American Journal of Data Mining and Knowledge Discovery
2026, Vol. 11, No. 1, pp. 1–7
https://doi.org/10.11648/j.ajdmkd.20261101.11

Research Article

Enhancing Security and Privacy in MVC Architecture Using a Data Encryption-Based Model

S M Alauddin Kabir*, Uzzal KumarAcharjee

Department of Computer Science and Engineering, Jagannath University, Dhaka, Bangladesh

Abstract

The Model-View-Controller (MVC) architecture is widely used in modern software development due to its modular design, scalability, and clear separation of concerns. Despite these advantages, traditional MVC applications often lack built-in security measures, leaving them vulnerable to data breaches and unauthorized access. This research proposes an enhanced MVC architecture that integrates data encryption to improve security and privacy. The proposed approach incorporates AES and RSA encryption techniques within the Model and Controller layers to protect sensitive data both at rest and during transmission. Secure data handling is also ensured at the View layer to prevent unintended data exposure. This encryption-based enhancement strengthens data confidentiality and integrity while preserving the core MVC structure. An experimental implementation of the proposed architecture was conducted using an MVC-based web application framework. The system was evaluated in terms of response time, computational overhead, and resistance to common security threats. The results indicate that the encryption-enhanced MVC model significantly improves data confidentiality and integrity while introducing only minimal performance overhead. The findings demonstrate that integrating encryption mechanisms into the MVC architecture provides an effective and practical solution for developing secure and privacy-preserving software systems, making the proposed approach suitable for modern applications that require compliance with contemporary cybersecurity and data protection requirements.

Keywords

Model, View, Controller, Security, Database

1. Introduction

Modern web applications increasingly manage sensitive and mission-critical information, including personal identifiers, authentication credentials, financial records, and healthcare data. With the widespread adoption of cloud computing, e-commerce platforms, and e-governance systems, ensuring data confidentiality and user privacy has become a primary concern for software developers and system architects. At the same time, the growing sophistication of cyber attacks has exposed significant vulnerabilities in conventional web application architectures, leading to frequent incidents of data breaches and unauthorized access [1].

The Model–View–Controller (MVC) (Figure 1) is one of the most commonly used architecture in web application development due to its clear separation of concerns, modular structure, and ease of maintenance. By dividing application logic into Model, View, and Controller components, MVC

*Corresponding author: smakabir.jnu@gmail.com (S M Alauddin Kabir)

Received: 23 December 2025; **Accepted:** 12 January 2026; **Published:** 27 January 2026

improves scalability and supports rapid development [2]. However, despite these architectural advantages, traditional MVC frameworks do not inherently provide mechanisms for securing sensitive data. As a result, developers often rely on external or ad-hoc security measures, which may not be systematically integrated into the application lifecycle.

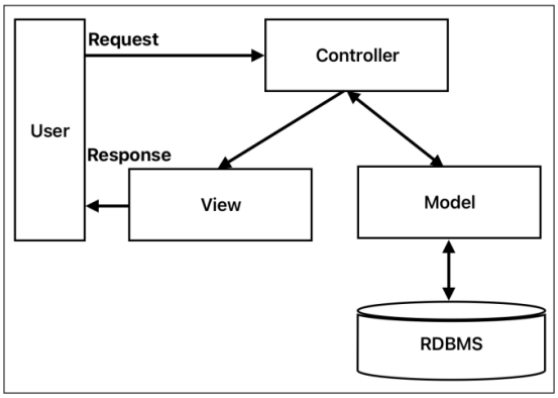

Figure 1. *MVC Architecture.*

In many MVC-based systems, sensitive data are stored in databases in plaintext or transmitted with insufficient cryptographic protection. Such practices expose applications to numerous security threats, including SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and session hijacking [3]. While authentication and authorization mechanisms can restrict access to application resources, they do not adequately protect data once an attacker gains access to the database or intercepts communication channels. Consequently, data at rest and data in transit remain vulnerable in conventional MVC implementations.

Existing studies on securing MVC applications primarily focus on access control models, secure coding practices, and vulnerability mitigation techniques [4]. Although some research incorporates encryption mechanisms, these solutions are often applied at isolated stages of the application or implemented as middleware components. Such approaches may lack architectural coherence and can introduce performance overhead or maintenance complexity. Therefore, there is a clear research gap in developing a structured, architecture-level encryption strategy that seamlessly integrates with the MVC pattern while preserving its design principles.

