Indonesian Journal of Electrical Engineering and Computer Science
Vol. 39, No. 3, September 2025, pp. 1661~1672
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v39.i3.pp1661-1672

Enhancing privacy in document-oriented databases using searchable encryption and fully homomorphic encryption

Abdelilah Belhaj[1], Soumia Ziti[1], Souad Najoua Lagmiri[2], Karim El Bouchti[3]
[1]Intelligent Processing Systems and Security (IPPS) Team, Faculty of Sciences, Mohammed V University in Rabat, Rabat, Morocco
[2]IRSM, Higher Institute of Management, Administration and Computer Engineering in Rabat, Rabat, Morocco
[3]Laboratory of Computer Systems Engineering (LISI), Faculty of Science Semlalia, Cadi Ayyad University, Marrakech, Morocco

Article Info

Article history:

Received Oct 2, 2024
Revised Apr 20, 2025
Accepted Jul 2, 2025

Keywords:

Encryption
Fully homomorphic
NoSQL database
Oriented document
Searchable encryption

ABSTRACT

In cloud-based not only SQL (NoSQL) databases, maintaining data privacy and the integrity are critically challenged by the risks of unauthorized external access and potential threats from malicious insiders. This paper presents a proxy-based solution that provides privacy-preserving by combining searchable encryption and brakerski-fan-vercauteren (BFV) fully homomorphic encryption (FHE) to facilitate secure search and aggregate query execution on encrypted data. Through extensive performance evaluations and security analyses, we show that our approach offers a robust solution for privacy-preserving data operations, with performance overhead introduced by the use of FHE. This solution gives an opportunity for a robust framework for secure data management and querying in NoSQL databases, with promising implications for practical deployment and future research. This work represents a significant advancement in the secure handling of data in NoSQL oriented databases, supplying a practical solution for privacy-conscious organizations.

Corresponding Author:

Abdelilah Belhaj
Intelligent Processing Systems and Security (IPPS) Team, Faculty of Sciences
Mohammed V University in Rabat
Morocco
Email: abdelilah_belhaj@um5.ac.ma

1. INTRODUCTION

Outsourcing the storage and the processing of private data to third-party cloud service providers have become a common decision for many organizations because of its cost-efficiency, scalability, and convenience. However, this practice makes applications vulnerable to unauthorized access by an external or by malicious insiders. It introduces significant security and privacy risks that must be carefully considered. not only SQL (NoSQL) databases [1], widely used for their flexibility in managing unstructured data, present specific challenges in maintaining privacy. These databases are often used to store sensitive information, such as medical records, financial data, user authentication data, and confidential documents. However, ensuring data privacy while enabling efficient search and aggregate queries presents a significant challenge. Encrypting data before sending it to the cloud database is a well-established strategy to protect privacy. Nevertheless, this approach introduces new challenges such as performing search queries and aggregate queries on encrypted data. In addition, balancing the trade-offs between security, efficiency, and performance are remarked to be complex in order to provide a practical solution for secure data management in cloud environments. Given that encrypted data cannot easily be queried, several schemes have been proposed to address such challenges. The researchers [2]-[5] focus on the security issues and challenges associated with cloud storage, particularly addressing compromised accounts, insider threats, and loss of control over data.

In relational databases, various models have been developed to achieve transparent data encryption (TDE). These models provide solutions for securing data in use through secure query processing on encrypted data, such as secure query database SDB [6], CryptDB [7], and MONOMI [8]. It is denoted that CryptDB is a query processing system designed for encrypted SQL databases. It uses a homomorphic encryption scheme that allows it to compute the sum of plaintext values by performing multiplication on ciphertexts. The authors of [9] introduce a novel approach to safeguarding encryption keys within a DBMS by utilizing a master key that is generated through encryption of a table or column. Xu *et al.* [10] propose cryptMDB, a practical encrypted MongoDB being a document-oriented database widely used. This scheme utilizes an additive homomorphic asymmetric cryptosystem to encrypt user data. It cannot support complex aggregate queries. Moreover, the authors of [11] propose a secure document database (SDDB) for document- oriented databases. The scheme is inspired by PIRATTE [12] and CryptDB concepts.

To address these limitations, the present work proposes a model ensuring privacy preserving using searchable encryption scheme to enable secure search and the Brakerski/Fan-Vercauteren (BFV) fully homomorphic encryption (FHE) scheme [13]-[18] to execute aggregate queries in document-oriented databases. Our contribution is to develop and implement a server proxy designed to ensure privacy-preserving search and aggregation in document-oriented databases. In the following sections, we will expose in detail the architecture, the implementation of the system. Then, we discuss its practical implications, challenges, and possible direction for future work.

