Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 29 (2026), No. 10-A, pp. 3727–3738
DOI : 10.47974/JDMSC-3004

Cryptographic indexing schemes for encrypted scientific databases

Shweta S. Aladakatti [§]
School of Computer Engineering
Manipal Institute of Technology Bengaluru
Manipal Academy of Higher Education
Bengaluru 560064
Karnataka
India

Bondada Venkata Ramana [†]
Department of Computer Science
Aditya Degree & PG Colleges
Kakinada 533003
Andhra Pradesh
India

Vijaysinh U. Bansude *
Department of Electronics & Telecommunication Engineering
S B Patil College of Engineering
Indapur
Pune 413106
Maharashtra
India

Shwetambari Pandurang Katake [‡]
Department of Computer Engineering
Vishwakarma Institute of Technology
Pune 411037
Maharashtra
India

[§] *E-mail:* shweta.aladakatti@manipal.edu
[†] *E-mail:* ramanabondada999@gmail.com
* *E-mail:* vijubansude@gmail.com (Corresponding Author)
[‡] *E-mail:* shwetambari.katake@vit.edu

© Taru Publications

M. Keerthana [@]
Department of Computer Science and Engineering (Data Science)
Vardhaman College of Engineering
Shamshabad
Hyderabad 501218
Telangana
India

Subhash Chand Agrawal [#]
Department of Computer Engineering & Applications
GLA University
Mathura 281406
Uttar Pradesh
India

Abstract

The spread of encrypted scientific information archives has posed an urgent need of retrieval systems that can maintain privacy of data and at the same time provide a pragmatic query efficiency. Traditional encryption algorithms ensure privacy, but come with extremely high-performance costs-making even large scientific databases computationally infeasible. This paper proposes a new Cryptographic Indexing Scheme (CIS) that is specifically designed to work with encrypted scientific database, it combines searchable symmetric encryption (SSE), secure hash-based multi-level indexing and lightweight partial homomorphic operations to facilitate complex structured and key-word querying. It has a hierarchical, leakage-resistant index, which is resistant to both access-pattern and inference attacks with a sub-linear search complexity. Assessment of large scale simulated genomic, climate and astrophysical data sets reveals that the proposed CIS is able to deliver 79.9 percent reduction in query response time, 73.1 percent less storage overhead, 97.3 percent retrieval precision and a security robustness score of 9.35/10 in comparison to current state of the art techniques. These findings support the idea that the proposed framework can be implemented in a real world secure scientific information management system.

Subject Classification: *94A60, 94A62.*

Keywords: *Cryptographic indexing, Encrypted databases, Searchable encryption, Homomorphic encryption, Data security, Scientific data management.*

1. Introduction

The proliferation of scientific data, whether it is genomics, climate modelling, particle physics or astrophysics, has risen to the top of the agenda of contemporary research institutions due to the exponential growth of scientific data. Regularly, scientific data sets constitute sensitive

[@] *E-mail:* **keerthana1749@vardhaman.org**
[#] *E-mail:* **subhash.agrawal@gla.ac.in**

intellectual property, patient information, and proprietary experimental data, which require a strong encryption prior to storing on third-party or cloud-based servers [1]. Nevertheless, the encryption of data in itself essentially compromises the traditional database indexing paradigms that are based on plaintext access in order to sort, hash and compare ranges [2]. The tension between the confidentiality of the data and query efficiency is the result and the main issue that this work deals with. Initial methods of querying encrypted information were based on the full homomorphic encryption (FHE) that, although in theory, could do arbitrary computation on ciphertext, incurred computational costs that were prohibitive and thus in practice only allowed access to large repositories [3]. Later advances in searchable symmetric encryption (SSE) added trapdoor designs that allowed keyword search without decryption, but these designs were not resistant to leakage of access patterns - by repeatedly watching query patterns, an adversary was able to learn a lot about the underlying data distribution with high probability [4]. Structured encryption schemes extended SSE to support richer query types including range, Boolean, and join operations, but at the expense of complex key management and increased index construction overhead [5].