This paper addresses these challenges by proposing a data encryption-based enhancement to the MVC architecture using a hybrid cryptographic approach. The proposed model employs the Advanced Encryption Standard (AES-256) to secure sensitive data at rest within the Model layer, ensuring confidentiality and integrity of stored information [5]. In addition, the Rivest–Shamir–Adleman (RSA) algorithm is used within the Controller layer to support secure key exchange and protect data during transmission [6]. By integrating encryption and decryption operations directly into the MVC workflow, the proposed approach ensures end-to-end protection of sensitive data throughout the application lifecycle.

Unlike traditional security add-ons, the proposed framework preserves the separation of concerns inherent to MVC. The Model layer is responsible for encrypted data storage and retrieval, the Controller manages cryptographic operations and secure data flow, and the View layer displays only decrypted data to authenticated users. This architectural integration minimizes the risk of accidental data exposure and enhances overall system security without significantly degrading performance.

The main contributions of this study include: (i) the design of a hybrid AES–RSA encryption framework tailored for MVC-based web applications; (ii) systematic integration of cryptographic protection at the architectural level; (iii) improved resistance against common web application attacks targeting sensitive data; and (iv) performance-aware implementation suitable for real-world deployment. Experimental evaluation demonstrates that the proposed approach strengthens data security while maintaining acceptable computational efficiency.

The remainder of this paper is organized as follows. Section II reviews related work on MVC security and cryptographic integration. Section III describes the proposed encryption-enhanced MVC architecture, implementation details and experimental setup. Section IV& V discusses the results and performance analysis. Finally, Section VI concludes the paper and outlines future research directions.

2. Related Work

Security in web applications has been a major research focus due to the increasing number of cyber threats and data breaches targeting sensitive user information. Early studies emphasized secure coding practices and vulnerability mitigation techniques to protect web applications from common attacks such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). Howard and LeBlanc [1] highlighted the importance of defensive programming and secure design principles to reduce exploitable vulnerabilities in software systems. While these approaches improve code robustness, they provide limited protection for sensitive data once a system is compromised.

Several researchers have investigated architectural security enhancements for web-based systems. The Model–View–Controller (MVC) architecture has been widely adopted due to its modularity and maintainability; however, its native design does not explicitly address data security concerns. Krasner and Pope [2] originally proposed MVC as a user interface paradigm without incorporating mechanisms for data confidentiality or cryptographic protection. As a result, subsequent studies explored supplementary security layers to compensate for these limitations.

Access control and authentication mechanisms have been

extensively studied as means of securing MVC-based applications. Role-Based Access Control (RBAC) models have been integrated into MVC frameworks to restrict unauthorized access to application resources [3]. Although such approaches improve authorization management, they do not ensure the confidentiality of sensitive data stored in databases or transmitted across application layers. Attackers gaining backend access may still retrieve data in plaintext form.

To address data protection concerns, cryptographic techniques have been applied in web application security. Stallings [4] emphasized the importance of encryption for securing data at rest and data in transit, particularly in distributed systems. Some studies proposed encrypting database fields containing sensitive information using symmetric encryption algorithms such as the Advanced Encryption Standard (AES) [5]. These approaches significantly improve data confidentiality but are often implemented as isolated solutions, lacking architectural integration with MVC workflows.

Hybrid encryption schemes combining symmetric and asymmetric cryptography have also been explored. Public-key algorithms such as Rivest–Shamir–Adleman (RSA) have been widely used for secure key exchange, while AES is employed for bulk data encryption due to its efficiency [6]. Research by Kumar and Singh [7] demonstrated that hybrid encryption improves security in web-based systems; however, their approach focused primarily on communication channels and did not address structured integration within MVC components.

More recent studies have proposed middleware-based or framework-specific security extensions to enhance MVC applications. These solutions typically introduce encryption modules at the controller or service layer [8]. While effective, such implementations may increase system complexity, reduce portability across frameworks, or impose performance overhead if not carefully designed. Furthermore, many existing solutions lack a clear separation of cryptographic responsibilities across MVC layers, which can negatively impact maintainability [9].