2. PROPOSAL MODEL

2.1. System architecture

A secure NoSQL database as a service (DBaaS) in a public cloud infrastructure involves robust security, in particular, where the data owner cannot fully trust the cloud service provider (CSP). It is needed to ensure both the confidentiality and integrity of data in a NoSQL DBaaS within a public cloud infrastructure. We propose a secure proxy server that handles all security-related tasks and the cloud server remains unaffected. We use a searchable encryption scheme that allows users to search over encrypted data without decrypting it. We adopt a FHE to perform aggregations over encrypted data. The architecture of the system is illustrated in Figure 1. The proposal model mainly contains three parts: client, proxy server and server provider (SP) where the clients can only interact with NoSQL database via proxy. The proxy is responsible for protecting data privacy by applying the cryptographic modules. The proxy stores field keys for each sensitive field in its key store. Firstly, data submitted by users will be encrypted using encryption tools and then saved in database. To query the encrypted database client sends a request to the proxy server. The proxy rewrites the query that involves sensitive fields and submits the rewritten query to the SP. Finally, the proxy decrypts the encrypted results and send the plaintext result to client.

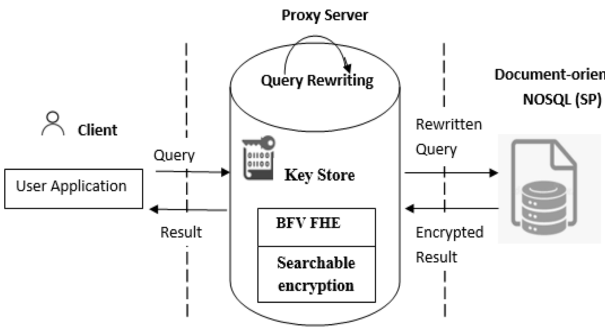


Figure 1. System architecture

We distinguish two types of sensitive fields search fields and aggregation fields. Search fields are data fields that we need to query or search against denoted by search field. However, aggregation fields refer to the data fields on which aggregation operations are performed including multiplication, addiction, sums, averages, counts denoted by Agg_fields. Our approach involves generating and managing separate keys for each sensitive field in a collection, so that each sensitive data is encrypted and decrypted with its own key. Data can be classified based on its sensitivity degree which refers to how critical and confidential the data are. Depending on the sensitivity of the field, different security measures and encryption techniques can be used including key size. Proxy server stores all data encryption in crypto_vault such that:

```
Crypto_vault {
'Collection_i': {
  'sensitiveField1": {
            'keyField'  : 'key1'
            'class': ' confidential '
             }
'sensitiveField2": {
            'keyField'  : 'key2'
            'classe' : 'secret'  }
'sensitiveField3": {
              'Sceme: 'Fully Homomorphic Encryption'
             'classe' : 'aggregate field '   }
……………….
'sensitiveField_n": {
            'keyField'  : 'key_n'
             'class': highly confidential' }
Sensitive_agg_Fields: 'FHE fully homomorphic encryption BFV scheme'}
```

2.2. Proposal algorithms

2.2.1. Searchable encryption scheme used for sensitive searchable field

Advanced encryption standard (AES) is a highly trusted encryption algorithm that ensures secure data transmission and storage. Two methods are suggested for implementing searchable encryption for sensitive serach_fields including random encryption and deterministic encryption. We use AES encryption scheme which can be configured to be either deterministic or random, depending on the mode and the use of initialization vectors IVs [19]-[21]. We provide a detailed explanation of the methods for utilizing both types of AES encryption.

a) Randomized encryption

This model of encryption ensures that the same plaintext will not produce the same ciphertext even though encrypted with the same key. It encrypts each plaintext using random initialization vectors IVs. However, the use of a random initialization vector makes searching encrypted field difficult. In this way, we need to rely on approaches that enable us searching on encrypted data while maintaining its confidentiality such as Blind Indexing. Blind index is used to stores the hash value corresponding to plaintexts by using HMAC algorithm [22]-[24]. Alternatively, we can use other cryptographic algorithms for generating a blind index such as SHA3-512 or key derivation function (KDF) like Argon2 or PBKDF. Besides that, there are certain techniques including order-preserving encryption (OPE). However, the later introduces some challenges and considerations associated with its application for textual data. Furthermore, it is crucial to store the random initialization vectors IV alongside the ciphertext in the document in the collection so that it can be used for decryption. Given a document which contain two sensitive fields sensitiveField1 et sensitiveField2. We would like to insert the document in the collection of document-oriented NoSQL as follow:

```
          document = {
               'sensitiveField1': 'secretValue1', '
                'sensitiveField2': 'secretvalue2',
           '    ……
             Other_fields  :'data'
               }
DB.myCollection .insert_one (document)
```

At first, proxy server intercepts and parses the incoming client requests. It identifies sensitive fields and determines the key used to encrypt each sensitive field from Crypto_Vault mentioned above. It encrypts the plaintext using AES scheme with random initialization vector IV and adds the vector IV in the document. At the time of insertion of document, proxy server calculates hash value corresponding of the plaintext and store the hash value along with the ciphertext in order to optimize search performance. Below is Algorithm 1, which outlines the steps for inserting a document containing searchable sensitive fields using randomized encryption.

