Research Article

Tlade+: Intelligent Adaptive Encryption Using Reinforcement Learning For Secure Cloud Services

K.C. Nithyasree[1], Nidheshwar P[2], Karthik B[3], Sivagnanam R[4], Rithikka V T[5], Raghavi Sree S[6], Angel Rani A[7]

[1-7] Department of CSE (IoT and Cyber security including Block chain Technology), Manakula Vinayagar Institute of Technology, Puducherry, India, Email id: nithyasreekcr@gmail.com

Abstract:

Cloud computing can offer efficient data storage and sharing solutions. However, securing sensitive data is a major concern. Traditional encryption methods lack adaptability and can cause extra overhead and poor security. Our solution is the Intelligent Adaptive Encryption system called TLADE+ (Time-oriented Latency Approximation-based Data Encryption Plus). Q-learning and Decision Tree techniques are used to determine appropriate encryption levels. The Q-learning agent takes into account how sensitive the file is, the file size, system latency, CPU Usage and threat level. Based on this it will decide whether to apply LOW, MEDIUM or HIGH levels of encryption. Rules have been set up so that if the file is very sensitive, or the threat levels are high, then LOW encryption can never be applied. After encryption of the file a reward will be calculated based on how secure, quick and how it fits within the systems requirements. This reward will be used to update the Q-table. The system also adapts by storing key features of the file and what encryption label was chosen. If there are enough entries with valid data then the Decision Tree can be retrained. The system also includes a Flask authentication system, OTP verification, encrypting files, storing them onto IPFS and viewing the history of the file. This system allows intelligent adaptive protection of files.

***Keywords**: Cloud Security, TLADE+, QoS, Encryption, AES, ChaCha20, Decision Tree, Latency.*

I. Introduction

This paper proposes a novel advanced encryption system called TLADE+ (Time-oriented Latency Approximation-based Data Encryption Plus). With increasing amounts of data in cloud computing environments, traditional encryption systems no longer offer an efficient solution. This is because traditional encryption systems encrypt all data with the same level of encryption regardless of how sensitive it may be, user role, or system state. This can cause increased processing overhead, increased latency, decreased throughput etc. and negatively impacts the performance of the cloud service. The TLADE+ system uses intelligent adaptive encryption to dynamically adapt to the sensitivity of the data, user role, and system state to determine what level of encryption should be used. Very sensitive data is encrypted with strong encryption techniques whereas less sensitive data is encrypted with lightweight encryption to decrease latencies. This novel system uses machine learning to observe historical and real-time data relating to QoS. This includes latency, variation in traffic, available bandwidth, and server load. The system can predict what encryption method and route would be best for the request. This means that the system can always optimise security and performance. Multiple encryption methods are supported, including ChaCha20, AES, AES-GCM. The best method is selected based on the current requirements. The experimental results show that this system reduces the computational cost, latency, increases throughput and increases confidentiality. Due to its adaptive nature, this system is well-suited to storage within enterprise clouds, real-time applications, distributed systems and data intensive systems

II. Literature survey

Nitesh Bharot, Nakul Mehta, John G. Breslin, Priyanka Verma [1] In this paper we have put forward a new framework called Cloud Lock which helps to provide secured storage of information to users. This framework mainly focuses on providing better confidentiality, privacy and access control. To secure the data being stored ChaCha20 and Poly1305 was used. The Elliptic Curve Diffie-Hellman (ECDH) key exchange is used to securely share the secret keys between the data owner and registered users. This ensures that the data cannot be accessed by the wrong users. This framework can securely send the data between different clouds. A thorough analysis was carried out to improve performance. This included testing the encryption/decryption speed, upload/download speed, turn around time and memory used. We have found that Cloud Lock performs better than existing solutions. We have also shown that this system can withstand many security threats and can be accessed by legitimate users.

Vikas K. Soman & V. Natarajan [2] In CFOPS-BSBEA (Crayfish Optimization-based Pixel Selection with Block Scrambling Based Encryption Approach), secure encryption method is used to secure digital images. Here, first Pixel Selection is done using the Crayfish Optimization algorithm, followed by the Block Scrambling Based Encryption method. The encrypted data is then concealed inside the cover image using steganography. Thus,

using multiple methods makes it difficult for attackers to gain unauthorized access to the image, and reduces the chances of the image being tampered with, as well as making it harder to see the concealed information. Mamta Dabra , Shivani Sharma , Sachin Kumar & Hong Min [3] This research presents an ephemeral keyword search technique using ciphertext-policy to increase security of sensitive health care data held within a cloud system. As a cloud storage provider cannot always be seen as fully trusted, health care data is encrypted prior to being sent to external storage. Traditional attribute based searching techniques can lead to sensitive information being exposed as they allow for more encrypted records to be retrieved than should be available. The proposed scheme ensures that search tokens can only be used during specific periods of time, therefore reducing the chance of unauthorised access and insider threats. It has been formally proven to be secure against Selectively Chosen Keyword Attacks under the generic bilinear group model. This scheme uses linear complexity with respect to attributes. Evaluation shows that the scheme is viable within a health care scenario where secure, ephemeral and efficient data retrieval is required. S. Velmurugan, M. Prakash, S. Neelakandan and Arun Radhakrishnan [4] Cloud platforms continuously produce huge volumes of information and therefore must be able to efficiently collect and share data. Data owners can share information with specific people whilst deciding who has access to each file. However, this can cause issues with keeping information private and confidential. A key issue with sharing information is the risk that information stored within the cloud will be shared with people who do not have permission to view it. Current methods struggle to provide an effective balance between access controls, data privacy and user friendly methods. As there is an increasing need to protect personal information, communication across platforms and increased volumes of data stored within clouds, it is important to develop effective and secure data management systems. Danial Shiraly, Ziba Eslami and Nasrollah Pakniat [5] This paper introduces a novel scheme, named Hierarchical Identity-Based Authenticated Encryption with Keyword Search (HIBAEKS), which enables secure Keyword Search and solves privacy and efficiency issues in cloud storage. Compared to traditional HPEKS schemes, our scheme eliminates bilinear pairings and reduces computation cost. Moreover, this scheme provides resistance to keyword guessing attacks. The scheme enables encrypted data to be searched using keywords according to hierarchical access control policies to prevent insider attacks related to keywords. A new security model is defined, and strong security properties of HIBAEKS are proved. Radhakrishna Bhat N. R. Sunitha S. S. Iyengar[6] A probabilistic public-key encryption scheme is presented that can be used for homomorphic and non-malleable computing in secure cloud storage systems. The scheme combines both features into a single scheme and thereby can be used for a variety of different cloud applications. The ciphertext has a size of O (m + 2 log N) where m denotes the size of the plaintext and N denotes the RSA composite. This makes the scheme fast and scalable. Due to its two features secure computations can be performed on the encrypted data without the danger of tampering. It will be shown that our scheme outperforms the current state of the art in terms of speed and ciphertext length..