In summary, existing research has contributed valuable insights into web application security, access control, and cryptographic protection. However, most approaches either focus on isolated security mechanisms or fail to integrate encryption systematically at the architectural level of MVC. This gap highlights the need for a structured, hybrid encryption-based enhancement that preserves MVC design principles while providing comprehensive protection for sensitive data at rest and in transit—an issue addressed by the proposed approach in this study [10, 11].

3. Methodology

The methodology of this study follows a systematic approach to enhance security and privacy in Model–View–Controller (MVC)–based web applications through the integration of a data encryption-based model (Figure 2). The proposed methodology focuses on architectural modification, encryption mechanism design, and security evaluation while preserving MVC principles.

This research adopts a design-and-implementation–based methodology. A conventional MVC architecture was first analyzed to identify security and privacy limitations related to sensitive data handling. Based on these findings, an enhanced MVC architecture was designed by introducing an abstract encryption and decryption layer between the Model and the database.

3.1. Proposed Secured MVC Architecture

The enhanced architecture incorporates a dedicated encryption layer that operates transparently within the data access process. The Controller manages user requests and invokes Model functions. Before any sensitive data is stored or retrieved, the Model communicates with the encryption layer, which applies cryptographic operations. This approach ensures that encryption logic remains decoupled from application logic, improving maintainability and scalability.

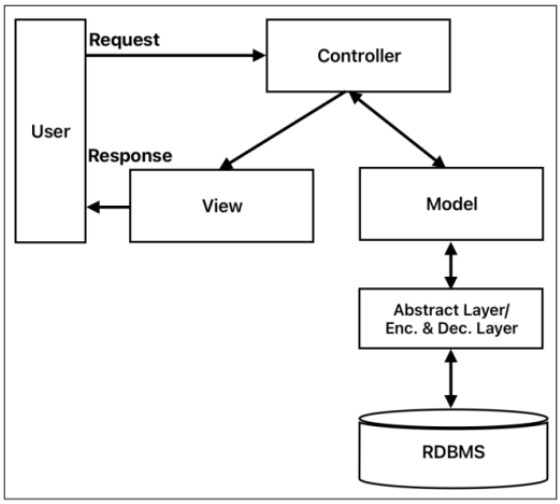

Figure 2. *Proposed MVC Architecture.*

A hybrid encryption approach was employed:

*AES-256*symmetric encryption was used to secure sensitive data stored in the database, ensuring confidentiality and integrity.

*RSA-2048*asymmetric encryption was utilized for secure exchange and protection of AES keys during data transmission.

This hybrid strategy combines the efficiency of symmetric encryption with the secure key management capability of asymmetric encryption.

Data Processing Flow: User requests are received by the Controller and validated before being forwarded to the Model. When data storage is required, sensitive fields are encrypted by the encryption layer prior to database insertion. For data

retrieval, encrypted values are decrypted before being passed to the Controller and rendered by the View. Throughout this process, plaintext data is never directly stored in the RDBMS.

3.2. Implementation

Algorithm-1: Secure Encryption–Decryption Workflow in MVC Model

Input: Plaintext data array D, secret key SK

Output: Encrypted database records and decrypted output data

Algorithm-1 Steps:

1) Generate a 256-bit encryption key by applying SHA-256 on the secret key SK.
2) For each data field in the input array:
   a) Generate a random 16-byte Initialization Vector (IV).
   b) Encrypt the data using AES-256-CBC with the generated key and IV.
   c) Concatenate the cipher text and IV and encode the result using Base64.
3) Store the encrypted values in the database.
4) During data retrieval:
   a) Decode the stored value from Base64.
   b) Separate the cipher text and IV.
   c) Decrypt the cipher text using AES-256-CBC.
5) Return the decrypted plaintext data to the application layer.

Algorithm-1: The proposed algorithm describes a secure encryption–decryption workflow integrated within the Model layer of the MVC architecture. AES-256-CBC is employed to ensure strong data confidentiality, while a unique initialization vector (IV) is generated for each data element to prevent cryptographic pattern analysis. The Initialization Vector (IV) is concatenated with the ciphertext and stored in encoded form within the same database field, enabling reliable extraction during decryption without requiring separate storage. The encryption process is applied prior to database storage, and decryption is performed only during authorized data retrieval. This design ensures that sensitive data remains protected at rest while maintaining transparency and modularity within the application logic.

3.3. System Requirements

This section specifies the hardware and software requirements necessary for the design, development, and evaluation of the proposed encryption-based Model-View-Controller (MVC) system. The environment was carefully configure d to ensure compatibility with crypto graphic operations, secure database interactions, and performance testing tools. The requirements are categorized into hardware and software components.