Algorithm 1. Insertion of the document containing searchable sensitive fields using randomized encryption

```
Input :   document, sensitiveFields [ ]
for each  field in document  do:
        if      field  in sensitiveFields do :
                Let key = crypto_vault [myCollection][field]
                Generate random vector iv=random()
                 cipherText = Encrypt (document[field],algorithms.AES(key), modes.CBC(iv))
                document[field]=cipherText
                document[field_iv] = iv
                create the blind index
                       hash= HMAC(document[field],blind_key)
               document[field_blind_index'] = hash
               DB. myCollection .insert_one(document)
        end.
```

The server proxy encrypts each sensitive field and stores its IV and the blind index in separate fields. Blind index helps the proxy to find the document, while the vector IV is needed to decrypt the ciphertext once a sensitive field is found. However, this method has a solution for searchable encryption. However, the extra data is added and stored alongside with each sensitive field, which compromises the storage and the performance.

b) Deterministic encryption

A deterministic encryption scheme is a cryptosystem that consistently generates the same ciphertext for a given plaintext and key. This property makes it useful for practical applications, particularly in searchable encryption and ensuring data integrity. It allows for efficient searching and indexing of encrypted data. It enables databases to perform equality checks and joins directly on encrypted data. Furthermore, only the encryption key needs to be managed, avoiding the need to handle or store initialization vectors IVs. Nevertheless, it is highly vulnerable from a data protection perspective. It poses several security risks, particularly in terms of pattern exposure.

To overcome this problem, we use a field encryption key in a dcument denoted by FDK which will be exploited to encrypt and decrypt the sensitive field in each document. Algorithm 2 shows how to insert document using deterministic algorithm based on field document encryption. FDK can be generated from combining the field encryption key stored in crypto_vault with a document's unique identifier of the document manually generated. This could be done according to the following relationship:

$$FDK = Hash(key_{field} \oplus \_id\_Document)$$

The following is Algorithm 2, which outlines the steps for inserting a document containing searchable sensitive fields using deterministic encryption with a field encryption key in each dcument FDK.

Algorithm 2. Insertion of the document containing searchable sensitive fields using deterministic encryption

```
:Input:   document, sensitiveFields [ ]
Generate _id document
for  each  field in document do:
        if      field  in sensitiveFields do :
               Let key = crypto_vault [myCollection][field]
                Combine the key and the document _id   to derive FDK
                          FDK=hash (key_field ⊕ _id)
                cipherText     =     Encrypt     (document[field],     algorithms.AES(FDK),
deterministic_mode)
                document[field]=cipherText
                creates the blind index
                        hash= HMAC (document[field], blind_key)
               document[field_blind_index'] = hash
               DB. myCollection. insert_one(document)
 end.
```

2.2.2. Fully homomorphic encryption

Aggregation of sensitive fields in encrypted form involves performing operations such as summation, averaging, or other statistical computations. FHE allows mathematical operations to be performed on encrypted data without the requirement for decryption. This trait makes it useful for data security and confidentiality, particularly in cloud computing. In this work, we adopt a brakerski-fan-vercauteren (BFV) [25], [26] scheme which has several advantages including its simplicity and its compatibility. It is designed based on the ring-learning with errors (RLWE) problem.

In BFV, The plaintext and the ciphertext spaces are defined over two distinct polynomial rings denoted by $P = R_t = Z_t[x]/(x^n + 1)$ and $C = R_q \times R_q$, where $R_q = Z_q[x]/(x^n + 1)$, $n \in Z$ is the ring dimension and $t \in Z$ and), $q \in Z$ are the plaintext and ciphertext coefficients, respectively. The insertion of documents containing aggregate fields follows a structured process. Formally, an FHE has four components which are key generation denoted by generate_keys, encryption, decryption and evaluation. The computational complexity of all operations of the encryption scheme must be polynomial in order to make the scheme efficiently implementable for practical use. Here, we provide a concise description of each operation of the scheme:

- generate_keys: generates public key pk, secret key sk and evaluate_key derived from sk.
- Encryption (message m\in P): this operation produces a cipher such that c = Encrypt(m, pk) and adds some noise to ensure security.
- Decryption (ciphertext $c \in C$): defines the opposite process of Encrypt, removes the noise to recover the original plaintext denoted by = decrypt(m,sk)
- Evaluation: this process enables efficient and secure homomorphic computation, particularly multiplication. It is crucial to reduce the noise and the size of the ciphertext.

