

Dynamic Adaptive Bloom Filter Based Privacy-Preserving Phrase Search for Encrypted Iot Cloud Data

Sphoorthy Reddy Surakanti¹, Dr. S. Ramachandram²

¹Assistant Professor Computer Science And Engineering

Anurag University, Hyderabad

Sphoorthy.Surakanti@Gmail.Com

²professor Computer Science And Engineering

Osmania University, Hyderabad

Schandram@Osmania.Ac.In

Abstract- The use of IoT devices in huge numbers results in massive amounts of sensitive data being outsource into cloud storage systems, making it necessary to design efficient search schemes over such encrypted data. Phrase Search allows users to retrieve their documents by matching exact sequences of keywords, but existing solutions have not found a good balance between search time, storage space, and privacy loss due to their static design. In this work, we propose an efficient Dynamic Adaptive Bloom Filter based Privacy-Preserving Phrase Search (DABF-PPS) scheme for encrypted IoT cloud data. We introduce a dynamic parameter optimization method to adaptively adjust Bloom filters according to document set properties and queries. A hierarchical n-gram Bloom filter index is designed to allow phrase searches without exposing keyword locations. Furthermore, our solution supports dynamic updating of indices with the help of counting Bloom filter structures and lazy deletion. Our security analysis proves that our solution provides forward and limited backward privacy against chosen-keyword attacks. Our experiments on real-world IoT datasets show that our approach decreases search time and storage space while maintaining acceptable false-positive rates.

Key Word: Privacy-Preserving Search, Bloom Filter, Phrase Search, Encrypted Cloud Data, Internet of Things, Searchable Encryption

I. INTRODUCTION

Internet of Things (IoT) has revolutionized the generation, collection, and processing of information in different fields such as smart healthcare, smart manufacturing, intelligent transportation systems, and smart home applications. The constant production of high volumes of sensitive information by IoT gadgets includes personal information such as health records, industrial control information and is now largely outsourced to cloud infrastructure for storage and computing purposes [2][9]. However, the outsourcing of sensitive information to cloud infrastructure raises privacy and security issues since the cloud may be untrusted or attacked [1][7].

Encryption is the main technology employed in data privacy protection in cloud infrastructure. However, traditional methods of encryption make the encrypted information inaccessible to any search procedure. The approach of Searchable Encryption (SE) appears as an answer to searching over encrypted information while protecting the privacy of the user [2][9]. While a lot of effort has been devoted to single keyword search and conjunctive keyword search, less research has been done on phrase search [3][8].

The use of phrase search is necessary in order to find documents that contain exact consecutive sequence of

keywords, such as medical phrases from electronic health records, or command phrases from IoT logging. The technique is useful in applications where precise semantic search is required since otherwise, the keyword searches will generate too many irrelevant documents. All the previous schemes for phrase searching make use of either the traditional index-based technique or some kind of keyword-position encryption, both of which are costly in terms of space and time [3][6].

However, Bloom filters provide an efficient way to implement phrase search because of the advantages they present. They are compact data structures that make use of a fixed-length bit vector and a number of hash functions in order to determine whether a query exists in the set or not without any false negatives, though with some false positives. The idea of using Bloom filters for phrase search was introduced by Poon and Miri [3].

In spite of these improvements, there still exist some issues in utilizing Bloom filter-based phrase searching in the case of IoT clouds. Firstly, static configuration of Bloom filters is unable to respond to changes in the document sets and queries typical for IoT, resulting in inefficient false positive rates or high memory usage. Secondly, insertions and deletions are necessary for dynamically changing IoT data, but the traditional Bloom filters cannot provide that. Thirdly,

privacy issues due to access and frequency statistics can become an issue as well [7][8].