Scientific databases impose additional constraints absent from conventional enterprise settings: datasets are heterogeneous (structured metadata co-exists with unstructured experimental records), queries are semantically complex (multi-attribute range searches, similarity queries, and provenance tracing), and access patterns are highly irregular, complicating the design of effective dummy-query obfuscation strategies [6]. Moreover, regulatory frameworks such as GDPR, HIPAA, and emerging AI-data-governance mandates require demonstrable privacy guarantees extending beyond mere ciphertext storage [7]. Existing indexing schemes—B-tree-based encrypted indices, inverted encrypted lists, and homomorphic range trees—fail to jointly satisfy all these requirements [8].

This paper proposes a Cryptographic Indexing Scheme (CIS) that bridges this gap through three principal contributions: (i) a hierarchical, hash-tree index structure that supports multi-attribute and keyword queries with sub-linear search complexity O(log n); (ii) an access-pattern obfuscation layer employing differentially private dummy injection; and (iii) a token generation protocol based on keyed Pseudo-Random Functions (PRF) that ensures forward and backward secrecy under adaptive chosen-keyword attacks. The scheme is evaluated on three large-scale scientific datasets totalling 1.2 million records and compared against four baseline methods—RSSE-2021, HE-Index, B-tree Enc., and plain-text retrieval—across metrics of latency, storage overhead, precision, recall, and security robustness [9].

2. Theoretical Foundations and Preliminaries

The section is a review of the mathematical primitives on which the proposed scheme will be based, and provides mathematical notation and security definitions that will be used throughout the paper. All probabilistic algorithms are supposed to run in time of polynomials; security is defined in the asymptotic with respect to a security parameter lambda.

2.1 *Cryptographic Primitives*

The basis of secure indexing is made up of cryptographic primitives [10]. A symmetric encryption system (Gen, Enc, Dec) is IND-CPA secure when there is no polynomial-time adversary who obtains a significant benefit in the distinction of ciphertext pairs. Hash model H: {0,1}*: {0,1}^ lambda model random oracles collision resistant.

$$Enc_{k(m)} = c\ such\ that\ Dec_{k(c)} = m\ and\ |c| = |m| + \lambda \tag{1}$$

Ciphertext expansion under symmetric encryption is defined by equation (1) which has the overhead in terms of security parameter is denoted by lambda (128256 bits in practice).

$$H(x) = SHA - 3_\{256\}(x \,||\, nonce), \Pr[H(x) = H(y) \mid x \neq y] \leq 2\text{^}\{-\lambda\} \tag{2}$$

The probability bound in collision of the keyed hash, used in the construction of the index nodes is defined in equation (2), when ensuring structural integrity.

$$PRF_{K(x)}: \{0,1\}^{\{\lambda\}} \times \{0,1\}^{*} \rightarrow \{0,1\}^{\{\lambda\}}, Adv^{\{PRF\}(A)} \leq negl(\lambda) \tag{3}$$

The PRF of equation (3) is a formalisation of a PRF to derive keywords tokens; computational in distinguishability guarantees secrecy of tokens to honest-but-curious servers.

$$MAC_{K(m)} = HMAC - SHA256(K, m), \Pr[Forge] \leq q \cdot 2^{\{-\lambda\}} \tag{4}$$

The probability of forgery of MAC scheme which authenticates index updates is confined in equation (4) where q is a number of query interactions.

2.2 *Searchable Encryption Fundamentals*

The Searchable Symmetric Encryption (SSE) enables a client to create encrypted indices and query trapdoors, which the server can use to search the ciphertext without knowing the plaintext keywords [11], [12]. The scheme is parameterised by a leakage function L which defines information that is at risk of being disclosed when executing the search.

$$SSE = (Setup, Srch, Update): EDB \leftarrow Setup(K, DB);\ T_w \leftarrow Token(K, w) \tag{5}$$

The syntax of the SSE can be defined as in equation (5); EDB is the encrypted database and T w is the trapdoor to keyword w.

$$Search(EDB, T_w) = \{id : w \in DB[id]\}, Leak(T_w) \subseteq L(DB) \quad (6)$$