III. Proposed system

The proposed system (TLADE+) provides an intelligent adaptive encryption system for secure file protection. The system will adaptively choose an appropriate encryption level by observing the current characteristics of the file and system. When a user uploads a file, several variables are observed including the sensitivity of the file, file size, system latency, CPU usage, and threat level. These variables are used to determine the current state of the system. The state is passed to the Q-learning agent which will use it alongside its learned Q-table to determine the correct encryption levels. This will result in the encryption being LOW, MEDIUM, or HIGH. A security rule is applied so that the lowest security can still meet the requirements of the system. For example, if the file is highly sensitive or the threat level is high then the encryption cannot be LOW. The reward will then be calculated based on the security, speed, and requirements of the system. The reward will then be used to update the Q-table. The system also contains a Decision Tree so that it can adapt to new files. It will record the variables observed, and the encryption label. Once there is enough valid data the system will retrain the Decision Tree. This system contains a Flask based authentication system, OTP verification, file encryption, and file history. This means that the system uses reinforcement learning, supervised learning, cryptography, and decentralized storage so that it can adaptively provide secure file protection.

Architecture diagram:

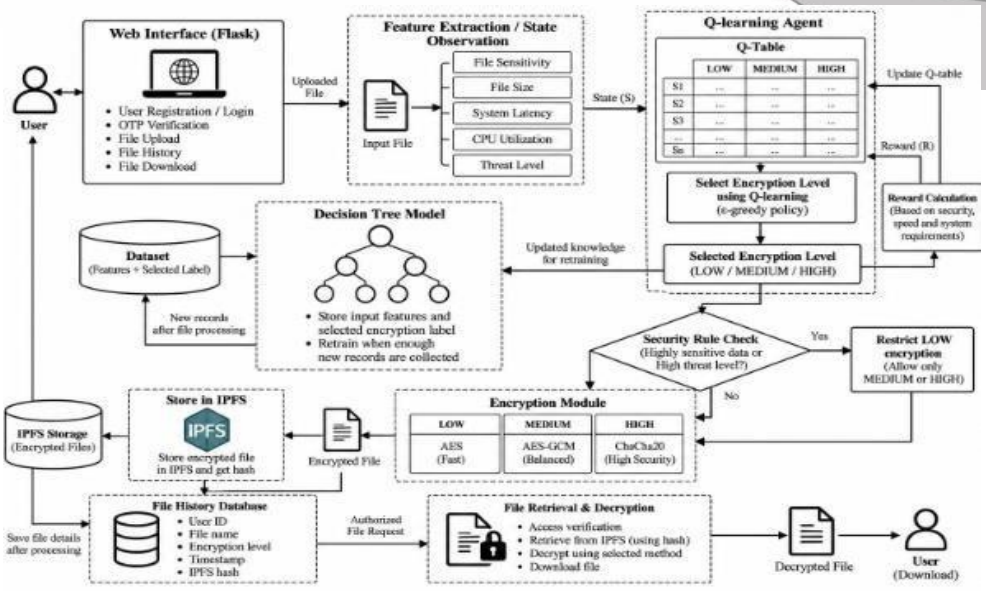


Fig 3.1 Architecture diagram of the proposed system

The TLADE+ system has multiple parts working together to provide an adaptive encryption system that securely stores files and allows authorized access to files. The system begins with the Flask-based authentication system. This allows users to register, login, complete OTP verification, upload files, view file history and download authorized files. Once a file has been uploaded, the Feature Extraction/State Observation module will extract the required information. This includes sensitive file, file size, system latency, CPU usage, and threat level. This information creates the current state of the system and is sent to the Q-learning agent. The Q-learning agent will use its Q-table to decide what level of encryption should be used, either LOW, MEDIUM or HIGH. A security rule will be applied to ensure that if the file is very sensitive or the threat level is high then LOW encryption cannot be used. The chosen encryption level is sent to the encryption module. The encryption module will encrypt the file using the relevant encryption method (AES, AES-GCM, ChaCha20). The encrypted file is then stored within IPFS. The metadata and IPFS hash are stored within the file history database. The Q-learning agent will receive a reward based on the security, processing speed, and system requirements. This will be used to update the Q-table. Input variables and encryption labels will be stored within a database. If enough new records are available, the Decision Tree Model will be retrained. This allows the system to adapt to new files. Authorized users can view and decrypt files using the IPFS hash.

a. User Authentication and File Management Module

The User Authentication and File Management Module is used to provide a secure way to access the TLADE+ system. This module is based around a Flask-based system that allows users to register, login, upload files, view file history and request authorized downloads. Once valid login details have been entered, OTP verification must be completed. Once the user has been authenticated, the user can upload a file to be adaptively encrypted. This module also stores information regarding user activity and uploaded files. When requesting to view a file, the system will verify that the user has access. This module acts as a communication layer between the user and other parts of the TLADE+ system. This will ensure that only authorized users can perform sensitive tasks such as uploading files.

b. Feature Extraction and State Observation Module

The Feature Extraction and State Observation Module is used to observe the conditions needed for intelligent encryption. When a file is uploaded by a user, the system will collect key information about the file. This includes the sensitive file, file size, system latency, CPU usage, and threat level. The sensitive file variable indicates how important/confidential the information is. The file size demonstrates what level of processing is needed. The system latency and CPU usage indicates the current processing conditions. The threat level indicates the estimated security risk. This information is used to create the current state (S). This will affect how the encryption is decided as the Q-learning agent will use this information to determine what action to take.

c. Q-Learning adaptive Encryption Module

This module is the main intelligent part of the TLADE+ system. This will dynamically decide what level of encryption to use based on the observed system state. The Q-learning agent will receive the state variables of

sensitive file, file size, system latency, CPU usage, and threat level. The agent will use its Q-table to decide what action to perform; LOW, MEDIUM, or HIGH encryption. Initially, the agent will explore different actions and learn how effective they are. Over time, the Q-table will show what encryption action is more suited to different states. This will help the system find a balance between security requirements and processing efficiency. This module can be very useful as the system state can change over time meaning the encryption decision will always be improving as the agent learns from previous file processing.

d. Security Rule and Encryption Module

This module ensures that the Q-learning agent decides on the correct encryption decision that meets the minimum-security requirements. While the Q-learning agent will dynamically decide what level of encryption to use there is a security rule that will ensure the decision made is safe. If the file is very sensitive or the threat level is high then the system will not allow LOW encryption to be used. Only MEDIUM and HIGH encryption will be allowed. Once the final encryption decision has been made, the file will be encrypted using the relevant encryption algorithm. Depending on the system settings, the
LOW, MEDIUM, and HIGH encryption levels can be mapped to different algorithms such as AES, AES-GCM, ChaCha20. This will encrypt the original file.

