ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

A Secure and Efficient Cloud Storage Framework for Confidential Financial Data Management

[1]Charles Ubagaram, [2]Narsing Rao Dyavani, [3]Bhagath Singh Jayaprakasam, [4]Rohith Reddy Mandala, [5]Venkat Garikipati, [6]Zerihun Olana Asefa
[1]Tata Consultancy Services, Ohio, USA
charlesubagaram17@gmail.com
[2]Uber Technologies Inc, California, USA
nrd3010@gmail.com
[3]Cognizant Technology Solutions, Texas, USA
Bhagath.mtech903@gmail.com
[4]Tekzone Systems Inc, Rancho Cordova, California, USA
rohithreddymandala4@gmail.com
[5]Innosoft, Maryland, USA
venkat44557@gmail.com
[6] Department of Information Technology, Jimma University, Ethiopia
zerihun.olana@ju.edu.et

Article Info

Received: 31-01-2025 Revised: 18-03-2025 Accepted: 30-03-2025

Abstract

One of the features cloud computing offers to the financial sector is to provide it with scalability, and flexibility to handle enormous amounts of very sensitive market data. Nevertheless, migration of financial data into the cloud brings along security threats of being accessed by unauthorized persons and compliance issues. Hence, this framework is developed to integrate the secure storage of cloud infrastructure with encryption, indexing, and reliable storage. Daily stock index price data from the global exchange is collected and encrypted immediately at the client end using the Advanced Encryption Standard (AES-256), also the encryption keys are managed securely through Amazon Web Services Key Management Service (AWS KMS). A blind index is generated with Secure Hashing Algorithm (SHA-256) as a hash function with a secret salt, allowing efficient private keyword searches without disclosing the actual data. Encrypted data and hashed indices are then stored in AWS Simple Storage Service (AWS S3) with server-side encryption. Performance tests showed that encrypting 1200 MB of data took only 4.8 seconds, and indexing 10,000 keywords required just 5700 milliseconds, confirming practical efficiency. Overall, this work delivers a scalable, end-to-end secure cloud storage solution that improves privacy and trust in managing sensitive financial data in the cloud.

Keywords:

AWS, Key Management Service, AES-256, SHA-256, Cloud, Finance, Data Management.

1. INTRODUCTION

Cloud computing opens up the opportunity for the financial industry to manage and store huge amounts of classified data flexibly and scalably. But this also poses large security issues that demand a slew of encryption measures, treating of data with care, and compliance that ensure both the protection and trustworthiness of the data. In the modern-day big polycentric enterprises, cloud computing has turned capital in supporting the financial sector in processing massive chunks of data. Financial institutions are now relying more and more on cloud-based infrastructure for the handling and secure treatment of sensitive information such as transactions, customer records, and compliance logs [1]. However, we have seen that massive data processing and storing presents a nightmare for data confidentiality, data integrity, and unauthorized data access issues. The studies also disclosed that

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

traditional data management systems as well as older relational databases were unable to cope with Big Data workloads and their performance requirements [2] [3]. As cloud adoption grows, so does the necessity for enhanced security measures to safeguard outsourced data from evolving threats [4]. Encryption, secure indexing, and reliable data governance are essential to make sure trust isn't broken and regulations are met in the financial sector. To tackle these risks, new frameworks must guarantee end-to-end protection from the moment data is collected to its storage and retrieval. This work proposes such a framework, designed to ensure robust security for sensitive financial data in cloud environments.

Different approaches have been developed to protect data in cloud environments [5], including Attribute-Based Encryption (ABE), Proxy Re-Encryption (PRE), Hierarchical encryption, and Elliptic Curve Cryptography (ECC)-based approaches. [6] [7] Some studies advocate using searchable encryption and trust models to handle access control and privacy. [8] [9] [10] [11] While the operations of these methods strengthen cloud security on certain planes, many still suffer from issues such as high computational overhead, difficult key management, limited sampling, or simply being unfriendly for real-time or search. For instance, ABE can become inefficient for large-scale dynamic datasets due to its complex access structures and revocation challenges. In a parallel manner, distributed storage based on ECC shall grant solid data security for encryption, though it shall be really difficult to allow for flexible keyword search on compatible cloud storage services [12] [13] [14]. Despite the great strides made, gaps remain in establishing a fine balance between strong encryption on one side and lightweight, and practicable search and retrieval on the other, especially as far as large, sensitive financial datasets are concerned.