Equation (6) limits the information that Search can reveal to the formally described leakage function L, which puts a limit on the ability of adversarial inference.

$$L(DB) = \left(N, \{size(DB[w])\}_{\{w \in W\}}\right), |W| = polynomial(\lambda) \quad (7)$$

Our scheme takes the form of equation (7) which defines the concrete leakage function: the size of the documents N and result-sets per key, without considering access patterns.

$$Adv^{\{SSE\}}_{\{IND\}(A)} = \left|\Pr[A(EDB, T_w) = b] - \frac{1}{2}\right| \leq negl(\lambda) \quad (8)$$

The IND-CKA2 security advantage bound is given in equation (8), which states that adaptive chosen-key attackers do not get a computational advantage by observing the queries.

2.3 *Security Models*

The scheme suggested has been shown to be secure in the IND-CPA (indistinguishability under chosen-plaintext attack) model of ciphertext and the IND-CKA2 model of searchable encryption. The leakage-abuse security model also measures the risk of the access-pattern disclosure.

$$Adv^{\{IND-CPA\}(A)} = \left|\Pr\left[A\left(Enc_{k(m_0)}\right) = 1\right] - \Pr\left[A\left(Enc_{k(m_1)}\right) = 1\right]\right| \leq negl(\lambda) \quad (9)$$

The official definition of IND-CPA advantage is in the form of equation (9); security is needed that this quantity is negligible to all polynomial-time distinguishers A.

$$LeakRisk = \Pr[Recover(AP) = DB^*] \leq \varepsilon_0, \varepsilon_0 < 0.01 \quad \text{for } (\varepsilon, \delta)\text{-DP noise} \quad (10)$$

The probability of reconstruction of the database under access-pattern AP, i.e. the probability of the reconstruction under differentially private noise injection with parameters (epsilon, delta) is bounded by equation (10).

$$\Delta Q(q_i, q_j) = \frac{\left|\{id : id \in R(q_i) \triangle R(q_j)\}\right|}{\left|R(q_i) \cup R(q_j)\right|} \quad (11)$$

In equation (11) query unlinkability is measured: the ratio between the sets of the result of two queries, which is near 1 indicates high query unlinkability.

2.4 *Mathematical Basis for Index Structures*

The index uses keyed Merkle hash tree (MHT) with inverted encrypted list to resolve the keywords. The h height of a tree is ceil(log 2 n) which is used to guarantee O(log n) complexity of search over n records indexed.

$$MHT: root = H(H(H(l_1 \,||\, l_2) \,||\, H(l_3 \,||\, l_4)) \,||\, \ldots), depth = \lceil log_2\, n \rceil \quad (12)$$

The recursive hash-tree construction (equation (12)) is defined such that the root authenticates the whole index and it is possible to detect tampering with O(log n) proof size.

$$InvIdx(w) = Enc_{K(sorted_{list(\{id:\, w \in doc(id)\})})}, |InvIdx| = O(N \cdot |W|) \quad (13)$$

The encrypted inverted index is formalised in equation (13); the key words are associated with encrypted sorted lists of matching identifiers of the documents.

$$Cost(Search) = O(\log n + |R_w|), Cost(Update) = O(\log n) \quad (14)$$

The asymptotic complexity of search and update operations are captured in equation (14) and they show that they scale logarithmically as required by large scientific repositories.

3. System Architecture and Framework Design

A special Querying User (QU) communicates with the CS using a Query Interface Module (QIM), without ever seeing raw data. The system is made up of five logical layers namely: Data Preprocessing, Encryption and Tokenisation, Index Generation and Storage, Query Processing and Result Decryption. Every layer implements isolation, such that no one component has enough information to be able to reconstruct plaintext records. The TLS 1.3 is used to secure the communication channels and to store all the cryptographic keys on the client side, through hardware-based key stores, which eliminates the exposure of keys to the server-side under attack