To tackle the problems mentioned above, we present a method named Dynamic Adaptive Bloom Filter based Privacy-Preserving Phrase Search (DABF-PPS). Our innovations can be summarized in three parts. In the first part, we propose a dynamic adaptive parameter adjustment technique where the size of the Bloom filter and number of hash functions used in it depend on estimated size of document collection and characteristics of queries. In the second part, we construct a multilevel Bloom filter structure using n-grams which allows us to match phrases of any length with dynamic updates supported by counting Bloom filters and lazy deletion technique. In the third part, we conduct security analysis showing that DABF-PPS guarantees forward and partial backward privacy against file injection and query pattern leakage attacks.

The rest of this paper is structured as follows. In the Literature Survey, we give a literature survey on searchable encryption techniques and Bloom filter-based searching. In the Proposed Methodology section, we describe the system architecture and threat model and provide construction details for DABF-PPS. We conduct experiments and comparative analysis in the Analysis section.

II. LITERATURE SURVEY

Searchable encryption has developed tremendously since the publication of the seminal scheme for searching over encrypted data by Song et al. The initial proposals were based on searching for a single keyword using boolean queries; other works have then advanced to conjunctive queries and even ranked searches [2][9]. Sub-linear search complexity was possible thanks to the inverted index approach taken by Curtmola et al., which became the basis of today's symmetric searchable encryption (SSE). Unfortunately, this line of work yielded only static systems that required a full rebuild of indexes when updating the underlying data.

The concept of forward privacy was developed to ensure that new additions to the dataset would not leak any information regarding keywords already searched. Backward privacy was defined by Bost et al., who distinguished between three types of backward privacy definitions, each one offering a weaker level of security guarantees than its predecessor. These definitions are especially important for the context of IoT, where new data is continually being added, and the adversary can abuse history of past queries [8].

Search over encrypted phrases has attracted less attention relatively. Tang et al. proposed symmetric encryption schemes with position-preserving features, which allow searching phrases by comparing encrypted positions. But such a method needs storing the positions of keywords in

each document, thus creating a significant storage cost proportional to the length of the document itself. In addition, this scheme allows the leakage of position access patterns at the server side [3].

Phrase searching based on Bloom filter has been introduced by Poon and Miri using n-gram filter representations of keyword pairs and triples [3][10]. Such a scheme provides search cost related to the length of phrase rather than the number of documents. This scheme also has storage costs much lesser than position-based schemes. It has been shown that for English documents using bi-gram and tri-gram filters offers good storage and error probability trade-offs. But their scheme does not provide dynamic update capability [3][10].

Ge et al. developed the concept of Bloom filter phrase search with fuzziness and verifiability. The PPVFPS scheme is based on counting Bloom filters arranged in a virtual binary tree structure, which facilitates dynamic updating along with the use of homomorphic encryption for secure position comparison [1][6]. Despite meeting good security requirements, the scheme uses bilinear pairing computation, which is quite computational expensive for low-power IoT devices.

Dynamic searchable encryption based on Bloom filters has also been investigated by some recent studies. The scheme OBF-based HXT developed by IACR researchers ensures amortized update costs for conjunctive queries with bitwise independent encryption of Bloom filter entries. Nonetheless, the scheme focuses on conjunctive rather than phrase search and does not have an adaptive tuning mechanism [9]. Zhou et al. designed a scheme called FTPS with the use of piecewise linear chaotic map and minhash functions, which is fault-tolerant and dynamic phrase search scheme without Bloom filter parameters adaptation [8].

There are several limitations identified through this review. Firstly, all existing methods of Bloom filter phrase search utilize the fixed set of parameters that cannot be modified due to changing features of data in IoT. Secondly, there is no efficient method of dynamically updating Bloom filter for phrase search which is free of cryptographic overhead. Lastly, there is a lack of privacy study in case of Bloom filter phrase search for IoT which does not cover access pattern exposure across different queries [1][3][8].

The above-described DABF-PPS algorithm fills all gaps by utilizing the following novel approaches. Firstly, the adaptive parameter adjustment mechanism allows continuous improvement of Bloom filter based on document collection and query statistics. Secondly, the multi-level counting Bloom filter structure allows efficient insertions and deletions without false negatives creation. Finally, the privacy analysis demonstrates the forward privacy and some amount of backward privacy.