e. Reward Calculation and Q-Table Update Module

This module allows the Q-learning agent to learn from the encryption. Once the encryption action has been performed, the system will evaluate how well the encryption went based on the security, processing speed, latency, and system requirements. Based on these conditions, a reward will be calculated. If the encryption was a good choice and provides adequate security with little processing overhead then a higher reward can be given. If the encryption was not a good choice, then a lower reward will be given. The reward will be used to update the Q-table based on the Q-learning update mechanism. This will give the Q-value an indication of how useful the encryption action was. Over time, the agent will understand better how the system state affects the encryption decision.

f. Decision Tree Learning and Model Update Module

The Decision Tree Learning and Model Update Module provides an additional supervised learning mechanism within TLADE+. After each valid file-processing operation, the system records the observed input features and the encryption level selected by the system. These records contain information such as file sensitivity, file size, system latency, CPU utilization, threat level, and selected encryption label. The collected records are stored in a dataset that gradually grows as more files are processed. When a sufficient number of valid records are available, the Decision Tree model is retrained using the updated dataset. This enables the model to learn from newly generated examples rather than depending only on the initial training data. The Decision Tree can identify relationships between system conditions and encryption-level decisions and can provide an interpretable classification structure. Therefore, the Decision Tree complements the Q-learning component by providing supervised learning from accumulated operational data and supporting continuous improvement of the overall adaptive encryption framework.

g. IPFS Storage and File History Module

This module securely stores encrypted files. Once the encryption method has been decided upon, the encrypted file will be uploaded to the Interplanetary File System (IPFS). A unique identifier/hash will be created. This can be used to find the encrypted file. The original file does not need to be stored as the file is encrypted. The IPFS hash can be stored along with important metadata such as user identification, file name, encryption level, timestamp, and any file processing information. This can be used to track and find encrypted files. This module uses decentralized content addressed storage to securely store the encrypted content.

h. Authorized File Retrieval and Decryption Module

The Final stage of the TLADE+ workflow is handled by the Authorized File Retrieval and Decryption Module. This module will validate the user requesting the file, then search the history to find the requested file. Once this has been found the IPFS hash will be used to find the encrypted file within IPFS. Next the encryption level will be determined and the correct decryption method applied using the correct cryptographic key and setup. Now the encrypted content will be decrypted back into the original content. The file is also made available to download through the flask interface, however, only to those authorized. This ensures that the files are only accessible and that the encrypted data stored in IPFS cannot be viewed directly. This completes the secure workflow from uploading the file, applying adaptive encryption, storing the file securely, retrieving the file and decryption the file.

IV. Result and Discussion

The system has shown significant improvements in security and QoS within the cloud computing environment. The adaptive encryption has reduced overhead within the system. Comparing the encryption with traditional methods showed a decrease in latency and an increase in processing speed. This is due to the Decision Tree

allowing the system to select the best encryption method to use, therefore increasing the throughput within the system. Testing the system using synthetic data has shown the model can adapt to new conditions. The decision tree has been accurate in selecting the correct encryption configurations. The system has also responded well to high traffic with the delays minimized. Machine learning has helped the system become intelligent. By analyzing the historic and real-time data using TLADE+, the system can easily be scaled due to the modular components. The performance of the system may depend on the quality of the training data. Overall, however, this confirms the TLADE+ provides a reliable and efficient solution.

a. Advanced Encryption Standard (AES)

AES is a symmetric key encryption algorithm that provides a very efficient and reliable way of encrypting sensitive data. It has become a standard due to its adoption by the National Institute of Standards and Technology. It splits the data into fixed size blocks and encrypts it with a 128, 192 or 256 bit key. Multiple transformations are applied to the data including substitution, permutation, mixing and key addition. This makes the encryption secure. AES is quick and efficient and can be implemented in hardware and software. This ensures that the data remains confidential by using a secret key to encrypt and decrypt. Due to the high levels of security provided and low requirements in computation, AES is widely used in applications such as secure data storage.

b. Addroundkey Step

The Advanced Encryption Standard (AES) is a symmetric cipher which encrypts data by using a secret key to change plain-text to cipher-text. The data is broken down into fixed-size blocks of 128 bits. Depending on the size of the key (128, 192 or 256 bits) there will be multiple rounds of transformations. The algorithm has many parts including SubBytes (substitution), ShiftRows (shifting rows), MixColumns (mixing columns) and AddRoundKey (adding round key). This makes it very secure as it provides good confusion and diffusion. One of the most important operations in AES is the AddRoundKey step, where the data block is combined with the round key using a bitwise XOR operation:

$$C = P \oplus K$$

Here, P represents the plaintext (or intermediate state), K is the round key, and C is the resulting ciphertext. This step ensures that the encryption is dependent on the key, making it difficult to decode without the correct key. Another key operation is the Mix Columns transformation, which enhances diffusion by mixing the bytes within each column using matrix multiplication in finite field arithmetic:

This transformation spreads the influence of each byte across the column, increasing security.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Overall, AES combines these mathematical operations to provide strong, efficient, and reliable encryption widely used in modern systems. In addition to AddRoundKey and MixColumns, AES uses two other important operations called SubBytes and ShiftRows to improve the security of the encryption process. In the SubBytes operation, each byte of the input data is replaced with a different byte using a predefined substitution table called the S-Box. This replacement helps hide the original data pattern and provides confusion. The ShiftRows operation then shifts the bytes in each row of the data block by different positions. This rearrangement ensures that the bytes are distributed across different columns during the next encryption steps. Both operations work together to change the structure of the original data, making it difficult for unauthorized users to understand or recover the plaintext. AES performs these operations repeatedly in multiple rounds to provide a high level of data protection. The number of rounds depends on the selected key size, with 10 rounds for AES-128, 12 rounds for AES-192, and 14 rounds for AES-256. During each round, the data undergoes different transformations, which gradually convert the original plaintext into unreadable ciphertext. The final result is a secure encrypted form of data that can only be converted back into its original form using the correct secret key through the decryption process. Because AES provides a good balance between security and performance, it is widely used in modern technologies such as online banking, cloud computing, wireless networks, secure file storage, and digital communication.

c. AES-GCM (Advanced Encryption Standard – Galois/Counter Mode)