To handle these drawbacks, a lightweight client-side AES-256 encryption [15] is employed in conjunction with hashed keyword indexing [16] [17] [18] and secure cloud storage for providing a minimal yet robust solution toward protecting financial data. So different from the methods relying on complicated multi-layered cryptographic models with enormous computational requirements, this method sets its focus upon using a practical pipeline from storage systems integrating well with the existing cloud service. Using a simple hashed index through SHA-256 [19], the framework supports fast and private retrieval of data that prevents any exposure of raw keywords to the cloud [20]. The keys used for encryption are managed by a trusted cloud-based KMS allowing for secure, automatic rotation of these keys. This combination, never seen before in the literature, keeps data confidential during transmission, storage, and accessibility, and yet it remains lightweight and easy to implement. Hence, this framework directly tackles the efficiency and usability issues found in current solutions, providing a scalable method that is secure and efficient for financial cloud applications of today. The key contributions of this proposed work are as follows;

- Design a secure, confidential, and efficient framework for storing and retrieving sensitive financial market data in the cloud.
- Collect daily stock index price records from major global exchanges using validated APIs and secure transmission protocols.
- Implement strong client-side encryption using AES-256 with secure key management through AWS KMS to keep data safe both when stored and while it's being sent.
- Develop a blind indexing mechanism using SHA-256 with a secret salt to enable privacy-preserving and efficient keyword search within encrypted cloud storage.

The organization of this paper is as follows: Section 2 reviews related studies. Section 3 describes the proposed framework. Section 4 presents and discusses the performance evaluation results. Finally, Section 5 concludes the paper.

2. LITERATURE SURVEY

There have been many previous works on the development of secure frameworks for secure management and protection of private data. Ogiela et al. [21] and Yan et al. [22] provided sophisticated secure sharing procedures and cryptographic secret-splitting methods to share secret data among trusted parties via intelligent linguistic methods. Albeit such methods provide high confidentiality, they suffered from unbearable computational and administrative overheads, rendering the implementation in real-time large-scale financial data processing

ISSN:0976-0172
Journal of Bioscience And Technology
www.jbstonline.com

impractical. To resolve this, the proposed system uses AES-256 client-side encryption and blind indexing with SHA-256 instead of advanced multi-level sharing schemes. This keeps the security paradigm mostly intact while minimizing complexity in the operation.

To strengthen cloud storage security, Cho et al. [23] and Mandal et al. [24] proposed distinct zero-trust models to secure data in Amazon S3, illustrating access control lists and key management for enhancement of security by forensic validation. However, their suggestions lack strong client-side encryption or privacy-preserving search required in financial data workflow security. Meanwhile, for multi-cloud secure multimedia analysis, Li et al. [25], Kanwal et al. [26], and Al-Wahah [27] have considered Semantic-Based Access Control (SBAC) and models like OBAR and SIM. while Li et al [28], Thabit et al. [29], and Adee and Mouratidis [30] introduced SA-EDS to split and distribute files securely. Although these methods strengthened data protection, they added computational load that makes financial data tasks harder. The proposed system simplifies this by combining AES-256 encryption and SHA-256 blind indexing with AWS KMS, balancing strong security with practical performance.

Chang and Ramachandran [31], and Zbořil and Svatá [32] have in turn proposed and extended Chang's [33] CCAF with additional multiple layered security models and BPMN simulations for assessing the behavior and security performance of such a system. CCAF is suitable and hence considered for large-scale data centers but showed breach response times ranging from 50 to 125 hours, which is unacceptable for financial institutions needing to protect and recover almost instantly. Also, based more on policy controls, CCAF does not deal with granular controls such as encryption right away or privacy-preserving indexing for fast, secure access. Rath et al. [34] and Hawedi et al. [35] offered reusable security patterns and best practices for Cloud SaaS, but they mostly grasped general aspects rather than concrete encryptions and indexing for live financial datasets. As a response to fill these voids, the framework embraces AES-256 encryption combined with blind indexing using SHA-256 with a secret salt and key storage in AWS KMS to provide complete protection for sensitive financial market data in the cloud.

3. PROPOSED METHODOLOGY

The proposed system aims to build a framework to ensure secure, confidential, efficient cloud storage and retrieval of critical financial market information. Daily stock index price data is collected. Immediately, the data is encrypted on the client side with AES-256, a widely trusted symmetric encryption standard, with keys managed by a secure cloud-based KMS to simplify safe key handling. The keywords are first turned into hashes with SHA-256 together with a secret salt so that they can be searched efficiently without revealing the raw data. Finally, the encrypted items and their indexes are uploaded to secure cloud storage containing server-side encryption for end-to-end protection. Thus, Figure 1 displays an overview of the workflow in the proposed system.