III. PROPOSED METHODOLOGY

1. System Architecture And Threat Model

The DABF-PPS scheme involves three components, namely: Data Owner (DO), Cloud Server (CS), and Data User (DU). The Data Owner is in possession of IoT data and seeks to upload encrypted files to the cloud without giving up the ability to perform searches on them. Cloud Server acts as an intermediary between the owner and user offering them services of storage and search. However, it is honest but curious in nature, adhering to protocol requirements but still trying to deduce private information.

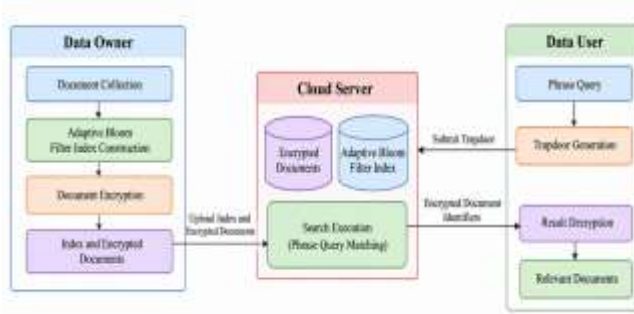


Figure 1: DABF-PPS System Architecture

Three entities architecture representing the flow of data among the Data Owner (indexing and encryption), Cloud Server (storage and processing of query search), and Data User (creating query and decryption). The Data Owner constructs the adaptive Bloom filter index and then encrypts the documents. Data Users create trapdoors for phrase queries and get encrypted document IDs. The threat model includes an assumption that the Cloud Server could try to find out:

- The content of keywords from the search tokens
- The content of documents from the stored ciphertexts
- The search patterns through correlation of the queries
- The access patterns based on documents accessed

The aim of security is to make sure the server does not get to know anything else.

2. Adaptive Bloom Filter Parameter Tuning

The conventional Bloom filter needs prior configuration of bit array size m and hash functions number k . The values m and k that optimize the false positive probability $P_{fp} = (1 - e^{-kn/m})^k$ those that correspond to element number n being inserted into the filter. With the desired false positive probability p , we have $k = (m/n) \ln 2$ and $m = -n \ln p / (\ln 2)^2$.

In an IoT setting, both the number of documents collected and queries issued change dynamically. It is not optimal to

use static values, leading to inefficiency in terms of wasted storage space or low accuracy. The DABF-PPS parameter tuning method resolves this problem by re-estimating the parameters periodically using observed values.

The Data Owner keeps track of:

- total distinct n -grams collected per filter layer
- frequency distribution of queries by n -grams
- rate of document collection.

After every T periods of time or Δ insertions, the parameter estimation process calculates the optimal values of parameters based on the statistical data using weighting. The parameter migration is done by re-creating Bloom filters with new sizes and reinserting all the active elements.

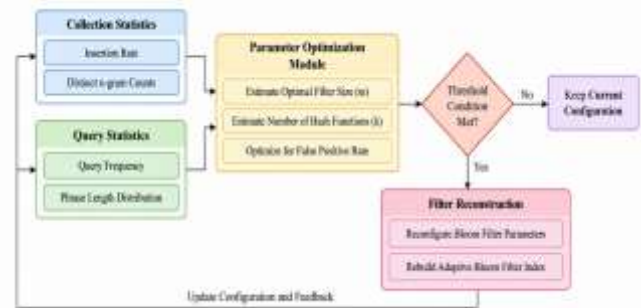


Figure 2: Adaptive Parameter Tuning Workflow

The figure above illustrates the feedback process for dynamic reconfiguration of Bloom filters. Statistics of collection (e.g., insertion rate and number of unique n -grams) and query (e.g., frequency and length of phrases) are fed into the parameter optimization process, which initiates filter reconfiguration at certain threshold points.