Presented is Algorithm 3, which describes the steps for encryption using BFV Fully Homomorphic Encryption:

Algorithm 3. BFV encryption

Input m , pk(a,b) where a, b are polynomials in appropriate ring
Output $c = (c1, c2)$ where $c1, c2$ are two encrypted polynomials
Create a polynomial in polynomial ring from message (m)
Generate a random polynomial r in the ring of polynomials.
Calculate $c1$:
$c1 = (a.r) \, mod \; q$
Calculate c2: $c2 = (b.r + m) \, mod \; q$

Return $c = (c1, c2)$

Below is Algorithm 4, which describes the process of BFV decryption:

Algorithm 4. BFV decryption

Input $c = (c1, c2)$, secret key sk
Output m the plaintext message
Calculate m:
$m = (c2 - c1 \, . \, s) \, mod \, q$
Convert the polynomial m into original message
Return m

3. METHOD

3.1. Implementation

In this section, we describe in detail the architecture of the proposed model which ensures privacy-preserving by using searchable encryption scheme and BFV homomorphism encryption on NoSQL document databases. The architecture of secure model comprises three main components including client requests, proxy and cloud server provider CSP which provide NoSQL document-based databases. The main logic lies in the proxy server which manages and stores all field keys and blind keys used for creating blind indexes. In addition, it rewrites and transforms the query to match the encrypted indexes stored alongside with the sensitive field. It configures all parameters of a FHE BFV scheme. We adopt deterministic encryption according to the algorithm mentioned above which relies on FDK field document Key for each sensitive field which is derived from the key field and _Id document.

To implement FHE with the BFV scheme in Python, libraries such as TenSEAL and PySEAL can be utilized. They offer Python bindings for working with homomorphic encryption. In this work, we choose the TenSEAL library [27] to achieve practical implementation of the BFV scheme. TenSEAL is designed to be user-friendly, and simplifies many complex aspects of FHE. It is an open-source Python library allowing for easy and efficient computation on encrypted data. We need to configure some main parameters including polynomial modulus degree (poly_modulus_degree), coefficient modulus (coeff_modulus), plaintext Modulus (plain_modulus). It is crucial for maintaining a balance between security, performance, and functionality in order to make the encryption scheme efficient, and suitable for specific applications.

The BFV scheme is used for aggregate queries such as addition, multiplication, sum and average. However, equality-based queries are performed by searchable encryption scheme and AES encryption scheme. The choice of AES key size should align with the degree of sensitivity of the field such as class 1 (low sensitivity fields AES-128), class 2 (medium sensitivity fields AES-192) and class 3 (high sensitivity fields AES-256). It depends on several factors, such as the desired security level and performance.

We consider the following document that contains sensitive fields, namely social security number SSN, credit_card_number and balance. The latter as an aggregate field is encrypted using BFV scheme. Figure 2 illustrates the document after encryption.

Document without encryption: {
"name":"abdel",
"email":"abdel15@gmail.com",
"ssn":"15-154-1895",
"credit_card_number":"145120241524",
"balance":1526
}

Figure 2 illustrates the document after the insertion and encryption of sensitive fields using BFV for encrypting sensitive aggregate fields and the searchable encryption algorithm for encrypting sensitive searchable fields.


```
_id: ObjectId('66bd0c03bcff804a809b0dc6')
name : "abdel"
email : "abdel15@gmail.com"
ssn_blind_index : "3baf3f3bd030bcfc2b2dcad6b425c15adeaa22a6520dd31e34a10a5ffed53f0c"
ssn : "rBW713ktQ9b++J74VHb/9A=="
credit_card_number_blind_index : "344f29b7a3e86e7857e4e2629834fd7cfa86209a6e4bbec5ebe96a9ee51683a5"
credit_card_number : "1b9XjI8nS4GVR2SYLUgqtA=="
balance : Binary.createFromBase64('CgEBEpa0FF6hEAQBAgAAFhoFAAAAAAAotS/9oGEABgAcXA6O/Z/cci0QoLxmUw1ktPXI…
```


Figure 2. Encrypted document visualized by MongoDB compass

3.2. Comparison in FHE

FHE, such as the BFV scheme, supports arithmetic operations like addition and multiplication but does not support comparison operations. Comparison operations, such as determining whether $a > b$ is quite challenging because they are non-linear. However, comparison operators can be integrated into various document database aggregation pipelines to filter, compute, or project data based on conditions. There are several approaches to perform comparison on encrypted data but they are computationally expensive and complex. Subtraction and checking the sign after decryption is the traditional approach to enable comparison on FHE. This approach decrypts the result of the homomorphic subtraction. If the difference is positive, $a > b$; if negative, $a < b$; and if zero, $a == b$.

