENIX Conference on Security Symposium*, ser. SEC'16. USA: USENIX Association, 2016, pp. 757–772.

[7] F. Jiang, Z. Pan, Q. Li, L. Huang, and D. Zhang, "Research on the Application of Transparent Encryption in Distributed File System HDFS," in *19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, 2020, pp. 1–4.

[8] M. M. Shetty and D. H. Manjaiah, "Data security in hadoop distributed file system," in *2016 International Conference on Emerging Technological Trends (ICETT)*, 2016, pp. 1–5.

[9] J. Hu and A. Klein, "A Benchmark of Transparent Data Encryption for Migration of Web Applications in the Cloud," in *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2009.

[10] M. Mushtaq, U. Akram, I. Khan, S. Khan, A. Shahzad, and A. Ulah, "Cloud computing environment and security challenges: A review," *International Journal of Advanced Computer Science and Applications*, vol. 8, pp. 183–195, 10 2017.

[11] A.-S. K. Pathan, *The State of the Art in Intrusion Prevention and Detection*. Auerbach Publications, 2014, ISBN: 978-1-4822-0351-5.

[12] L. Kamiski, M. Kozowski, D. Sporysz, K. Wolska, P. Zaniewski, and R. Roszczyk, "Comparative review of selected internet communication protocols," *Foundations of Computing and Decision Sciences*, vol. 48, no. 1, pp. 39–56, 2023.

[13] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and survey of collaborative intrusion detection," *ACM Comput. Surv.*, vol. 47, no. 4, May 2015.

[14] C. J.-W. Chew, V. Kumar, P. Patros, and R. Malik, "ESCAPADE: Encryption-Type-Ransomware: System Call Based Pattern Detection," in *Network and System Security*. Springer-Verlag, 2020, pp. 388–407.

[15] A. S. Li, R. Safavi-Naini, and P. W. L. Fong, "A capability-based distributed authorization system to enforce context-aware permission sequences," in *Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies*, ser. SACMAT '22. New York, NY, USA: ACM, 2022, pp. 195–206.

[16] S. Dey, C. Dey, S. Sarkar, and W. Meier, "Revisiting Cryptanalysis on ChaCha From Crypto 2020 and Eurocrypt 2021," *IEEE Transactions on Information Theory*, vol. 68, no. 9, pp. 6114–6133, 2022.

[17] L. Seitz, J.-M. Pierson, and L. Brunie, "Key management for encrypted data storage in distributed systems," in *Second IEEE International Security in Storage Workshop*, 2003, pp. 20–20.

[18] I. Kuzminykh, M. Yevdokymenko, and D. Ageyev, "Analysis of Encryption Key Management Systems: Strengths, Weaknesses, Opportunities, Threats," *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)*, pp. 515–520, 2020.

[19] J. Joyce, G. Lomow, K. Slind, and B. Unger, "Monitoring distributed systems," *ACM Trans. Comput. Syst.*, vol. 5, no. 2, pp. 121–150, Mar. 1987.

[20] M. Putrevu, V. S. C. Putrevu, H. Chunduri, and S. Shukla, “RTR-Shield: Early Detection of Ransomware Using Registry and Trap Files,” in *Information Security Practice and Experience*. Springer Nature Singapore, 11 2023, pp. 209–229.

[21] D. Morato, E. Berrueta, E. Magaa, and M. Izal, “Ransomware early detection by the analysis of file sharing traffic,” *Journal of Network and Computer Applications*, vol. 124, 09 2018.

[22] S. Mehnaz, A. Mudgerikar, and E. Bertino, “RWGuard: A Real-Time Detection System Against Cryptographic Ransomware,” in *Research in Attacks, Intrusions, and Defenses*. Springer International Publishing, 09 2018, pp. 114–136.

[23] S. K. Sharma, “TransCrypt: Design of a Secure and Transparent Encrypting File System,” Master’s thesis, IIT Kanpur, August 2006.

[24] P. Vaidyanathan and S. F. Midkiff, “Performance evaluation of communication protocols for distributed processing,” *Computer Communications*, vol. 13, no. 5, pp. 275–282, 1990, DOI: 10.1016/0140-3664(90)90015-9.

[25] D. Zou, W. Zhang, W. Qiang, G. Xiang, L. Yang, H. Jin, and K. Hu, “Design and implementation of a trusted monitoring framework for cloud platforms,” *Future Generation Computer Systems*, vol. 29, pp. 2092–2102, 10 2013, DOI: 10.1016/j.future.2012.12.020.

[26] D. Long and B. Montague, “Swift/raid: A distributed raid system,” *Computing Systems*, vol. 7, pp. 333–359, 06 1994.

[27] S. Wu, W. Zhu, B. Mao, and K.-C. Li, “PP: Popularity-based Proactive Data Recovery for HDFS RAID systems,” *Future Generation Computer Systems*, vol. 86, 04 2017.

[28] J. S. Plank, “A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems,” *Software: Practice and Experience*, vol. 27, no. 9, pp. 995–1012, Sep. 1997.