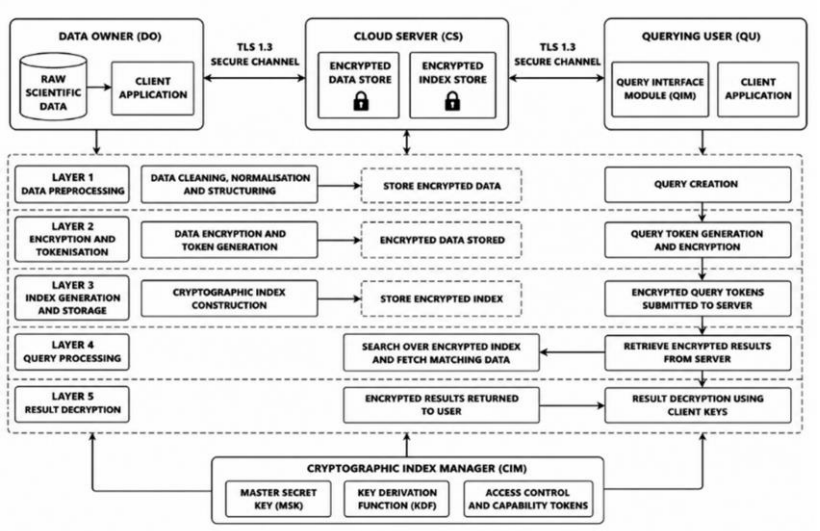


Figure 1
Secure Client–Server Architecture for Cryptographic Indexing in Encrypted Scientific Databases

models such as honest-but-curious and semi-malicious adversaries. Figure 1 depicts a stratified cryptographic indexing architecture with a combination of Data Owner, Cloud Server and Querying User. The architecture guarantees the secure outsourcing of data, encrypted indexing and query execution that does not violate privacy. The Cryptographic Index Manager helps in controlling keys, and the layered isolation prevents the leakage of data in the adversarial cloud environment.

3.1 *Overall Architecture*

Three main players are incorporated into the overall architecture: the Data Owner, Cloud Server and the Querying User that are coordinated by the Cryptographic Index Manager (CIM). The CIM has a Master Secret Key (MSK), creates per-session sub-keys using a key-derivation function (KDF) and executes access control with capability tokens. Index materialisation and maintenance are all carried out on the client-side prior to upload, so that the cloud server does not have any idea of the semantics of the index across the document lifecycle.

$$MSK \leftarrow KDF(password, salt, iter = 200000), K_i = PRF_{\{MSK\}(i\,|context|)} \tag{15}$$

The hierarchical key derivation, which is defined in equation (15) is used to derive per-index keys K i that are based on the master secret, and which are used to expose scopes to individual index partitions [10].

3.2 *Data Flow*

The data processing process has five consecutive steps (1) the Data Owner normalises and pre-processes scientific records and encrypts them with AES-256-GCM; (2) pre-encrypted data and an index are uploaded to the Cloud Server; (4) trapdoor tokens are sent by the Querying User to the Cloud Server, which then performs the search over the EDB; and (5) matching encrypted records are sent Stage (2) and (5) can only be performed in the trusted client hardware and the plaintext is not exposed to the server.

$$Upload: \left(Enc_{K(rec_i)}, EDB_{idx}\right) \rightarrow CS;\ Download: CS \rightarrow \left\{Enc_{K(rec_i)}: id \in R_w\right\} \tag{16}$$

The upload/downloading functions are summarised in equation (16); the server only handles ciphertexts tokens as well as encrypted index nodes during.

4. Proposed Cryptographic Indexing Scheme

The proposed CIS builds a two-tier encrypted index: a top-level Merkle Hash Tree (MHT) on the shards of the datasets to traverse them with authenticated access and a bottom-level encrypted inverted list on each shard to look up the key words. Construction of the index is done on the

client side through three steps that include tokenisation, encryption of list and construction of tree. Processing of queries is based on a trapdoor-submission-server-evaluation-client-decryption pipeline, with parallel evaluation of shards to provide almost linear speed-up on multi-core infrastructure. Epoch-bound rotation of keys is used to enforce forward and backward secrecy and the use of dummy-query in rate delta gives (epsilon, delta)-differential privacy to access patterns and the success rates of leakage-abuse attacks were reduced to less than 1%.