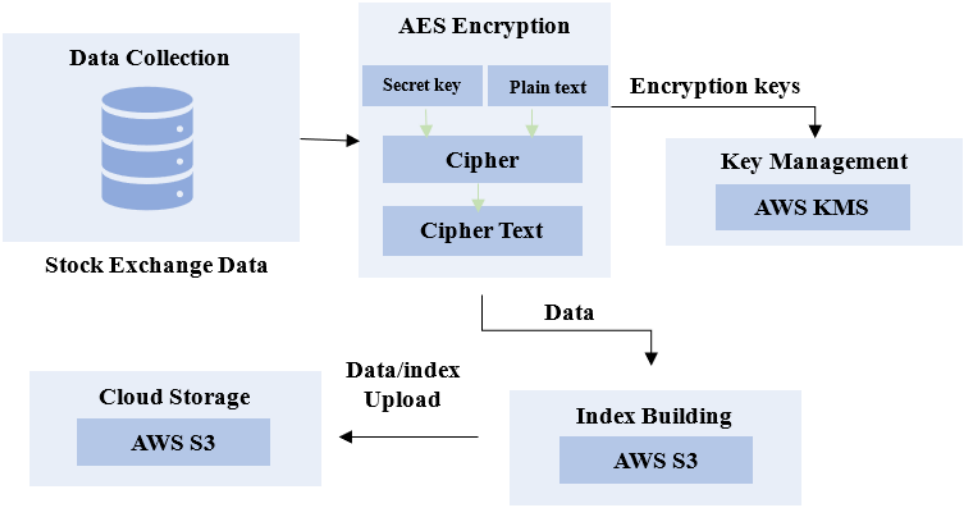

Figure 1: Entire architecture of the proposed methodology.

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

3.1. Data Collection

In this work, the financial data obtained contains daily stock prices of market indexes from major global exchanges such as the United States, China, Canada, Germany, Japan, amongst others. The broader market data was extracted from Yahoo Finance, which has been able to provide several decades of solid price information for most exchanges. Each record was checked to ensure that it was both accurate and complete before entering the system. By securing a reliable, long-term dataset through a secure channel, this work guarantees that sensitive finance information enters the pipeline in a trusted, authenticated manner. Once acquired, the raw data proceeds directly to the encryption step, hence adhering to our secure data management workflow, so the data is guaranteed end-to-end protection through its lifecycle in storage and processing in the cloud.

Dataset Link: Stock Exchange Dataset

3.2. Data Encryption

Data encryption refers to the conversion of ordinary readable information into an incoherent form using cryptographic algorithms, which only a specified user with an actual decryption key may access. Thus, in cloud-based financial data management, encryption serves in preventing the unauthorized view of information, data breaches, and insider threats. When strong encryption algorithms such as AES-256 are used, the data is treated to a double layer of protection, i.e., protection when it is transmitted and protection when it is stored. Proper encryption allows the secret to remain secret; therefore, even if the data gets intercepted or exposed somewhere in the cloud, it will never be intelligible to an attacker, thus maintaining the confidentiality and integrity of those critical financial records.

- ***AES-256***

AES, the standard encryption algorithm, is a block-based symmetric encryption method assumed to handle sensitive data. Being the strongest AES uses a 256-bit encryption key for maximum security. AES breaks data into 128-bit blocks for encryption. During the encryption of one block, AES-256 carries out 14 rounds of complicated transformations on the plaintext to transform it into the ciphertext. For encrypting this 128-bit block of plaintext, AES-256 performs multiple rounds that implement key-dependent operations. The initial phase of encryption is key expansion, where the 256-bit cipher key (K) is taken and Nr + 1 round keys (W[0] to W[59]) are created by the key expansion algorithm as shown in equations (1) and (2):

$$W[i] = W[i - Nk] \oplus \text{SubWord}(\text{RotWord}(W[i - 1])) \oplus \text{Rcon}[i/Nk], \text{for } i \equiv 0(\text{mod}Nk) \quad (1)$$

and

$$W[i] = W[i - Nk] \oplus \text{SubWord}(W[i - 1]), \text{if } Nk > 6 \text{ and } i \equiv 4(\text{mod}Nk). \quad (2)$$

Here, Nk = 8 for AES-256 (the no. of 32-bit words in the key). The SubWord function substitutes every byte in a word with a value from the S-box , RotWord shifts the positions of the bytes in a word in a loop, and Rcon[i] is a round constant that varies per round.