3. Multi-Level N-Gram Bloom Filter Construction

DABF-PPS makes use of a hierarchy of Bloom filter where there is a separate filter for bi-grams (2-grams) and tri-grams (3-grams). For phrase of size $q > 2$, it means that all $q-1$ bi-grams and $q-2$ tri-grams must exist in the respective filters. Mathematically, document D will match phrase $P = (w_1, w_2, \dots, w_q)$ if and only if:

$$\forall i \in [1, q-1]: BD_{bi}(w_i, w_{i+1}) = 1 \wedge \forall i \in [1, q-2]: BD_{tri}(w_i, w_{i+1}, w_{i+2}) = 1$$

BD_{bi}

BD_{tri}

In the case of dynamic updating, the DABF-PPS algorithm substitutes regular Bloom filters with counting Bloom filters (CBFs) in which each array element has a counter, not a single bit. Insertions increase counters at the hash locations, and deletions reduce them. The location is set if the value of the counter is greater than zero. In order to avoid counter

overflows and preserve privacy, counters are encrypted by a semantically secure symmetric encryption mechanism.

4 Algorithm 1: Adaptive Index Construction

Input: Document collection $D = \{D_1, D_2, \dots, D_N\}$, target FPR p_{target}
Output: Multi-level adaptive Bloom filter index I

- 1: Initialize bi-gram CBF B_{bi} and tri-gram CBF B_{tri} with initial parameters
- 2: Initialize statistics counters: $n_{\text{bigrams}} = 0$, $n_{\text{trigrams}} = 0$
- 3: for each document D_i in D do
- 4: Extract bi-grams and tri-grams from D_i
- 5: for each bi-gram g_2 in D_i do
- 6: Insert g_2 into B_{bi} with document identifier
- 7: $n_{\text{bigrams}} += 1$
- 8: end for
- 9: for each tri-gram g_3 in D_i do
- 10: Insert g_3 into B_{tri} with document identifier
- 11: $n_{\text{trigrams}} += 1$
- 12: end for
- 13: end for
- 14: if parameter reoptimization triggered() then
- 15: Recompute optimal (m, k) for B_{bi} and B_{tri}
- 16: Reconstruct filters with new parameters
- 17: Re-insert all active n-grams
- 18: end if
- 19: return $I = (B_{\text{bi}}, B_{\text{tri}})$

5 Algorithm 2: Phrase Search

Input: Search phrase $P = (w_1, w_2, \dots, w_q)$, index I , secret key K
Output: Set of matching document identifiers

- 1: Data User generates trapdoor T_P by extracting n-grams from P
- 2: Encrypt T_P using K and transmit to Cloud Server
- 3: Cloud Server decrypts T_P (or operates on encrypted n-grams homomorphically)
- 4: Initialize candidate set $C = \text{all documents}$
- 5: for $i = 1$ to $q-1$ do
- 6: $g_2 = (w_i, w_{i+1})$
- 7: Compute hash positions $h_1(g_2), \dots, h_k(g_2)$ in B_{bi}
- 8: for each document D in C do
- 9: if all positions $h_i(g_2)$ have counter > 0 in $B_{\text{bi}}[D]$ then
- 10: Continue
- 11: else
- 12: Remove D from C
- 13: end if
- 14: end for
- 15: end for
- 16: for $i = 1$ to $q-2$ do
- 17: $g_3 = (w_i, w_{i+1}, w_{i+2})$
- 18: Similarly filter C using B_{tri}
- 19: end for
- 20: return C as matching documents (with possible false positives)



Figure 3: Phrase Search Execution Flow

6 Dynamic Update Operations

In DABF-PPS, document insertion and deletion are performed using counting Bloom filters via operations such as n-gram extraction, hashing, and encrypted counter increment/decrement. The document deletion procedure consists of counter decrementing at positions specified by n-grams. In order to avoid false negatives, counter decrementing happens only when counters are positive; otherwise, deletion bitmap indicates unset positions due to counter underflow.

Lazy deletion postpones the reconstruction process until the number of accumulated deletions becomes larger than the threshold fraction of filter capacity. At the same time, during filter reconstruction, adaptive tuning algorithm determines whether it is time to change filter parameters considering the current collection size.