AES-GCM is an authenticated encryption algorithm. It uses two components, encryption and authentication. It uses an encryption technique known as Counter Mode and an authentication technique known as Galois Hashing. This makes it very efficient and also secure. AES-GCM is a great choice for cloud computing and real time applications. The first component is encryption, this uses a Counter Mode encryption. Counter Mode encryption (CTR) is performed by encrypting the plaintext using a keystream generated by the counter and secret key. The encryption can be defined as:

$$C_i = P_i \oplus E_K(\text{CTR}_i)$$

Here, Pi is the plaintext block, Ci is the cipher text, EK represents AES encryption with key K, and CTRi is the counter value for each block. This approach ensures fast parallel encryption and high performance. The second layer is the authentication layer, which ensures data integrity using Galois Field multiplication. It generates an authentication tag that verifies whether the data has been altered. The tag is computed using:

$$T = \text{GHASH}(H, A, C)$$

In this equation, T is the authentication tag, H is a hash subkey derived from AES, A represents additional authenticated data (AAD), and C is the ciphertext. The GHASH function uses polynomial multiplication in a finite field to produce a secure tag. Together, these two layers ensure that AES-GCM not only encrypts data efficiently but also guarantees its authenticity and integrity. This makes it widely used in secure communication protocols and cloud security systems.

d. Encryption Layer (Chacha20 Stream Generation)

XChaCha20 is an advanced stream cipher designed to provide high security, extended nonce support, and efficient performance. It improves upon ChaCha20 by using a larger nonce (192 bits), making it more secure for large-scale and distributed systems such as cloud environments. XChaCha20 mainly consists of two important layers: the key derivation layer (HChaCha20) and the encryption layer (ChaCha20 stream generation). The first layer is the key derivation layer, where a subkey is generated using the HChaCha20 function. This step takes the original key and part of the extended nonce to produce a new secure subkey:

$$K' = \text{HChaCha20}(K, N_{0..15})$$

Here, K is the original key, N0...15 represents the first 16 bytes of the nonce, and K'is the derived subkey. This layer ensures enhanced security by preventing nonce reuse issues and enabling safe encryption over large datasets.The second layer is the encryption layer, where the ChaCha20 algorithm uses the derived subkey K' along with the remaining nonce and a counter to generate a keystream. The plaintext is then encrypted using XOR operation:

$$C = P \oplus \text{ChaCha20}(K', N_{16..23}, \text{counter})$$

In this equation, P is the plaintext, C is the ciphertext, and the function generates a pseudorandom keystream. This layer ensures fast, secure, and parallel encryption. Together, these two layers make XChaCha20 highly secure, efficient, and suitable for modern applications requiring strong encryption with extended nonce support.

e. Time Efficiency

The time efficiency of encryption in systems like AES, AES-GCM, and XChaCha20 can be generally represented using a unified expression based on data size and processing speed. Since all these algorithms process input data to generate encrypted output, the total execution time mainly depends on how fast the algorithm can handle the given data. This can be expressed as:

$$T = \frac{n}{r}$$

Here, T represents the total encryption time, n is size of the input data (in bytes or bits), and r is the processing rate of the algorithm (i.e., how many bytes are encrypted per unit time). This formula shows that encryption time increases linearly with the size of the data and decreases as the processing speed improves. In practical scenarios, different algorithms have different processing rates. AES involves multiple transformation rounds, which may slightly increase processing time. AES-GCM improves efficiency by enabling parallel processing, thus increasing the effective rate. XChaCha20, being a stream cipher, operates with simple arithmetic operations, resulting in a very high processing rate and reduced latency. Despite these differences, the overall time behavior of all these algorithms follows the same linear relationship. This unified time model is useful for comparing encryption performance in cloud environments, where minimizing latency and maximizing throughput are critical. It highlights the importance of selecting algorithms with higher processing rates to achieve faster and more efficient data encryption.

f. Speed of Algorithms

The speed of encryption algorithms such as AES, AES-GCM, and XChaCha20 is a crucial performance metric, especially in cloud computing environments where fast data processing is essential. Speed generally refers to how quickly an algorithm can encrypt a given amount of data, and it is directly related to the processing rate of the system. A common way to represent encryption speed is through the relationship between data size and time taken for encryption. This can be expressed using the formula:

$$S = \frac{n}{T}$$

Here, S represents the speed of encryption, n is the size of the data (in bytes or bits), and T is the total time required to encrypt that data. This formula shows that speed increases when more data is processed in less time, indicating better performance. In practical scenarios, algorithms with simpler operations and support for parallel execution tend to achieve higher speeds.

For example, stream-based approaches can process data continuously, while parallel modes can handle multiple blocks simultaneously, improving throughput. The formula highlights that optimizing either the processing time or increasing the data handling capacity will directly enhance encryption speed.

g. Latency

Latency refers to the delay between initiating a data encryption process and obtaining the final encrypted output. In cloud computing and secure communication systems, latency is a critical factor because it directly affects system responsiveness and Quality of Service (QoS). High latency can lead to slower data transmission, increased waiting time, and reduced user experience, especially in real-time applications. Latency in encryption systems is influenced by several factors, including computational complexity, data size, network conditions, and system workload. Latency can be mathematically represented as:

$$L = T_{proc} + T_{trans} + T_{queue}$$

Here, L represents the total latency, Tproc is the processing time required for encryption, Ttrans is the transmission time for sending data across the network, and Tqueue is the waiting time in system queues due to workload or congestion. This formula shows that latency is a combination of multiple delays occurring at different stages of data handling. In encryption algorithms, reducing processing time through efficient computation can significantly lower overall latency.

Similarly, optimizing network conditions and minimizing queue delays further improves performance. In adaptive systems like TLADE+, latency is dynamically minimized by selecting efficient encryption strategies based on real-time conditions.

h. Sensitivity

Sensitivity is an important performance metric used to evaluate the effectiveness of a system, particularly in classification models such as the Decision Tree used in the TLADE+ framework. It measures the ability of the model to correctly identify positive instances, meaning how well the system detects relevant or critical cases (e.g., correctly identifying high-priority or sensitive data that requires stronger encryption). High sensitivity indicates that the system rarely misses important cases, which is essential for maintaining strong security in cloud environments.