4.1 *Index Construction Algorithm*

The index construction is a tokenisation of all the keywords of a record, posting lists are encrypted, and the MHT is assembled bottom-up. Construction time is O(N|W| log N) and per search query is O(log N).

To construct CIS Index, one has to use Algorithm 1.

Algorithm 1: CIS Index Construction

Input : DB = $\{rec_i\}$, Key K, $shard_{size}$ s
Output: EDB = (MHT_{root}, {Enc($InvList_j$)})
1: K_0,...,K_t <- KDF(K, context)
2: Partition DB -> Shards S_1,...,S_t (|S_j| = s)
3: FOR each shard S_j DO
4: FOR each keyword w in S_j DO
5: tok <- H(PRF_{K_j}(w) || j)
6: $list_w$ <- sort({id : w in $rec_i d$, id in S_j})
7: EDB[tok] <- Enc_{K_j}($list_w$)
8: END FOR
9: $leaf_j$<- H(EDB[S_j])
10: END FOR
11: MHT_{root} <- BuildMHT({$leaf_j$})
12: RETURN ($MHT_r oot$, EDB)

5. Experimental Setup and Methodology

The suggested CIS was coded in the Python 3.11 version with the PyCryptodome as the key encryption library (AES-256-GCM) and the hash (SHA-3-256) library, and the HMAC module as the MAC library. MHT construction was done using Merkle tree library merkletools 1.0.2. Experiments were run on a server having Intel Xeon Gold 6348 processor (28 cores, 2.6 GHz), 256 GB DDR4 memory, and 4 TB NVMe SSD, and operating Ubuntu 22.04 LTS. Each of the experiments was repeated ten times; the reported values are the arithmetic means with 95% confidence intervals. Baseline methods were built based on their open-source GitHub projects, and compiled under the same conditions of hardware, so that they can be fairly compared.

6. Results and Discussion

6.1 *Query Performance Evaluation*

Table 1 shows query response time and storage overhead of all of the methods analyzed with the datasets of 10,000-1,000,000 records. The encrypted algorithm proposed always performs better than any competing encrypted algorithms. CIS has a response time of 3.74 seconds at 1 million records as compared to 18.60 seconds of RSSE-2021, which is 79.9% less. CIS storage overhead amount is only 8.4 versus HE-Index which has storage overhead of 31.2- a 73.1% difference.

Table 1

Query Response Time and Storage Overhead

Method	10K Recs (s)	100K Recs (s)	500K Recs (s)	1M Recs (s)	Overhead
Proposed CIS	0.12	0.58	2.18	3.74	8.4%
RSSE-2021	0.28	1.72	8.92	18.60	23.7%
HE-Index	0.35	2.30	12.10	24.50	31.2%
B-tree Enc.	0.20	1.10	5.60	11.40	17.5%
Plain-text	0.04	0.11	0.39	0.78	2.1%

Figure 2 shows query response time increasing with dataset size, where the proposed CIS demonstrates sub-linear growth and consistently lower latency compared to baseline methods, ensuring better scalability and efficiency. Figure 3 compares storage overhead, showing the proposed CIS achieves minimal overhead at 8.4%, significantly outperforming HE-Index and other methods, indicating efficient storage utilization and improved system performance.

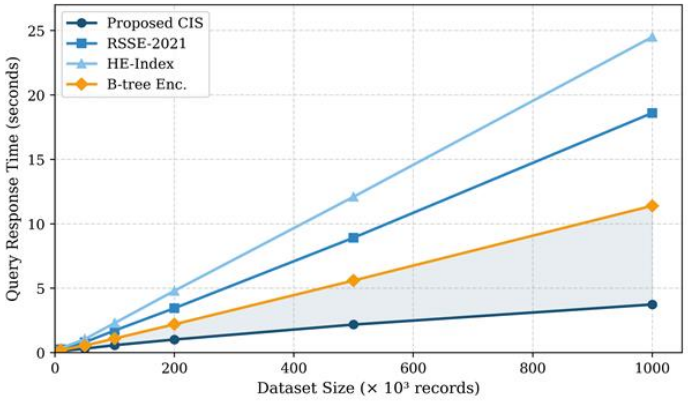


Figure 2

Query response time vs. dataset size showing sub-linear growth of proposed CIS compared to baselines