3.3.1. Hardware Requirements

The proposed system was implemented and tested on a workstation with the following minimum hardware specifications:

Processor: Intel Core i5 (8th Generation)
*Clock Speed:*2.40 GHz
*RAM:*8 GB
*Storage:*256 GB SSD

These specifications ensure adequate computational power for running cryptographic algorithms such as AES and RSA without significantly affecting response time or overall system performance.

3.3.2. Software Requirements

The software environment was designed to support PHP-based MVC frameworks and database-level encryption functions. The software components used in the development and testing phases are listed below:

Operating System: Ubuntu Linux 24 LTS
Web Server: Apache 2.4
Programming Language: PHP 8.3
MVC Framework: Codeigniter, Laravel, Falcon
Database: MySQL 8.0
Development Environment: Visual Studio Code
Testing Tools: ApacheJMeter for performance benchmarking

The above configuration provides an optimal environment for developing and evaluating the proposed encryption-enhanced MVC system. The combination of open-source frameworks, standard cryptographic libraries, and security testing tools ensures that the implementation remainscost-effective, reproducible, and scalable for future research or enterprise deployment.

4. Results

The experimental results evaluate the proposed encryption-based MVC model in terms of throughput, security effectiveness, and overall system performance, and compare it with a conventional non-encrypted MVC architecture as illustrated in Figures 3 to 5.

Figure 4 shows the throughput comparison under different concurrent user loads. The non-encrypted MVC consistently achieves higher throughput across all scenarios. At 5 concurrent users, the non-encrypted MVC processes approximately 218 requests/sec, while the encrypted MVC achieves around 201 requests/sec. As concurrency increases to 25 users, throughput decreases for both models; however, the encrypted MVC maintains stable performance with approximately 161 requests/sec compared to 175 requests/sec for the non-encrypted version. This indicates a throughput reduction of roughly 8–10%, attributable to encryption and decryption overhead, but still within acceptable operational limits.

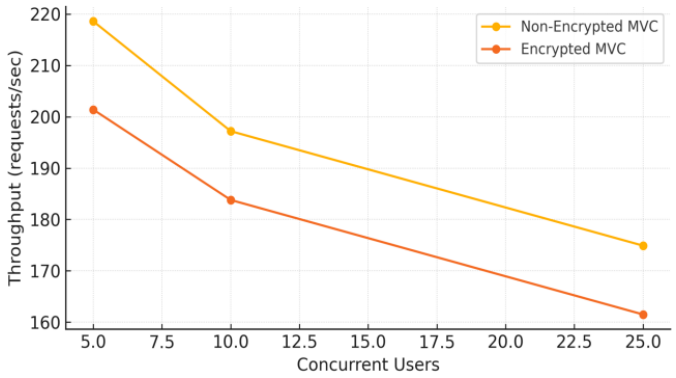

Figure 3. *Throughput comparison of Encrypted vs Non-encrypted MVC model.*

Figure 4 demonstrates a significant improvement in security-related metrics for the proposed encryption-based MVC model. The encrypted MVC scores notably higher in security, data confidentiality, and compliance compared to the traditional MVC model. In particular, data confidentiality approaches the maximum score, confirming the effectiveness of AES-based encryption for protecting sensitive data. The overall weighted average score further highlights the superior security posture of the proposed model.

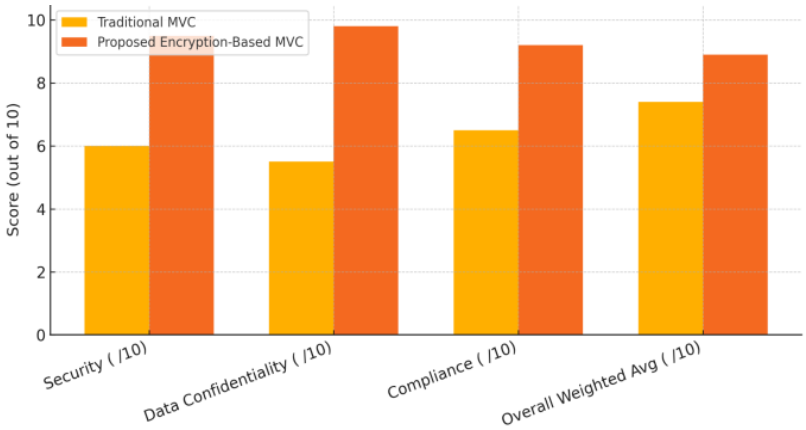

Figure 4. *Security Effectiveness comparison between MVC Model.*

Figure 5 compares average query execution time, throughput, and CPU utilization. The encrypted MVC shows a slight increase in average query time and CPU usage, reflecting the cost of cryptographic operations. Despite this, system throughput remains relatively high, and resource utilization stays within acceptable limits.