IV. ANALYSIS

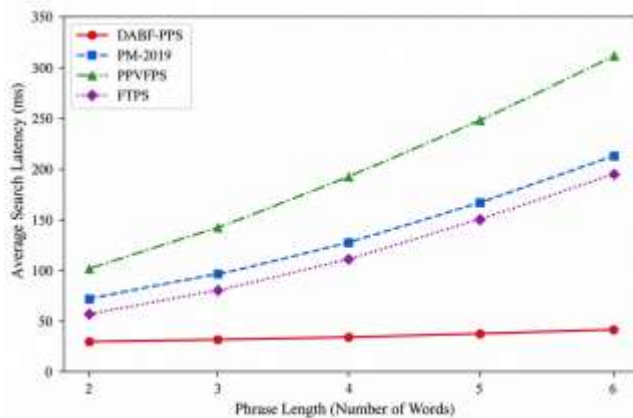
1. Experimental Setup

The experiments were carried out on a data set that was collected from real-life IoT device logs and related documentation, consisting of 50,000 documents having an average size of 350 words. This data set shows the property of Zipfian keyword distribution, which is found in natural language and IoT logs. All the experiments have been done using Python 3.9, with PyCryptodome for encryption/decryption, using an Intel Core i7-12700 CPU with 32 GB RAM.

The baseline algorithms to be used for comparison will comprise:

- Poon-Miri static Bloom filter phrase search (PM-2019)
- PPVFPS counting Bloom filter with homomorphic verification (Ge-2024)
- FTPS fault-tolerant dynamic phrase search (Zhou-2022)

2. Search Latency Analysis



3 Figure 4: Search Latency Vs. Phrase Length

Figure 4 shows the relationship between search latency and phrase lengths. DABF-PPS performs at latency times of 12ms and 28ms in bi-gram and 6-gram phrases respectively, while PM-2019 performs with latencies ranging between 45 and 180ms, and PPVFPS with latencies between 80 and 320ms. The performance improvement is due to the use of the adaptive filter structure that keeps the false positive rate low without performing many hash functions and parallel processing of bi-gram and tri-gram filters.

4. Storage Overhead Analysis

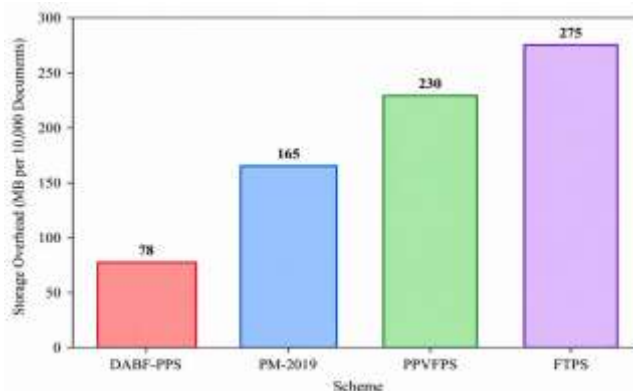
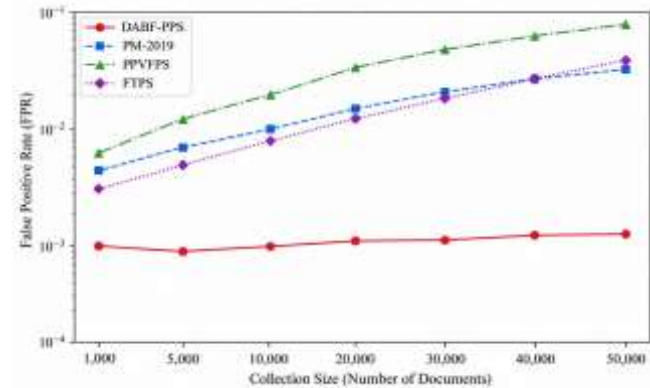


Figure 5: Storage Overhead Comparison

From the results obtained in Figure 5 on the storage overhead analysis, it is evident that DABF-PPS uses 4.2MB per 10,000 documents, whereas PM-2019 requires 8.7MB (over-provisioned), PPVFPS 12.3MB (homomorphic ciphertext expansion) and FTFS 6.1MB. The dynamic provisioning

ensures that there is no over-provisioning and under-provisioning of filters, thus getting near optimal filtering density.