There is another approach that involves combining FHE with OPE which can leverage the strengths of both schemes. On one hand, it enables secure computations on encrypted data, and on the other hand it maintains order properties for efficient comparison and querying. For each sensitive aggregate field, we associate two ciphertext values, one encrypted using FHE and another encrypted using OPE. In this way, we can perform operations on the FHE encrypted value and use the OPE encrypted value for efficient comparisons.

document = {
‘FHE_balance’: “fhe_encrypted_value”,
‘OPE_balance’: ‘ope_encrypted_value’, ………}

By parsing and analyzing queries, we can automatically identify which field is involved in operations like sum, avg, min, max, greater than, less than. Thus, we can decide whether to use the fhe_encrypted_value or ope_encrypted_value during query execution. However, although OPE maintains the order of the plaintexts in the ciphertexts; it is vulnerable to inference attacks. It leaks information about the relative ranking of the data. For example, if the attacker knows that $Enc(a) < Enc(b)$, he can infer that $a < b$.

3.3. Integrity verification algorithms

Verifying data integrity is crucial for maintaining the accuracy, consistency, and credibility of information. The proposed integrity verification, combines hashing to ensure data integrity and an XOR operation to detect changes. The following is Algorithm 5, which outlines the steps for inserting a document with integrity verification:

Algorithm 5. To insert document with integrity verification

```
Input Document
Output
xorResult = 0 // Convert data fields to a byte representation and perform XOR
 FOR each field IN data
         xorResult = xorResult XOR ConvertToByte(field)
 END FOR
hmac_result = HMAC_SHA256(SECRET_KEY, xor_result).
Document.AddField (integrity_HMAC)= hmac_result
Db.mycollection.insert(Document)
```

Algorithm 6 is provided below, demonstrating the procedure for verifying document integrity:

Algorithm 6 to verify document integrity

```
Input : Query to retrieve the document.
Output : A Boolean Value
xorResult = 0
 FOR each field IN data
         xorResult = xorResult XOR ConvertToByte(field)
 END FOR
hmac_result = HMAC_SHA256(SECRET_KEY, xor_result).
If document [ 'hash integrity] ≠ hmac_result
       Return False
Else return True
```

4. RESULTS AND DISCUSSION

The proposal algorithm has been tested by considering numbers of documents ranging from 1,000 to 5,000. The time taken by a query in a proxy system uses searchable encryption and FHE. It is crucial to consider all relevant components of the query process. Firstly, Database Response Time which is the time taken by the NoSQL database server to execute a query and return the results. It includes all the internal processing within the database, including searching, filtering, and aggregating data. Secondly, Encryption /Decryption Time which is the time taken by the proxy server to encrypt the sensitive fields using their corresponding scheme before sending them to the database and to decrypt the results received from the database. Besides, all the experimental procedures are performed on an Intel(R) Core (TM) i5-1035G1 CPU @ 1.00 GHz, 1190 MHz, 4 cores, 8 processors. Figure 3 illustrates the comparison of query times for searching a document in two scenarios: one where the database is unencrypted and the remaining one where it is encrypted using a searchable encryption scheme.

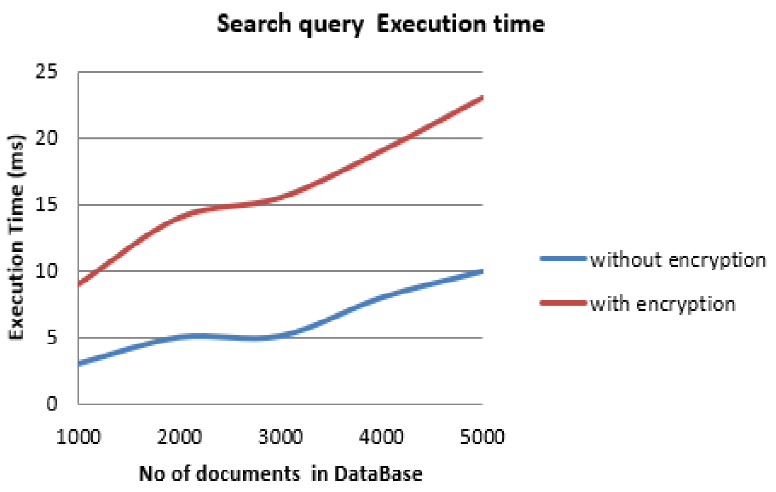


Figure 3. Search queries using searchable encryption

Figure 4. compares the time required for aggregate operations in two scenarios: one where the database is unencrypted and one where it is encrypted using a BFV homomorphic encryption scheme. It shows that the time required to perform aggregation queries on encrypted data is significantly longer than performing these operations on plaintext data.