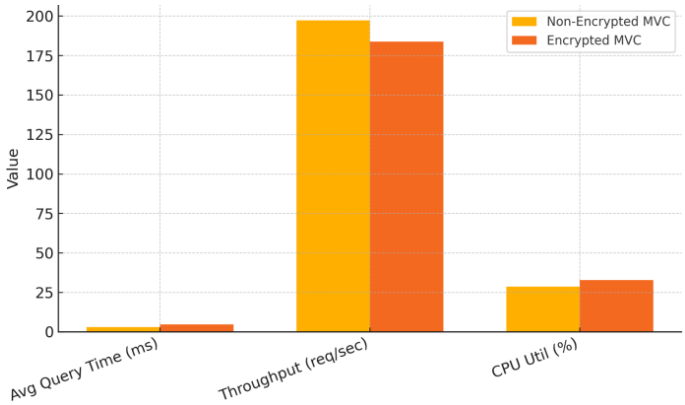

Figure 5. *Performance comparison between Encrypted vs Non-encrypted MVC Model.*

5. Discussion

The experimental results demonstrate that the proposed encryption-based MVC model introduces a modest performance overhead while significantly strengthening system security. As shown in Figure 3, the encrypted MVC exhibits lower throughput compared to the non-encrypted model under increasing concurrent users. This behavior is expected due to the computational cost of encryption and decryption operations. However, the throughput degradation remains limited and scales predictably, indicating that the system preserves acceptable scalability.

Figure 4 highlights a substantial improvement in security effectiveness, particularly in data confidentiality and compliance. These results confirm that integrating encryption at the architectural level provides stronger protection than traditional MVC implementations that rely mainly on validation and access control.

Figure 5 further reveals that although average query time and CPU utilization increase slightly in the encrypted MVC, the overall system performance remains within acceptable limits. This indicates a well-balanced trade-off between security and efficiency.

Overall, the discussion confirms that the proposed system successfully prioritizes data protection without significantly compromising performance, making it suitable for real-world secure web applications.

6. Conclusions

This study presented a security-enhanced Model–View–Controller (MVC) architecture to address persistent security and privacy challenges in modern web applications [12, 13]. Traditional MVC-based systems are vulnerable to common attacks such as SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), session hijacking, and inadequate protection of sensitive data [14]. To mitigate these risks, the research proposed a hybrid encryption-based security model integrated directly into the MVC architecture.

The proposed approach utilizes Advanced Encryption Standard (AES-256) to secure sensitive data stored in the database, ensuring confidentiality and integrity of data at rest. In addition, Rivest–Shamir–Adleman (RSA) cryptography is employed to protect AES keys during transmission, enabling secure data exchange and safeguarding data in transit. These encryption mechanisms were systematically embedded within the MVC framework, where the Controller manages encryption and decryption processes, the Model handles encrypted data storage and retrieval, and the View displays only decrypted information to authenticated users [15].

Security testing and evaluation demonstrated that the proposed architecture effectively reduces exposure to common web application vulnerabilities while maintaining acceptable performance. The findings confirm that integrating hybrid encryption into the MVC framework strengthens application security without compromising modularity or scalability. Overall, this research provides a practical and reproducible secure MVC implementation model, bridging the gap between theoretical security concepts and real-world web application development, and offering a robust foundation for building privacy-preserving web systems.

Abbreviations

MVC	Model View Controller
SQL	Structured Query Language
XSS	Cross-Site Scripting
CSRF	Cross-Site Request Forgery
RSA	Rivest–Shamir–Adleman (RSA) Cryptography
AES	Advanced Encryption Standard
RBAC	Role-Based Access Control

Conflicts of Interest

This work has no conflicts of interest in terms of financialand interpersonal from any of the authors.