The plaintext block (P) of 128 -bits is then mapped to the state matrix S . The primary round applies only the AddRoundKey operation shown in equation (3):

$$S = P \oplus W[0..3]. \quad (3)$$

For each of the 14 rounds, the state S is transformed through four main operations:

i. SubBytes: Each byte s_{ij} in S is replaced using the S-box as represented in equation (4):

$$s'_{ij} = \text{S} - \text{box}(s_{ij}). \quad (4)$$

ii. ShiftRows: Each row of the state is rotated cyclically by different offsets as displayed in equation (5):

$$\begin{cases} \text{Row }_0\text{: no shift} \\ \text{Row }_1\text{: } s_{1,j} = s_{1,(j+1)\text{mod}4} \\ \text{Row }_2\text{: } s_{2,j} = s_{2,(j+2)\text{mod}4} \\ \text{Row }_3\text{: } s_{3,j} = s_{3,(j+3)\text{mod}4}. \end{cases} \quad (5)$$

iii. MixColumns: Each column of the state is converted by matrix multiplication in the finite field GF(2^{ε}) as shown in equation (6):

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

$$\begin{bmatrix} s_0' \\ s_1' \\ s_2' \\ s_3' \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad (6)$$

This step mixes bytes within each column to diffuse the input bits.

iv. AddRoundKey: The current round key is added (XOR'ed) to the state as in equation (7):

$$S = S \oplus W[r \times Nb : (r+1) \times Nb], \text{ for round } r. \quad (7)$$

Here, Nb = 4 (no. of 32-bit words in the state). In the final round, the MixColumns step is omitted to preserve the correct structure for decryption. During decryption, each operation is inverted: InvSubBytes uses the inverse S-box, InvShiftRows applies opposite shifts, InvMixColumns uses the inverse mixing matrix, and the round keys are XOR'ed in reverse order.

This combination of formulas expresses the manner in which AES-256 iterates on the plaintext, implementing substitution, permutation, mixing, and key addition, achieving extreme resistance to cryptographic attacks and brute-force decryption attempts.

- ***AWS KMS***

AWS KMS is a cloud service that allows one to create, store, and manage cryptographic keys securely. It automatically generates, rotates, and manages access to the keys so that users may easily encrypt or decrypt data across AWS services. After encryption, keys are all specially stored and protected from unauthorized access and misuse within KMS.

3.3. Blind Indexing

Blind indexing refers to an activity by which secure, privacy-preserving references are created to allow stored data to be searched for and retrieved fast and efficiently without exposing original keywords. Just by contrast, in the case of simple hash indexing, an attacker would simply reverse-engineer the index by using a dictionary or brute force attack if there was no secret salt added. In this work, blind indexing uses SHA-256 with added randomness in hashing keywords or attributes so that only these masked values are saved and nothing of the original data is seen. This method enables fast lookups and also guarantees confidentiality of information stored in secure clouds.

- ***SHA-256***

SHA-256 is a cryptography-based hash function that converts any input data into a fixed 256-bit (32-byte) hash value. It does so by first preprocessing the input by padding, splitting it into 512-bit blocks, and then processing each block with logical functions, bitwise shifts, and modular additions. SHA-256 combines data very cleverly using constant values with a compression function so that a slight alteration in the input will produce a completely different output.

Each message block is first extended with extra bits to make its total length a multiple of 512 bits. This padding starts with a single '1' bit, followed by enough '0' bits, and ends with the original message length encoded as a 64-bit value. For every padded block, the algorithm sets up eight 32-bit working variables $a, b, c, d, e, f, g, h.$ initialized using fixed hash values (H_0 to H_7) derived from the fractional parts of the square roots of the first eight prime numbers. The variables x, y, and z are drawn from these working variables and hold the intermediate results while the algorithm processes and compresses each 512-bit chunk step by step.