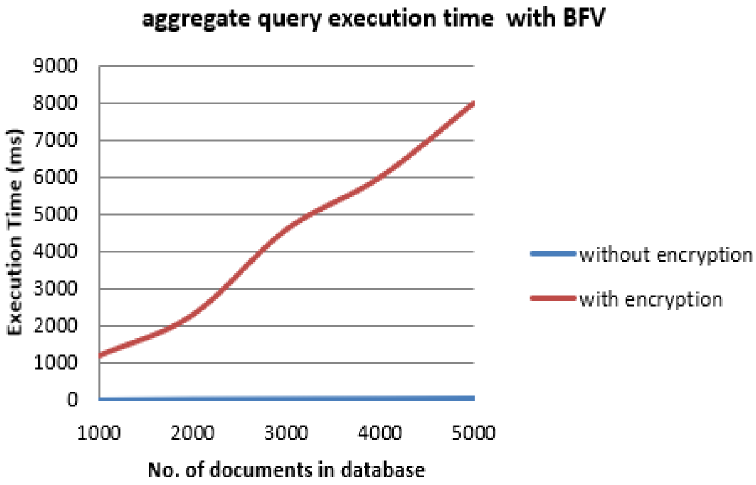
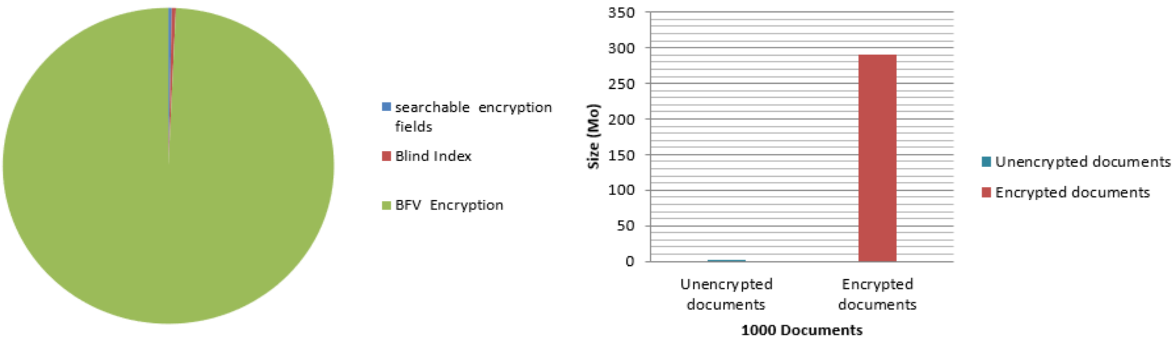


Figure 4. Aggregate query execution Time with BFV scheme

Figure 5 presents the storage impact of the BFV scheme. In part Figure 5(a), the storage overhead for a dataset with 1,000 documents is shown, highlighting the effect of aggregating documents on the required storage space. In part Figure 5(b), the storage overhead within a single document is detailed, illustrating the influence of additional elements such as metadata and encryption keys on the storage of an encrypted document. This figure sheds light on the challenges associated with increasing ciphertext size in encryption schemes, which can lead to additional storage and data management costs.



(a) (b)

Figure 5. Storage impact of the BFV (a) Storage overhead for a dataset with 1,000 documents and (b) Storage overhead inside a document

This study has introduced a novel secure proxy ensuring privacy-preserving that integrates a searchable encryption scheme with BFV FHE to enable secure search and aggregate queries in document-oriented databases. Our results demonstrate that the proposed approach effectively maintains data confidentiality while allowing for efficient query processing. Our approach based on the integration of searchable encryption with BFV provides robust privacy guarantees. The secure proxy ensures that both the query and the data remain encrypted. It significantly enhances privacy protection compared to existing solutions, which typically rely on either searchable encryption or partial homomorphic encryption alone. While the privacy benefits are important, the performance metrics indicate a trade-off between security and efficiency. Our implementation shows that the encryption and decryption processes introduce additional computational overhead. Specifically, the query execution time is significantly longer than non-encrypted queries execution time. Accelerating the BFV scheme involves a combination of choosing the right parameters, optimizing algorithms, using efficient libraries [28] and utilizing hardware accelerations [29]-[33].

As shown in Table 1, existing accelerators for the BFV scheme, including both hardware accelerators and software libraries that optimize its performance are listed with their key features.

Table 1. Existing BFV accelerators

Accelerator	Examples	Key features accelerated
GPU accelerators	NVIDIA GPUs, cuFHE	Polynomial multiplication, NTT, modular arithmetic
FPGA Accelerators	HEAX (Homomorphic encryption accelerator), custom FPGA designs	Custom pipelines for polynomial operations
ASIC accelerators	Microsoft F1, custom ASIC designs	Fully custom BFV operation acceleration (low latency)
Software libraries	Microsoft SEAL, PALISADE, HElib	Optimized polynomial arithmetic, multi-threading, batching
Cloud solutions	Microsoft Azure, IBM cloud	Offloading FHE operations to optimized cloud hardware

The use of accelerators for FHE BFV scheme is crucial for enhancing performance computational tasks, improving efficiency and ensuring that organizations can handle sensitive data securely and effectively. Figure 6 illustrates the significant reduction of execution after using BFV accelerators.