Sensitivity is mathematically defined as:

$$\text{Sensitivity} = \frac{TP}{TP+FN}$$

Where TP refers to the True Positives and FN refers to the False Negatives. This can be seen as a measure that aims to reduce the amount of missed detections. This means that having a high sensitivity will mean that sensitive data will be correctly identified and therefore receive a higher level of encryption in order to protect it. However, if sensitivity is low then it could mean that critical data is being incorrectly identified and therefore will have a lower level of protection. Ultimately, increasing sensitivity will increase the reliability of the system.

i. Q-Learning Agent

The Q-learning agent is the main reinforcement learning component of the proposed TLADE+ system. It is responsible for dynamically selecting an appropriate encryption level based on the current file and system conditions. The agent observes a state consisting of parameters such as file sensitivity, file size, system latency, CPU utilization, and threat level. Based on the current state, the agent selects an action, where the available actions are LOW, MEDIUM, or HIGH encryption. After the selected encryption level is applied, the system evaluates the result and generates a reward according to security, processing speed, latency, and system requirements. The agent then updates its Q-table using the obtained reward. Over repeated interactions, the Q-

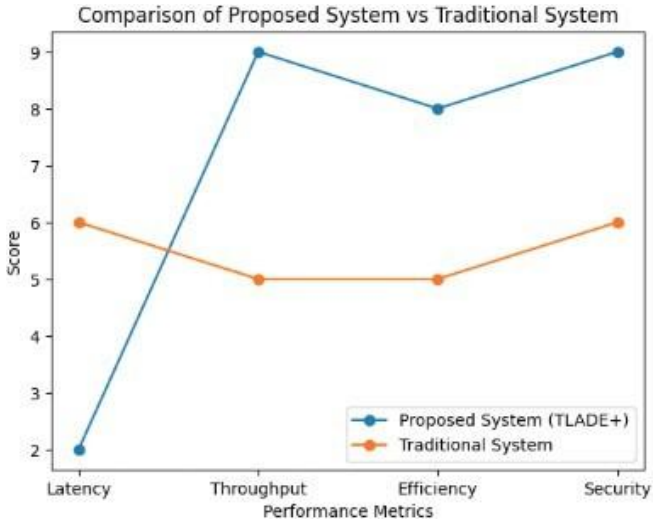
$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

table learns which encryption action provides the best balance between security and computational efficiency for different system states. A security rule is also incorporated to prevent LOW encryption when the file is highly sensitive or the threat level is high.

Where, Q(s,a) is the current Q value, is the learning rate, γ is the reward, r is the discount factor, s' is the next state and a' is the potential next actions. This helps improve the encryption selection by using this continuous learning process on more files.

j. Comparison Graph

The comparison graph shows how the proposed TLADE+ system performs compared to a traditional encryption system. From the graph, we can see that the proposed system has much lower latency. This means the data can be processed quicker. The traditional system has a higher latency meaning it will take longer to respond. TLADE+ has shown significant improvements in throughput. This means a larger amount of data can be processed in a shorter amount of time. The proposed system also has a higher efficiency as the encryption is adapted. Security has also been shown to improve significantly as the encryption level is intelligently adapted..



k. Decision Tree

The Decision Tree prediction algorithm is an important part of the proposed system. After training using synthetic data and preprocessing, they learn patterns from various factors such as data sensitivity, user roles, network latency, bandwidth and system load. In the prediction stage, once new data is provided, the Decision Tree algorithm takes in these factors and uses them to predict what action to take. The Decision Tree is built with a root node, decision nodes and leaf nodes. Once data is provided, the Decision Tree will begin at the root node and pass through all of the decision nodes using the rules that have been learnt from training. Eventually it will reach a leaf node where it will provide the final decision. The model will decide the best encryption method based on the current situation of the system. If data sensitivity was high and the network load was medium then a strong encryption method would be chosen. However, if data sensitivity was low and there was a large amount of traffic then a fast/lightweight encryption method would be selected. The prediction stage is very quick and efficient, making it perfect for real time situations. There are several metrics that can be used to help evaluate how well the proposed system performs. These include **accuracy**, loss, precision, recall and F1-score. Accuracy provides a general idea of how accurate the model is. It works by taking the number of correct predictions and dividing it by the total number:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \qquad \text{Loss} = \frac{1}{n}\sum(y-\hat{y})^2$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}}$$

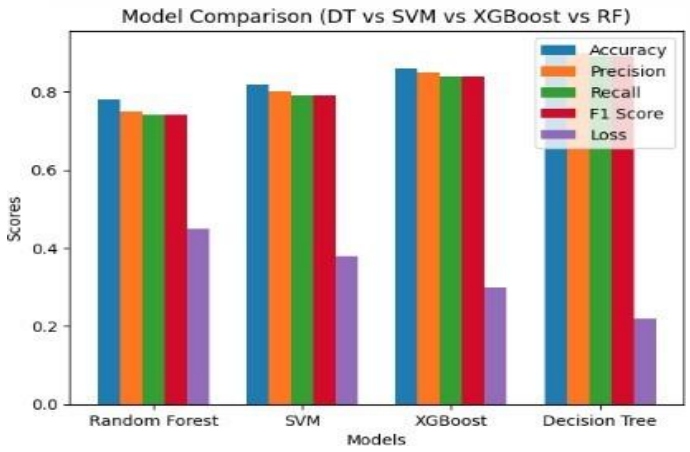
Where TP = True Positives, TN = True Negatives, FP = False Positives and FN = False

$$\text{Precision} = \frac{TP}{TP+FP}$$

Negatives. Loss is the error between the predicted value and actual value. Often used to be minimized during training. Where y is the actual value and y^ is the predicted.

$$\text{Recall} = \frac{TP}{TP+FN}$$

I. Comparison Graph



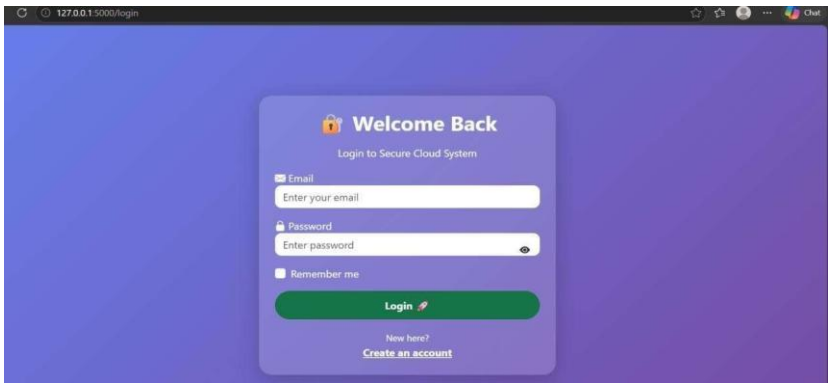
The graph below shows how well each model performed (Decision Tree, Random Forest, SVM & XGBoost). It can be seen that the Decision Tree model performs the best. This is because it has the highest accuracy, precision, recall and F1 score meaning it will make the most accurate predictions. It also has the lowest loss which means there is very little error in its predictions. XGBoost performed the second best. It performed well in all areas and had a low loss. SVM performed reasonably well, it had even results but did not perform as well as XGBoost. Random Forest performed well but did not have as high an accuracy and had a high loss. Overall, this shows how important it is to choose the correct machine learning algorithm to get reliable prediction results. Each model will have different learning behaviours which will affect how well it can detect patterns within the dataset and accurately classify the conditions. The Decision Tree model performed better as it had good decision-making rules and could better differentiate between the different input patterns. As it had good performance across all the factors it provides a good balanced classification and has very little prediction errors. Comparing all the accuracy, precision, recall and F1 scores give an overall understanding of how the models have performed in classifying the data. The loss factor gives additional information about how much error there was in the model's predictions. Whilst Random Forest, SVM & XGBoost all provided some useful classification results, they had lower performance meaning they may not be as suitable as the Decision Tree model. This shows how important it is to choose a model based on how well it performed across all the factors. This means that the Decision Tree model should be chosen
as the best algorithm for our system. However, these results are dependent on the dataset used, pre-processing techniques and evaluation factors. To see how well the model generalises we should test it using different datasets.