Each 512-bit block is split into 16 baseline 32-bit words, W[0] to W[15]. To expand these into 64 words, W[0] to W[63], the message schedule uses the recurrence mentioned in equation (8):

$$W[t] = \sigma_1(W[t-2]) + W[t-7] + \sigma_0(W[t-15]) + W[t-16] \text{ for } t = 16, \ldots, 63 \quad (8)$$

where,

$$\sigma_0(x) = \text{ROTR}^7(x) \oplus \text{ROTR}^{18}(x) \oplus \text{SHR}^3(x) \quad (9)$$

$$\sigma_1(x) = \text{ROTR}^{17}(x) \oplus \text{ROTR}^{19}(x) \oplus \text{SHR}^{10}(x) \quad (10)$$

Here, $\text{ROTR}^n(x)$ denotes a rotation of x by n bits to the right, $\text{SHR}^n(x)$ is a right logical shift by n bits (with zero-fill), and $\oplus$ represents bitwise XOR. This schedule ensures that each new word depends on a mix of earlier words, spreading entropy throughout the input.

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

The core compression function then processes each word W[t] through 64 rounds. For each round, two short term variables T_1 and T_2 are computed as shown in equations (12) and (13):

$$T_1 = h + \Sigma_1(e) + \mathrm{Ch}(e,f,g) + K[t] + W[t] \quad (11)$$

$$T_2 = \Sigma_0(a) + \mathrm{Maj}(a,b,c) \quad (12)$$

The functions are defined as:

$$\Sigma_0(x) = \mathrm{ROTR}^2(x) \oplus \mathrm{ROTR}^{13}(x) \oplus \mathrm{ROTR}^{22}(x) \quad (13)$$

$$\Sigma_1(x) = \mathrm{ROTR}^6(x) \oplus \mathrm{ROTR}^{11}(x) \oplus \mathrm{ROTR}^{25}(x) \quad (14)$$

$$\mathrm{Ch}(x,y,z) = (x \wedge y) \oplus (-x \wedge z) \quad (15)$$

$$\mathrm{Maj}(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (16)$$

Here, $\wedge$ implies bitwise AND and $\sim x$ is bitwise NOT. The round constant K[t] is derived from the decimal parts of the cube roots of the first 64 prime numbers. When combined with a secret salt, SHA-256 provides additional protection by making it nearly impossible for attackers to guess the actual input through precomputed attacks.

3.4. Cloud Storage

The cloud storage method secures and manages private data against unauthorized threats through the combination of strong encryption techniques, safe key handling, and control over access rights. AWS S3 is one of the popular cloud storage solutions that offers highly scalable, durable, and secure object storage within an unmatched price class. It is a truly flexible storage option with which most organizations perform backups, archiving, big data analytics, and even cloud-native applications. In this work, AWS S3 is used as the secure storage layer for encrypted financial data and its associated blind index. Due to data storage on S3, the system gets the options for automatic server-side encryption, high availability, and multi-region replication to maintain increased resilience to data loss. Integration with the S3 from AWS is perfect with familiar AWS services, such as KMS for encryption key management, hence improving the overall security posture. Its fine-grained access controls and versioning capabilities also provide full control over stored files should any situation of accidental deletion or unauthorized change arise. Thus, S3 from AWS proves itself an excellent solution for managing large quantities of critical financial information safely in the cloud.

4. RESULTS AND DISCUSSIONS

Python was used to build the implementation and test the encrypt time and the blind index creation time for performance evaluation of a proposed secure cloud storage system. Experiments showed that this system has remained in good security while providing for adequate processing time for all sizes of data and keyword volumes. The observed outcomes demonstrated that the system is both robust and efficient when the requirement is to secure sensitive financial data in the cloud.

4.1. Performance Analysis

The encryption time of the secure cloud storage framework was then tested to make sure that AES-256 encryption does not cause a considerable processing delay while encrypting sensitive financial data. Further, the blind index generation time based on SHA-256 was measured to make sure that secure privacy-preserving search can be supported without adding severe overhead.