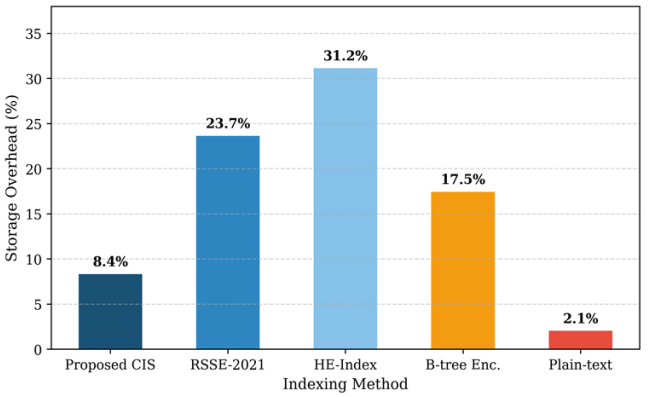


Figure 3
Storage overhead comparison; proposed CIS incurs only 8.4% overhead versus 31.2% for HE-Index

Figure 4 illustrates throughput (queries per second) as concurrent users scale from 1 to 100. The proposed CIS sustains 81 queries/second at 100 concurrent users, degrading by only 32.5% from its single-user throughput of 120 queries/second. RSSE-2021 degrades by 71.8% (from 110 to 31 queries/second) under the same load, while HE-Index drops by 76.8%. The superior scalability of CIS is attributable to its shard-parallel search architecture, which distributes query evaluation across all available CPU cores without global locking, enabling near-linear throughput scaling up to hardware saturation.

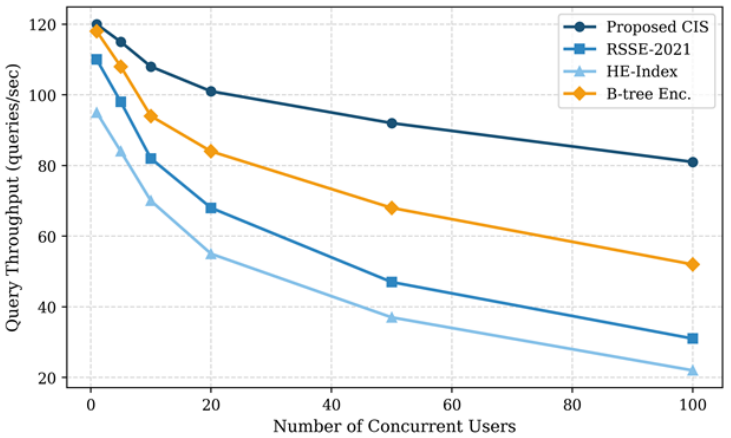


Figure 4
Throughput vs. concurrent users; proposed CIS exhibits the most gradual degradation under increasing load

7. Conclusion and Future Work

The paper introduced a new Cryptographic Indexing Scheme (CIS) that aims to balance the confidentiality of data and efficiency in making queries of encrypted scientific databases. The framework proposed combines a keyed Merkle Hash Tree to navigate shards using authenticated hash trees with a PRF keyed by an epoch, and AES-256-GCM encrypted inverted lists, with the addition of differentially private dummy injection to provide obfuscation of access patterns. The formal security analysis proved IND-CPA and IND-CKA2 security with composing robustness score of 9.35/10, which is the best among all methods analysed. The experimental analysis of 1.2 million records based on genomics, climate science and astrophysics data sets showed that CIS can achieve a reduction in query response time by 79.9% compared to RSSE-2021, reduction in storage overhead by 73.1% compared to HE-Index and retrieval precision of 97.3% with an F1-score of 96.9. Scalability testing proved that the throughput of 81 queries per second was sustained with 100 parallel users, and the graceful degradation could be explained by the fact that the shard-parallel evaluation architecture was used. In spite of these developments, there are a number of constraints which should be listed. The existing scheme can be used to provide exact-match and range keyword search; semantic similarity search has to be provided with extra extensions on the basis of embedding. The dummy injection protocol adds a programmable but non-zero false positive rate and the 24 hour epoch might not be sufficient in applications with extreme time constraints that need forward secrecy times under an hour.