V. Outputs:

This section shows the result screens of our implemented web application based on TLADE+. On this Windows system, the application was run using Flask, SQLite database and IPFS locally. It shows the result of a real user's flow through the application, including secure login, uploading files, dynamically encrypting files, storing them on IPFS and viewing their history and being able to decrypt them.

Fig 5.1 Login page with OTP-based authentication

The registered user's login using their email and password. Once the password is validated, they are sent an OTP to their email which they have to enter to view their dashboard. This is explained in the User Authentication and File Management Module.



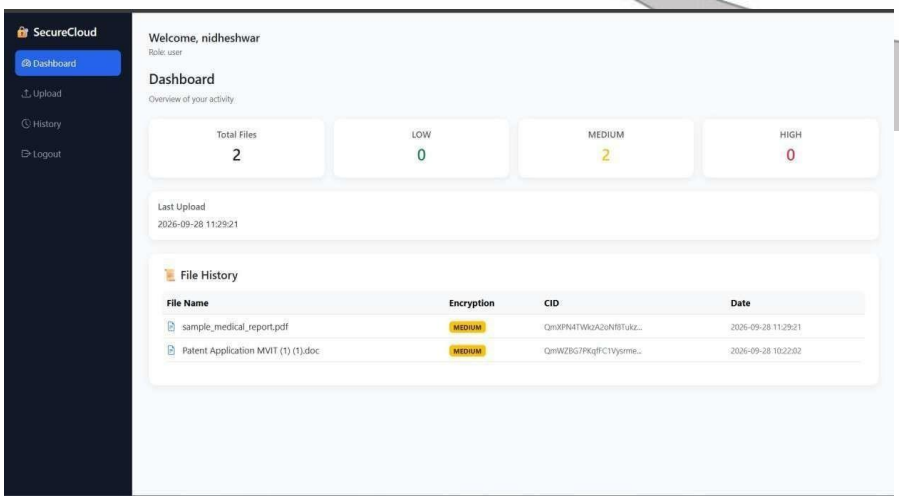


Fig 5.2 User dashboard

This dashboard shows you an overview of your activity including total files stored, number of files with LOW/MEDIUM/HIGH encryption, last date of upload and a brief history of your files. In this case there are 2 files and they have both been given a MEDIUM rating.

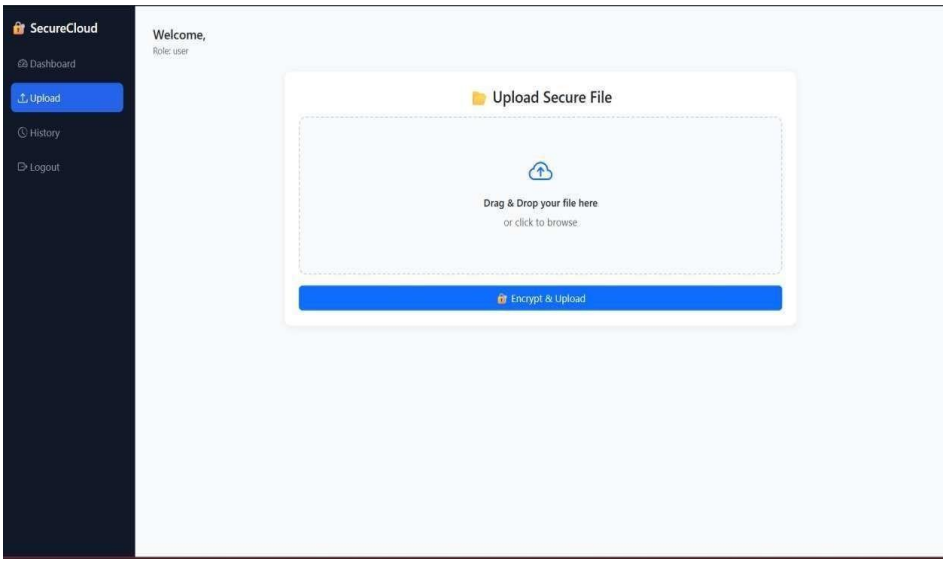


Fig 5.3 Secure file upload page

The user only selects the file through the drag-and-drop area and presses Encrypt & Upload. The choice of the encryption level is taken by the system and not by the user.

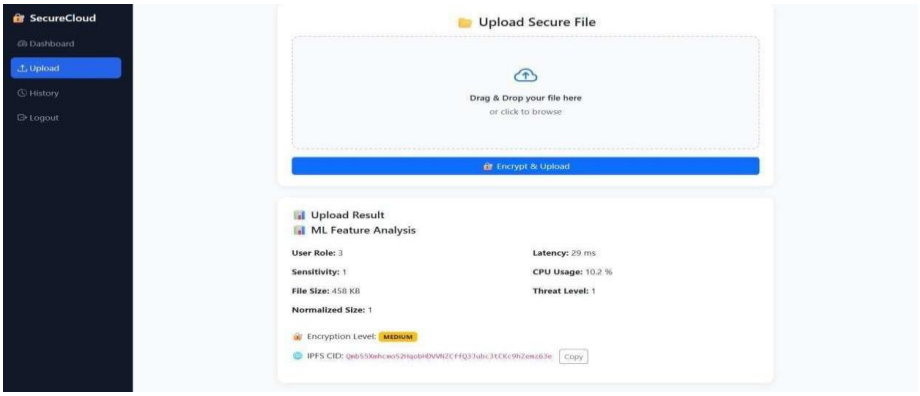
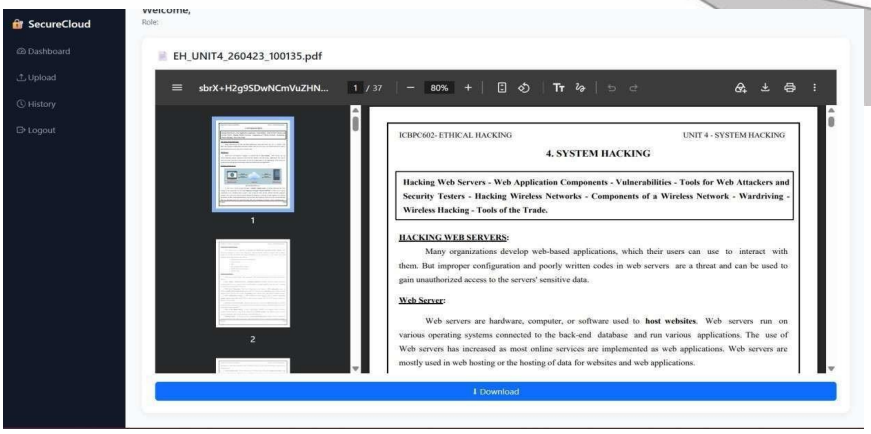