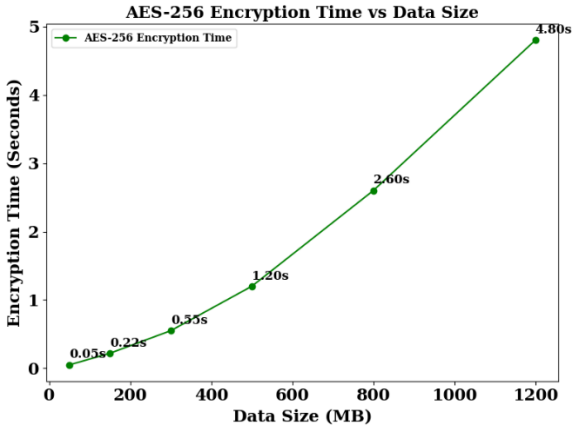

Figure 2: Evaluation of AES-256 Encryption Time for Varying Data Sizes

Figure 2 illustrates the AES-256 encryption time graph for various data sizes used for determining the encryption speed in the proposed cloud-based system. As the graph depicts, it takes roughly 0.05 seconds to encrypt a dataset of 10 MB, 0.22 seconds to encrypt a dataset of 200 MB, and about 0.53 seconds to encrypt 400 MB of data. The larger datasets take 1.20 seconds for 600 MB, 2.60 seconds for 800 MB, and finally, 4.8 seconds to encrypt 1200 MB. This data suggested that with a large

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

increment in data volume, the AES-256 encryption remains practically applicable from a computational perspective with only slight increments in processing time. Therefore, this study confirms that secure encryption could be performed efficiently upon large financial datasets to avoid becoming a bottleneck while maintaining confidentiality and compliance and thus support real-time LSTM forecasting workflows in the cloud.

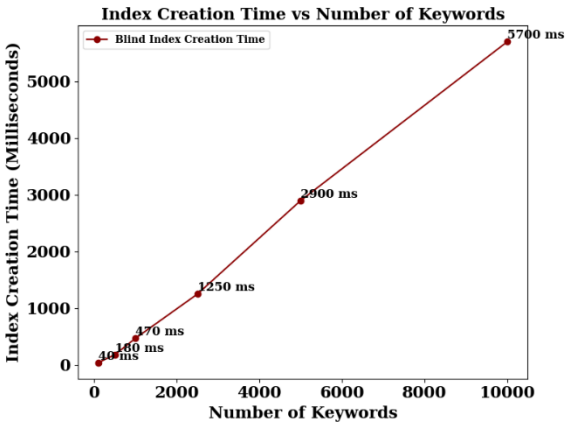

Figure 3: Blind Index Creation Time vs. Keyword Volume

Figure 3 represents the evaluation of blind index creation time using SHA-256 with a secret salt to check how well the processing time scales with the number of keywords. So the performance metric we are interested in is the time taken to create an index with various volumes of keywords, measured in milliseconds. The results show indexing 100 keywords requires nearly 40 ms; 500 keywords, 180 ms; 1,000 keywords, nearly 470 ms; 2,500 keywords, nearly 1,250; 5,000 keywords, 2,900 ms, and finally, 10,000 keywords require around 5,700 ms. Thus, the tests confirm that the blind index procedure comes with minimal added overhead, thereby remaining practical and ready to work in bigger datasets. The evaluation further corroborates the fact that the system provides support for secure and privacy-preserving keyword indexing that facilitates swift and safe search in the encrypted financial data reliably stored in the cloud.

5. CONCLUSION

In this work, a secure cloud-based framework was designed to store and manage sensitive financial market data with strong confidentiality and practical efficiency. Daily stock index price data from major global exchanges was collected. Client-side AES-256 encryption combined with secure key management through AWS KMS is done to ensure robust data protection. A blind index using SHA-256 with a secret salt was created to enable privacy protection, efficient keyword search without exposing raw information. Encryption performance indicates that data sizes ranging from 10 MB to 1,200 MB were encrypted within a timeframe of 0.05 to 4.8 seconds, with blind index creation scaling from 0.05 milliseconds up to 5700 milliseconds for 10,000 keywords. This attests to the system's realistic feasibility.. Together, these results confirm that the framework can secure and handle massive amounts of actual financial data for storing and retrieving in the cloud without a loss of performance. Future work will concentrate on incorporating advanced searchable encryption, secure enclaves to increase privacy, flexible querying, and responsiveness for today's financial applications.

6. REFERENCES

[1] S. Sakr, A. Liu, D. M. Batista, and M. Alomari, “A Survey of Large Scale Data Management Approaches in Cloud Environments,” *IEEE Commun. Surv. Tutor.*, vol. 13, no. 3, pp. 311–336, 2011, doi: 10.1109/SURV.2011.032211.00087.

[2] K. Ren, C. Wang, and Q. Wang, “Security Challenges for the Public Cloud,” *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan. 2012, doi: 10.1109/MIC.2012.14.

[3] K. Grolinger, W. A. Higashino, A. Tiwari, and M. A. Capretz, “Data management in cloud environments: NoSQL and NewSQL data stores,” *J. Cloud Comput. Adv. Syst. Appl.*, vol. 2, no. 1, p. 22, Dec. 2013, doi: 10.1186/2192-113X-2-22.