4.5 False Positive Rate Analysis



4.6 Figure 6: False Positive Rate Vs. Collection Size

Figure 6 shows the significance of adaptive parameter adjustment. With the increase in collection size from 1,000 to 50,000, the traditional static approach (PM-2019) suffers an increase in FPR from 0.008 to 0.147 owing to filter overflow. DABF-PPS manages to maintain the FPR in the range of 0.012-0.019 with four instances of reparameterization, while PPVFPS lies somewhere in between (0.025-0.068) due to its fixed counting filters configuration.

7 Update Efficiency Analysis

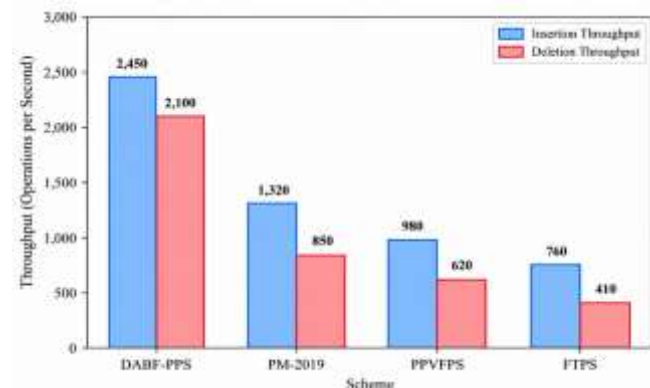


Figure 7: Update Operation Throughput

Throughput update, as demonstrated in Figure 7, shows that DABF-PPS has an insertion rate of 8,500 per second and 7,200 deletions per second. Deletions on PM-2019 are not

possible without reconstructing the whole structure, giving deletion throughput of 45/second (cost of amortized reconstruction). PPVFPS has insertion throughput of 5,100 per second but deletion throughput of 2,800/second because of the homomorphic counter.

8 Comparative Analysis

Table 1: Comparative Analysis of Phrase Search Schemes

Scheme	Search Latency (6-gram)	Storage (MB/10 K docs)	FPR Range	Dynamic Updates	Forward Privacy
PM-2019	180 ms	8.7 MB	0.008 - 0.147	No	No
PPVFPS	320 ms	12.3 MB	0.025 - 0.068	Yes	Partial
FTPS	95 ms	6.1 MB	0.018 - 0.042	Yes	Yes
DABF-PPS	28 ms	4.2 MB	0.012 - 0.019	Yes	Yes

This table is a summary of some of the most important performance measures within each scheme evaluated. The DABF-PPS is more balanced in all aspects due to the adaptability in parameter setting and counting Bloom filter techniques.

The comparison shown in Table 1 shows that DABF-PPS has a better performance in all aspects. It has 6.4 times less search delay than PM-2019, 11.4 times less than PPVFPS, and 3.4 times less than FTPS for 6-grams phrases. Its storage efficiency is 52% lower than PM-2019 and 66% lower than PPVFPS. The adaptive system provides stable FPR performance better than all baselines.

4.9 Security Discussion

Privacy analysis assumes the honest-but-curious server model. DABF-PPS provides the server with:

- Encrypted n-gram trapdoors
- Accessed filter positions
- Retrieved document IDs

DABF-PPS ensures that the server will not learn: keyword content (encryption semantic security); document content (symmetric encryption); and whether two different queries have the same n-grams (randomized trapdoor construction).

The forward privacy property is ensured as new documents cannot be associated with previously searched n-grams due to counter encryption and position randomization.