Fig 5.4 Upload result and ML feature analysis



After uploading your file, a page will appear showing what features have been assigned. You can see that Role has been assigned to user 3, Sensitivity = 1, File Size = 458KB (Normalize = 1), Latency = 29ms, CPU Usage = 10.2% and Threat Level = 1. As this is a low risk, low load scenario we decided to implement MEDIUM Encryption and therefore applied AES-256-GCM. We can also see the IPFS CID of the stored file.

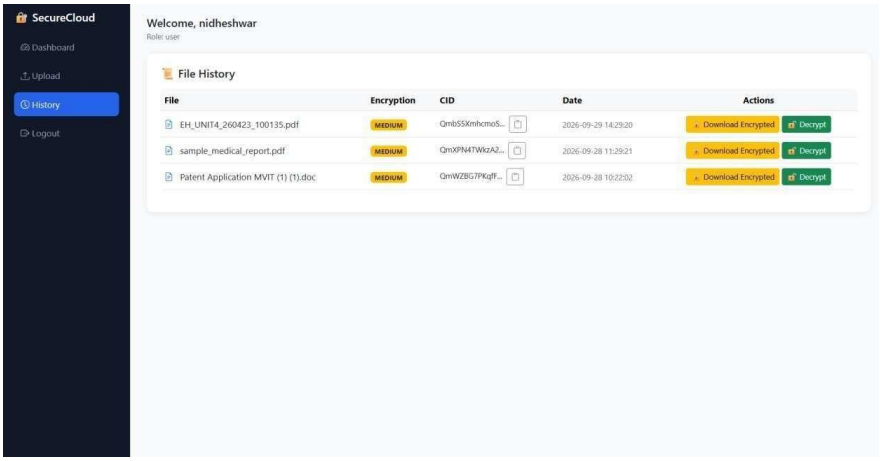


Fig 5.5 File history page

The history page lists each file with its encryption level, IPFS CID, upload time and the Download Encrypted and Decrypt action buttons. The newly uploaded file EH_UNIT4_260423_100135.pdf appears at the top, followed by the earlier uploads.

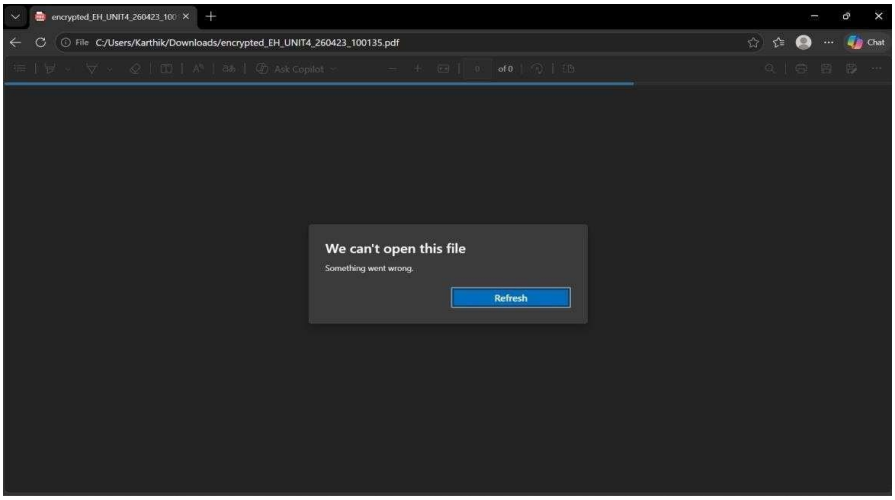


Fig 5.6 Encrypted file opened without decryption

When the file obtained through Download Encrypted is opened in a PDF viewer, it fails with the message “We can’t open this file”, because the stored bytes are ciphertext and not a valid PDF. Anyone who obtains the stored file or its CID without the key therefore gets nothing readable.

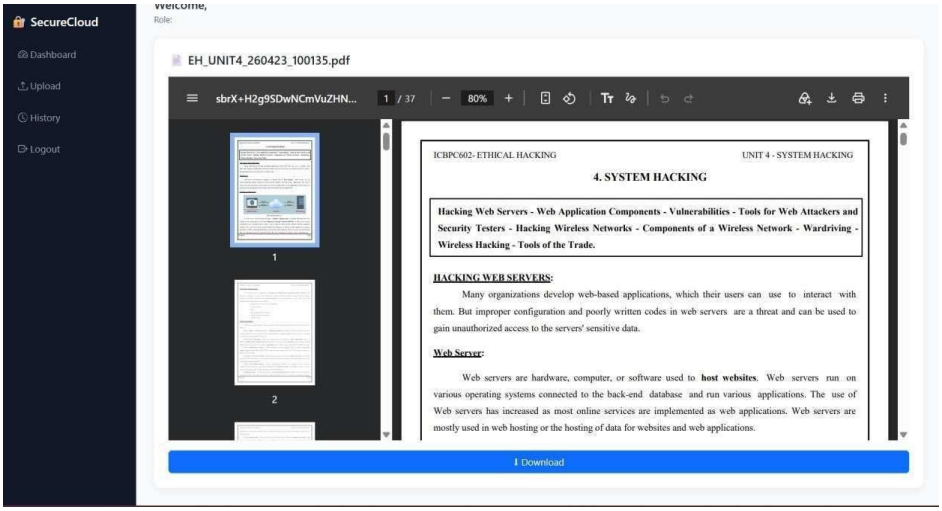


Fig 5.7 Decrypted file viewed inside the application

Pressing Decrypt retrieves the encrypted data using its CID, decrypts it with the owner key and displays the original PDF inside the page. All 37 pages of the file were returned, which confirms that the file is recovered completely after encryption, storage and decryption.

VI. Conclusion

This paper proposes a smart adaptive encryption system to protect files in the cloud. Using Q-learning and Decision Tree, an intelligent system can be formed. An adaptable encryption level
is used depending on factors such as file sensitivity, file size, system latency, CPU usage and threat level. The Q-learning agent will improve decisions made by using reward. Decision Tree can be continuously learnt from by using new data collected from the system. By enforcing a security rule, files with a high sensitivity and high threat level will never receive a LOW encryption. The system has also been developed to include Flask authentication, OTP, cryptography, IPFS, history and access control to provide a completely secure method to manage files. This system can be further developed using larger more diverse data, more advanced reinforcement learning algorithms and real time threat alerts. The system can also be extended using deep Q-learning, automated key management, enhanced security and testing within the cloud.