[4] P. J. Sun, “Privacy Protection and Data Security in Cloud Computing: A Survey, Challenges, and Solutions,” *IEEE Access*, vol. 7, pp. 147420–147452, 2019, doi: 10.1109/ACCESS.2019.2946185.

[5] D. W. Chadwick *et al.*, “A cloud-edge based data security architecture for sharing and analysing cyber threat information,” *Future Gener. Comput. Syst.*, vol. 102, pp. 710–722, Jan. 2020, doi: 10.1016/j.future.2019.06.026.

[6] M. D. Ryan, “Cloud computing security: The scientific challenge, and a survey of solutions,” *J. Syst. Softw.*, vol. 86, no. 9, pp. 2263–2268, Sep. 2013, doi: 10.1016/j.jss.2012.12.025.

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

[7] J. Tang, Y. Cui, Q. Li, K. Ren, J. Liu, and R. Buyya, "Ensuring Security and Privacy Preservation for Cloud Data Services," *ACM Comput Surv*, vol. 49, no. 1, p. 13:1-13:39, Jun. 2016, doi: 10.1145/2906153.

[8] S. Raghavendra *et al.*, "Index Generation and Secure Multi-user Access Control over an Encrypted Cloud Data," *Procedia Comput. Sci.*, vol. 89, pp. 293–300, Jan. 2016, doi: 10.1016/j.procs.2016.06.062.

[9] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Inf. Sci.*, vol. 305, pp. 357–383, Jun. 2015, doi: 10.1016/j.ins.2015.01.025.

[10] N. Subramanian and A. Jeyaraj, "Recent security challenges in cloud computing," *Comput. Electr. Eng.*, vol. 71, pp. 28–42, Oct. 2018, doi: 10.1016/j.compeleceng.2018.06.006.

[11] H. Tabrizchi and M. Kuchaki Rafsanjani, "A survey on security challenges in cloud computing: issues, threats, and solutions," *J. Supercomput.*, vol. 76, no. 12, pp. 9493–9532, Dec. 2020, doi: 10.1007/s11227-020-03213-1.

[12] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Feb. 2016, doi: 10.1109/TPDS.2015.2401003.

[13] Y. Zhou, N. Li, Y. Tian, D. An, and L. Wang, "Public Key Encryption with Keyword Search in Cloud: A Survey," *Entropy*, vol. 22, no. 4, Art. no. 4, Apr. 2020, doi: 10.3390/e22040421.

[14] A. Meharwade and G. A. Patil, "Efficient Keyword Search over Encrypted Cloud Data," *Procedia Comput. Sci.*, vol. 78, pp. 139–145, Jan. 2016, doi: 10.1016/j.procs.2016.02.023.

[15] L. Khakim, M. Mukhlisin, and A. Suharjono, "Security system design for cloud computing by using the combination of AES256 and MD5 algorithm," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 732, no. 1, p. 012044, Jan. 2020, doi: 10.1088/1757-899X/732/1/012044.

[16] S. Q. Ren *et al.*, "Secure searching on cloud storage enhanced by homomorphic indexing," *Future Gener. Comput. Syst.*, vol. 65, pp. 102–110, Dec. 2016, doi: 10.1016/j.future.2016.03.013.

[17] N. H. Sultan, N. Kaaniche, M. Laurent, and F. A. Barbhuiya, "Authorized Keyword Search over Outsourced Encrypted Data in Cloud Environment," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 216–233, Jan. 2022, doi: 10.1109/TCC.2019.2931896.

[18] N. Shekokar, K. Sampat, C. Chandawalla, and J. Shah, "Implementation of Fuzzy Keyword Search over Encrypted Data in Cloud Computing," *Procedia Comput. Sci.*, vol. 45, pp. 499–505, Jan. 2015, doi: 10.1016/j.procs.2015.03.089.

[19] M. Husni *et al.*, "Security audit in cloud-based server by using encrypted data AES -256 and SHA-256," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 830, no. 3, p. 032015, Apr. 2020, doi: 10.1088/1757-899X/830/3/032015.

[20] Y.-S. Zhao and Q.-A. Zeng, "Secure and Efficient Product Information Retrieval in Cloud Computing," *IEEE Access*, vol. 6, pp. 14747–14754, 2018, doi: 10.1109/ACCESS.2018.2816919.