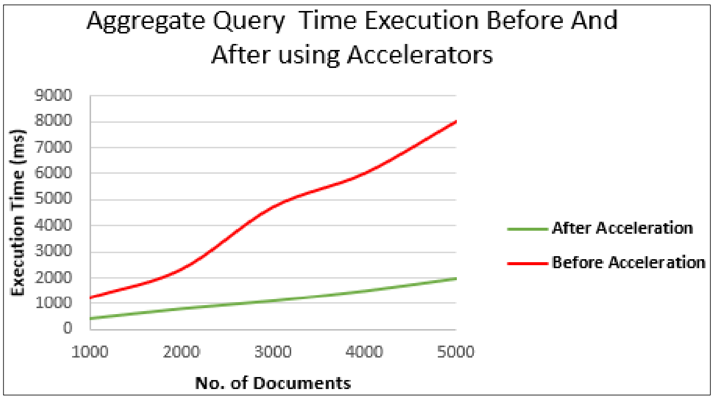


Figure 6. Aggregate query execution time with BFV scheme before and after using accelerators

Our results are consistent with and build upon the findings of earlier research on privacy-preserving query processing. For instance, similar work by SDDB and CryptMDB propose a practical encrypted MongoDB to achieve privacy-preserving utilizing an additive homomorphic encryption but did not address the aggregate query limitation. Conversely, our approach bridges this gap by integrating BFV FHE, which permits complex aggregate operations to be performed without decrypting the data. While the proposal model offers significant advancements, one limitation of our approach must be acknowledged which is the increased computational resource requirement. Additionally, its performance can be impacted by the size of the ciphertext and the complexity of the aggregate queries.

Future works could focus on optimizing the encryption schemes to reduce computational overhead. Furthermore, applying our model to different types of databases and query situations may offer additional insights into its flexibility and scalability.

5. CONCLUSION

The proposed proxy-based solution for document-oriented databases enhances preserving privacy and integrity in cloud NoSQL databases. Our system effectively has guarded against insider attacks, ensuring that sensitive information remains protected even in the presence of potential threats from within the organization. This hybrid encryption approach enables secure search and aggregate query execution in document-oriented databases using a combination of searchable encryption and BFV FHE. Future enhancements will focus on optimizing performance, improving scalability, by examining wider applications to further enhance and confirm the system effectiveness across various real-world situations.

ACKNOWLEDGMENTS

The authors thanks A. Abououard for scientific help.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Abdelilah Belhaj	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
Soumia Ziti		✓				✓		✓	✓	✓	✓	✓		
Souad Najoua Lagmiri	✓		✓	✓			✓			✓	✓		✓	✓
Karim El Bouchti					✓		✓			✓		✓		✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

INFORMED CONSENT

We have obtained informed consent from all individuals included in this study.

ETHICAL APPROVAL

All participants provided informed consent before participating in the study, and the study was conducted in accordance with the Declaration of Helsinki.

DATA AVAILABILITY

D ata availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] K. Eldahshan, A. Alhabshy, and G. Abutaleb, "A comparative study among the main categories of NoSQL databases," *Al-Azhar Bulletin of Science*, vol. 31, no. 2, pp. 51–60, Dec. 2020, doi: 10.21608/absb.2020.210374.

[2] A. Sharma, P. Jha, and S. Singh, "Data control in public cloud computing: issues and challenges," *Recent Advances in Computer Science and Communications*, vol. 14, no. 2, pp. 564–579, May 2019, doi: 10.2174/2213275912666190617164550.

[3] A. Purnima, Devendran, and Rajavarman, "Security issues and challenges in cloud computing," *Tuijin Jishu/Journal of Propulsion Technology*, vol. 44, no. 3, pp. 3173–3178, Oct. 2023, doi: 10.52783/tjjpt.v44.i3.1428.

[4] S. A. El-Seoud, H. F. El-Sofany, M. Abdelfattah, and R. Mohamed, "Big data and cloud computing: trends and challenges," *International Journal of Interactive Mobile Technologies*, vol. 11, no. 2, pp. 34–52, Apr. 2017, doi: 10.3991/ijim.v11i2.6561.

[5] P. Atzeni, F. Bugiotti, L. Cabibbo, and R. Torlone, "Data modeling in the NoSQL world," *Computer Standards and Interfaces*, vol. 67, p. 103149, Jan. 2020, doi: 10.1016/j.csi.2016.10.003.

[6] W. K. Wong, B. Kao, D. W. L. Cheung, R. Li, and S. M. Yiu, "Secure query processing with data interoperability in a cloud database environment," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Jun. 2014, pp. 1395–1406, doi: 10.1145/2588555.2588572.

[7] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in *SOSP'11 - Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, Oct. 2011, pp. 85–100, doi: 10.1145/2043556.2043566.