References

[1] Naimnule, F. A., Hanoe, F. A., Banusu, M. N. and Mano, M. O., Implementation of AES Encryption for Data Security on Web-Based Information Systems in FafinesuA Village. Krisnadana Journal, 4(3), pp. 122-130. https://doi.org/10.58982/krisnadana.v4i3.836

[2] Pratiwi, Ade. "Implementation of Modified MVC Model with Integrated Security in E-Procurement Application for Companies." Journal of Technology and Computer 2, no. 3 (2025): 176-183.

[3] M. H. Rahman, M. Naderuzzaman, M. A. Kashem, B. M. Salahuddin, Z. Mahmud, "Comparative Study: Performance of MVC Frameworks on RDBMS", International Journal of Information Technology and Computer Science(IJITCS), Vol. 16, No. 1, pp. 26-34, 2024. https://doi.org/10.5815/ijitcs.2024.01.03

[4] M. H. Rahman, Bin, F., Naderuzzaman, M., Arifur, M., and Masud, M., “Optimizing and Enhancing Performance of MVC Architecture based on Data Clustering Technique”, International Journal of Computer Applications, vol. 134, no. 12, pp. 42-46, 2016. https://doi.org/10.5120/ijca2016908099

[5] Egerton, Taylor Onate, and Davies Isobo Nelson., A Model for Enhancing Security and Privacy in Pervasive Computing using Homomorphic Encryption, 2025. https://doi.org/10.26438/ijcse/v13i8.2129

[6] Shah, Parth, Samarth Shah, and Anurag Agrawal. "Advanced Encryption Techniques for Enhancing Data Security and Pri-

vacy in Cloud Environments." 2025 1st International Conference on Secure IoT, Assured and Trusted Computing (SATC). IEEE, 2025. https://doi.org/10.1109/SATC65530.2025.11137337

[7] Naik, Apurva R., and Lalit B. Damahe. "Enhancing data security and access control in cloud environment using modified attribute based encryption mechanism." International Journal of Computer Network and Information Security 8.10 (2016): 53. https://doi.org/10.5815/ijcnis.2016.10.07

[8] OWASP Foundation, "OWASP Top Ten Web Application Security Risks," 2023. Available: https://owasp.org/www-project-top-ten/

[9] Egerton, Taylor Onate, and Davies Isobo Nelson., A Model for Enhancing Security and Privacy in Pervasive Computing using Homo-morphic Encryption, 2025. https://doi.org/10.26438/ijcse/v13i8.2129

[10] Y. Jiang, M. A. Rezazadeh Baee, L. R. Simpson, P. Gauravaram, J. Pieprzyk, T. Zia, et al., "Pervasive user data collection from cyberspace: Privacy concerns and counter meaures," Cryptography, Vol. 8, Issue. 5, pp. 1-5, 2024. https://doi.org/10.3390/cryptography8010005

[11] E. I. Egho-Promise, M. Sitti, "Big data security management in digital environment," American Journal of Multidisciplinary Research & Development (AJMRD), Vol. 6, pp. 01-34, 2024. http://ajmrd.com/wp-content/uploads/2024/02/ A620134.pdf

[12] E. Mollakuqe, A. Parduzi, S. Rexhepi, V. Dimitrova, S. Jakupi, R. Muharremi, et al., "Applications of Homomorphic Encryption in Secure Computation,"Open Research Europe, Vol. 4, pp. 158, 2024. https://doi.org/10.12688/openreseurope.18052.1

[13] A. K. Y. Yanamala, S. Suryadevara, "Adaptive Middleware Framework for Context-Aware Pervasive Computing Environments,"International Journal of Machine Learning Research in Cybersecurity and Artificial Intelligence, Vol. 13, pp. 35-57, 2022. https://ijmlrcai.com/index.php/Journal/article/view/28

[14] S. D. Pasham, "Privacy-preserving data sharing in big data analytics: A distributed computing approach,"The Meta science, Vol. 1, pp. 149-184, 2023. https://yuktabpublisher.com/index.php/TMS/article/view/130/118

[15] S. Aswathy, A. K. Tyagi, "Privacy Breaches through Cyber Vulnerabilities: Critical Issues, Open Challenges, and Possible Countermeasures for the Future,"In the Security and Privacy-Preserving Techniques in Wireless Robotics, ed: CRC Press, pp. 163-210, 2022.