[21] L. Ogiela, M. R. Ogiela, and H. Ko, "Intelligent Data Management and Security in Cloud Computing," *Sensors*, vol. 20, no. 12, Art. no. 12, Jan. 2020, doi: 10.3390/s20123458.

[22] Z. YAN, R. H. DENG, and V. VARADHARAJAN, "Cryptography and data security in cloud computing," *Inf. Sci.*, vol. 387, pp. 53–55, May 2017, doi: 10.1016/j.ins.2016.12.034.

[23] K.-H. Cho, J.-H. Cho, H.-W. Lee, and J. Kim, "Design and Forensic Analysis of a Zero Trust Model for Amazon S3," *J. Korea Inst. Inf. Secur. Cryptol.*, vol. 33, no. 2, pp. 295–303, 2023, doi: 10.13089/JKIISC.2023.33.2.295.

[24] S. Mandal, D. A. Khan, and S. Jain, "Cloud-Based Zero Trust Access Control Policy: An Approach to Support Work-From-Home Driven by COVID-19 Pandemic," *New Gener. Comput.*, vol. 39, no. 3, pp. 599–622, Nov. 2021, doi: 10.1007/s00354-021-00130-6.

[25] Y. Li, K. Gai, Z. Ming, H. Zhao, and M. Qiu, "Intercrossed Access Controls for Secure Financial Services on Multimedia Big Data in Cloud Systems," *ACM Trans Multimed. Comput Commun Appl*, vol. 12, no. 4s, p. 67:1-67:18, Sep. 2016, doi: 10.1145/2978575.

[26] T. Kanwal *et al.*, "Privacy-aware relationship semantics–based XACML access control model for electronic health records in hybrid cloud," *Int. J. Distrib. Sens. Netw.*, vol. 15, no.

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

6, p. 1550147719846050, Jun. 2019, doi: 10.1177/1550147719846050.

[27] M. A. H. AL-Wahah, "Semantic-Based Access Control Mechanisms in Dynamic Environments," *Theses Diss.*, Apr. 2019, [Online]. Available: https://scholarcommons.sc.edu/etd/5224

[28] Y. Li, K. Gai, L. Qiu, M. Qiu, and H. Zhao, "Intelligent cryptography approach for secure distributed big data storage in cloud computing," *Inf. Sci.*, vol. 387, pp. 103–115, May 2017, doi: 10.1016/j.ins.2016.09.005.

[29] F. Thabit, A. P. S. Alhomdy, A. H. A. Al-Ahdal, and P. D. S. Jagtap, "A new lightweight cryptographic algorithm for enhancing data security in cloud computing," *Glob. Transit. Proc.*, vol. 2, no. 1, pp. 91–99, Jun. 2021, doi: 10.1016/j.gltp.2021.01.013.

[30] R. Adee and H. Mouratidis, "A Dynamic Four-Step Data Security Model for Data in Cloud Computing Based on Cryptography and Steganography," *Sensors*, vol. 22, no. 3, Art. no. 3, Jan. 2022, doi: 10.3390/s22031109.

[31] V. Chang and M. Ramachandran, "Towards Achieving Data Security with the Cloud Computing Adoption Framework," *IEEE Trans. Serv. Comput.*, vol. 9, no. 1, pp. 138–151, Jan. 2016, doi: 10.1109/TSC.2015.2491281.

[32] M. Zbořil and V. Svatá, "Cloud Adoption Framework," *Procedia Comput. Sci.*, vol. 207, pp. 483–493, Jan. 2022, doi: 10.1016/j.procs.2022.09.103.

[33] V. Chang, Y.-H. Kuo, and M. Ramachandran, "Cloud computing adoption framework: A security framework for business clouds," *Future Gener. Comput. Syst.*, vol. 57, pp. 24–41, Apr. 2016, doi: 10.1016/j.future.2015.09.031.

[34] A. Rath, B. Spasic, N. Boucart, and P. Thiran, "Security Pattern for Cloud SaaS: From System and Data Security to Privacy Case Study in AWS and Azure," *Computers*, vol. 8, no. 2, Art. no. 2, Jun. 2019, doi: 10.3390/computers8020034.

[35] M. Hawedi, C. Talhi, and H. Boucheneb, "Security as a Service for Public Cloud Tenants(SaaS)," *Procedia Comput. Sci.*, vol. 130, pp. 1025–1030, Jan. 2018, doi: 10.1016/j.procs.2018.04.143.