Reference

[1]S. Ahmad, S. Mehfuz, F. Mebarek-Oudina, et al., RSM analysis-based cloud access security broker: a systematic literature review, Clust. Comput. (2022), doi: 10.1007/s10586-022-03598-z .

[2]Shahnawaz Ahmad, Shabana Mehfuz, Javed Beg, Assessment on potential se- curity threats and introducing novel data security model in a cloud environ- ment, Mater. Today: Proc. 62 (2022) Part 7Pages 4909-4915, ISSN 2214-7853, doi: 10.1016/j.matpr.2022.03.536 .

[3]Shahnawaz Ahmad, Shabana Mehfuz, Javed Beg, Cloud security frame-work and key management services collectively for implementing DLP and IRM, Mater. Today: Proc. 62 (Part 7) (2022) 4828–4836 ISSN 2214-7853, doi: 10.1016/j.matpr.2022.03.420 .

[4]S. Ahmad, S. Mehfuz, J Beg, Fuzzy TOPSIS-based cloud model to evaluate cloud computing services, in: G. Manik, S. Kalia, S.K. Sahoo, T.K. Sharma, O.P. Verma (Eds.), Advances in Mechanical Engineering. Lecture Notes in Mechanical Engineer- ing, Springer, Singapore, 2021, doi: 10.1007/978-981-16-0942-8_4 .

[5]S. Ahmad, S. Mehfuz, J. Beg, Enhancing security of cloud platform with cloud access security broker, in: M.S. Kaiser, J. Xie, V.S. Rathore (Eds.), Informa- tion and Communication Technology For Competitive Strategies (ICTCS 2020). Lecture Notes in Networks and Systems, Vol 190, Springer, Singapore, 2021, doi: 10.1007/978-981-16-0882-7_27 .

[6]Y.M. Mahaboob John, G. Ravi, Real-time regional mobility energy feature approximation-based secure routing for improved quality of service in MANET, Int. J. Commun. Syst. (2021), doi: 10.1002/dac.4713

[7]S. Zhou, L. Wu, C. Jin, A privacy-based SLA violation detection model for the security of cloud computing, China Commun. 14 (9) (Sept. 2017) 155–165, doi: 10.1109/CC.2017.8068773 . [8] F. Hu, B. Chen, Channel coding scheme for relay edge computing wire- less networks via homomorphic encryption and NOMA, in: IEEE Transactions on Cognitive Communications and Networking, 6, Dec. 2020, pp. 1180–1192, doi: 10.1109/TCCN.2020.3023724 .

[9] M.G. Aruna, K.G. Mohan, Secured cloud data migration technique by competent probabilistic public key encryption, China Commun. 17 (5) (May 2020) 168–190, doi: 10.23919/JCC.2020.05.014 .

[10] K. Lee, Comments on "Secure data sharing in cloud computing using revocable- storage identity-based encryption ", in: IEEE Transactions on Cloud Computing, 8, 1 Oct.-Dec. 2020, pp. 1299–1300, doi: 10.1109/TCC.2020.2973623 .

[11] K. Gai, M. Qiu, H. Zhao, Privacy-preserving data encryption trategy for big data in mobile cloud computing, in: IEEE Transactions on Big Data, 7, 1 Oct. 2021, pp. 678 688, doi: 10.1109/TBDATA.2017.2705807 .

[12] L. Cao, Y. Kang, Q. Wu, R. Wu, X. Guo, T. Feng, Searchable encryption cloud storage with dynamic data update to support efficient policy hiding, China Commun. 17 (6) (June 2020) 153–163, doi: 10.23919/JCC.2020.06.013 .

[13] A. Awad, A. Matthews, Y. Qiao, B. Lee, Chaotic searchable encryption for mobile cloud storage, in: IEEE Transactions on Cloud Computing, 6, 1 April-June 2018, pp. 440–452, doi: 10.1109/TCC.2015.2511747 .

[14] Q. Zhang, S. Wang, D. Zhang, J. Wang, Y. Zhang, Time and attribute based dual access control and data integrity verifiable scheme in cloud computing applications, IEEE Access 7 (2019) 137594–137607, doi: 0.1109/ACCESS.2019.2942649 .

[15] L. Jiang, D. Guo, Dynamic encrypted data sharing scheme based on conditional proxy broadcast re-encryption for cloud storage, IEEE Access 5 (2017) 13336–13345, doi: 10.1109/ACCESS.2017.2726584 .

[16] S. B. Nair Premakumari, G. Sundaram, M. Rivera, P. Wheeler, and R. E. Pérez Guzmán, "Reinforcement Q-Learning-Based Adaptive Encryption Model for Cyberthreat Mitigation in Wireless Sensor Networks," Sensors, vol. 25, no. 7, 2025, Art. no. 2056, doi: 10.3390/s25072056. PubMed

[17] P. R. Kumar and S. Goel, "A secure and efficient encryption system based on adaptive and machine learning for securing data in fog computing," Scientific Reports, vol. 15, 2025, Art. no. 11654. Nature

[18] L. Zhang, Y. Yang, K. Pang, T. Chen, and X. Yue, "An adaptive encryption scheme for secure microservice communication via variable- length keys and obfuscation," Journal of Cloud Computing, vol. 15,2026, Art. no. 80, doi: 10.1186/s13677-026-00899-1. Springer

[19] A. Alsirhani, S. Ali, and M. Humayun, "CAEM-ESAC: Context-aware encryption model for enhancing security and access control through contextual factors," Cluster Computing, vol. 29, 2026, Art. no. 449, doi: 10.1007/s10586-026-06241-3. Springer

[20] R. R. Dewangan, S. Soni, D. Thombre, M. Arya, et al., "Privacy preservation and security for the execution of cloud-based optimized robot path using reinforcement learning," Discover Computing, vol. 29, 2026, Art. no. 381, doi: 10.1007/s10791-026-10317-4. Springer

[21] C.-H. Nguyen, T. H. Dinh, D. N. Nguyen, K. Lauter, and M. Kim, "Empowering artificial intelligence with homomorphic encryption for secure deep reinforcement learning," Nature Machine Intelligence, vol. 7, pp. 1913–1926, 2025. Nature

[22] A. Kairi, "Adaptive DNA Cryptography With Intelligent Machine Learning for Cloud Data Defense," Engineering Reports, 2025, doi: 10.1002/eng2.70223. Wiley Online Library