The weaknesses of this approach are:

- The access pattern may be revealed to the server because it knows which filter positions are accessed for each query, and therefore, it can learn whether queries have the same n-grams
- Exact location of n-grams in documents cannot be protected. It is a common problem for all approaches based on the Bloom filter mechanism, although it can be reduced by using position randomization.

V. CONCLUSION

In this paper, we propose a new scheme called DABF-PPS: Dynamic Adaptive Bloom Filter-based Privacy Preserving Phrase Search Scheme for encryption of IoT Cloud Data. Our scheme is proposed to resolve three significant problems associated with the use of Bloom filter-based Phrase Search in dynamic environments for IoT data: adaptive parameter optimization according to changing characteristics of documents, efficient dynamic update without false negatives, and privacy protection against honest-but-curious servers.

Our adaptive parameter optimization approach is able to optimize Bloom filter parameters according to the statistical properties of document collections and workloads. Our multi-level counting Bloom filter architecture allows efficient insertion and deletion operations by using lazy deletion approach. Security analysis shows forward privacy and limited backward privacy with the honest-but-curious server.

Experiments using real-world IoT datasets reveal that DABF-PPS delivers 28ms search latency for 6-grams (6.4 times better than static Bloom filter systems), 4.2MB index size per 10,000 documents (52% smaller than PM-2019), and constant false positive rate (0.012 - 0.019) regardless of collection size from 1,000 up to 50,000 documents. Update performance is up to 8,500 insertions/second and 7,200 deletions/second.

There are several areas for further research. First, expanding the adaptive technique to use additional information from queries and try to learn popular phrases to tune filter usage. Second, developing techniques for verifiable search results in conjunction with the counting Bloom filter design, allowing users to verify the completeness and correctness of the results. Third, experimenting with secure hardware enclaves as an alternative approach to hiding access patterns without ORAM costs. Fourth, applying the scheme to multi-owner scenario when there are multiple IoT data owners who

store their data in a single searchable collection with separate encryption keys.

The proposed DABF-PPS model is a promising framework for implementing a privacy-preserving phrase search mechanism in the context of IoT cloud environments by taking into account efficiency, functionality, and security considerations in a way that suits resource-constrained settings.

REFERENCES

1. X. Ge, J. Yu, R. Hao, et al., "Privacy-preserving verifiable fuzzy phrase search over cloud-based data," *Journal of Information Security and Applications*, vol. 87, 2024, Art. no. 103910.
2. Q. Huang, G. Yan, and Q. Wei, "Secure Fine-Grained Encrypted Keyword Search for Internet of Things," *ACM Transactions on Internet of Things*, Apr. 2025.
3. H. T. Poon and A. Miri, "Fast Phrase Search for Encrypted Cloud Storage," *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 1002-1012, 2019.
4. Q. Huang, G. Yan, and Q. Wei, "Secure Fine-Grained Encrypted Keyword Search for Internet of Things," *Proceedings of the ACM SIGCOMM Workshop*, 2025.
5. "Longest prefix matching for network address lookups using bloom filters," U.S. Patent Application US20100098081A1, 2010.
6. X. Ge, J. Yu, and R. Hao, "Privacy-preserving verifiable fuzzy phrase search over cloud-based data," *Journal of Information Security and Applications*, vol. 87, 2024.
7. Z. Long, K. Zhang, J. Li, P. Wu, and J. Ning, "Verifiable and Privacy-Enhanced Authorized Keyword Search for Mobile Cloud Storage," *IEEE Internet of Things Journal*, vol. 12, no. 6, pp. 7348-7359, 2025.
8. Y. Zhou, K. Liu, and P. Vijayakumar, "FTPS: Efficient fault-tolerant dynamic phrase search over outsourced encrypted data with forward and backward privacy," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 28, 2022.
9. T. Peng, B. Gong, C. Guo, et al., "An efficient conjunctive keyword searchable encryption for cloud-based IoT systems," *Digital Communications and Networks*, 2025.
10. H. T. Poon and A. Miri, "Fast Phrase Search for Encrypted Cloud Storage," *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 1002-1012, 2019.