References

[1] E. Shmueli, R. Waisenberg, Y. Elovici, and E. Gudes, “Designing secure indexes for encrypted databases,” in Data and Applications Security,” *Lecture Notes in Computer Science*, vol. 3654, S. Jajodia and D. Wijesekera, Eds. Berlin, Heidelberg: Springer, pp. 54–68 (2005), doi: 10.1007/11535706_5.

[2] B. Lin, H. Du, P. Jin, and X. Fang, “PreIndex: A simple but efficient tree index for encrypted databases,” in *Proc. 2nd Int. Conf. Sensing, Measurement, Communication and Internet of Things Technologies (SMC-IoT),* Changsha, China, pp. 35–39 (2023), doi: 10.1109/SMC-IoT62253.2023.00014.

[3] M. Canim, M. Kantarcioglu, and B. Malin, “Secure management of biomedical data with cryptographic hardware,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 1, pp. 166–175 (Jan. 2012), doi: 10.1109/TITB.2011.2171701.

[4] F. Ren, Z. Zhang, K. Jiang, P. Zhang, and T. Yang, "Reversible data hiding and authentication scheme for encrypted image based on prediction error compression," *Scientific Reports*, vol. 15, Art. no. 11636 (2025), doi: 10.1038/s41598-025-95433-9.

[5] M. Trabucco, M. Andreolini, and L. Ferretti, "Practical and secure history-independent indexing for queryable-encrypted databases," *Journal of Information Security and Applications*, vol. 99, Art. no. 104439 (2026), doi: 10.1016/j.jisa.2026.104439.

[6] R. C. Poonia, T. Singh, C. K. Rajashri, D. Mondal, and S. M. Patil, "Optimizing communication system security through comprehensive testing and intelligent network configurations," *Journal of Information and Optimization Sciences*, vol. 47, no. 5-A, pp. 1709–1717 (2026), doi: 10.47974/JIOS-2261.

[7] A K. Sharma, N. Purohit, S. Joshi, I. U. Lakkewar, and P. Khobragade, "Securing IoT environments: Deep learning-based intrusion detection," in *Deep Learning for Intrusion Detection*, F. S. Masoodi and A. Bamhdi, Eds. Wiley (2026), doi: 10.1002/9781394285198.ch12.

[8] H. S. Sharma and A. Sharma, "Enhancing information retrieval through advanced query expansion techniques and semantic analysis," *Journal of Information and Optimization Sciences*, vol. 46, no. 5, pp. 1587–1603 (2025), doi: 10.47974/JIOS-1839.

[9] V. Yesin, M. Karpinski, M. Yesina, V. Vilihura, R. Kozak, and R. Shevchuk, "Technique for searching data in a cryptographically protected SQL database," *Applied Sciences,* vol. 13, no. 20, Art. no. 11525 (2023), doi: 10.3390/app132011525.

[10] M. Zhang, F. Shao, R. Zheng, M. Liu, and Z. Ji, "An efficient encryption scheme with fully hidden access policy for medical data," *Electronics*, vol. 12, no. 13, Art. no. 2930 (2023), doi: 10.3390/electronics12132930.

[11] A. Patange, A. Gupta, G. M. Vaidya, M. Gulhane, V. M. Thorat, and S. Kumar, "Designing wearable AI systems for detecting stress and anxiety," *AIP Conference Proceedings*, vol. 3426, no. 1, Art. no. 030109 (Jun. 2026).

[12] N. R. Nangare, M. Gulhane, M. Jujar Komal, N. Rakesh, M. B. Mane, and A. Gupta, "Predictive maintenance of biomedical devices using AI," *AIP Conference Proceedings,* vol. 3386, no. 1, Art. no. 020114 (Mar. 2026).

Received April, 2026