[8] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," *Proceedings of the VLDB Endowment*, vol. 6, no. 5, pp. 289–300, Mar. 2013, doi: 10.14778/2535573.2488336.

[9] K. El Bouchti, S. Ziti, F. Omary, and N. Kharmoum, "New solution implementation to protect encryption keys inside the database management system," *Advances in Science, Technology and Engineering Systems*, vol. 5, no. 2, pp. 87–94, 2020, doi: 10.25046/aj050211.

[10] G. Xu, Y. Ren, H. Li, D. Liu, Y. Dai, and K. Yang, "CryptMDB: a practical encrypted MongoDB over big data," in *IEEE International Conference on Communications*, May 2017, pp. 1–6, doi: 10.1109/ICC.2017.7997105.

[11] M. Almarwani, B. Konev, and A. Lisitsa, "Fine-grained access control for querying over encrypted document-oriented database," in *Communications in Computer and Information Science*, vol. 1221 CCIS, 2020, pp. 403–425.

[12] S. Jahid and N. Borisov, "PIRATTE: proxy-based immediate revocation of ATTribute-based encryption," *arXiv preprint*, 2012, [Online]. Available: http://arxiv.org/abs/1208.4877.
[13] Z. Zheng, K. Tian, and F. Liu, *Fully Homomorphic Encryption*. 2023.
[14] D. Zhao, "Advances and applications in fully homomorphic encryption research," *Applied and Computational Engineering*, vol. 135, no. 1, pp. 39–48, Feb. 2025, doi: 10.54254/2755-2721/2025.20959.
[15] D. P. Rajan, S. J. Alexis, and S. Gunasekaran, "Dynamic multi-keyword based search algorithm using modified based fully homomorphic encryption and Prim's algorithm," *Cluster Computing*, vol. 22, no. S5, pp. 11411–11424, Sep. 2019, doi: 10.1007/s10586-017-1399-x.
[16] N. Nurdin, "Comparative analysis between advanced encryption standard and fully homomorphic encryption algorithm to secure data in financial technology applications," *Jurnal CoreIT: Jurnal Hasil Penelitian Ilmu Komputer dan Teknologi Informasi*, vol. 10, no. 2, Dec. 2024, doi: 10.24014/coreit.v10i2.30809.
[17] J. Shen, Y. Zhao, S. Huang, and Y. Ren, "Secure and flexible privacy-preserving federated learning based on multi-key fully homomorphic encryption," *Electronics (Switzerland)*, vol. 13, no. 22, p. 4478, Nov. 2024, doi: 10.3390/electronics13224478.
[18] H. Okada, R. Player, and S. Pohmann, "Homomorphic polynomial evaluation using galois structure and applications to BFV bootstrapping," in *Lecture Notes in Computer Science*, vol. 14443 LNCS, 2023, pp. 69–100.
[19] B. Sarkar, A. Saha, D. Dutta, G. De Sarkar, and K. Karmakar, "A survey on the advanced encryption standard (AES): a pillar of modern cryptography," *International Journal of Computer Science and Mobile Computing*, vol. 13, no. 4, pp. 68–87, Apr. 2024, doi: 10.47760/ijcsmc.2024.v13i04.008.
[20] D. Limbachia, "Encryption of card details using AES with a 128-bit key a secure approach to data protection," *Interantional Journal of Scientific Research in Engineering and Management*, vol. 08, no. 03, pp. 1–5, Mar. 2024, doi: 10.55041/ijsrem29484.
[21] R. K. Kumar, M. H. Yogesh, K. R. Prasad, Sharankumar, and S. Sabareesh, "256-Bit AES encryption using SubBytes blocks optimisation," in *Lecture Notes in Electrical Engineering*, vol. 1106, 2024, pp. 621–628.
[22] W. Uriawan, R. Ramadita, R. D. Putra, R. I. Siregar, and R. Addiva, "Authenticate and verification source files using SHA256 and HMAC algorithms," *Preprints.org*. Jul. 01, 2024, doi: 10.20944/preprints202407.0075.v1.
[23] M. Najjar, "d-{HMAC} — An improved {HMAC} algorithm," *International Journal of Computer Science and Information Security*, vol. 13, no. 4, pp. 89–96, 2015.
[24] P. Sravan and S. Verdandi, "Building a secure and high-performance HMAC processor based on SHA-3 for the cloud-based financial sector," in *Proceedings of the 2nd IEEE International Conference on Advances in Computing, Communication and Applied Informatics, ACCAI 2023*, May 2023, pp. 1–8, doi: 10.1109/ACCAI58221.2023.10200480.
[25] R. Geelen and F. Vercauteren, "Bootstrapping for BGV and BFV revisited," *Journal of Cryptology*, vol. 36, no. 2, p. 12, Apr. 2023, doi: 10.1007/s00145-023-09454-6.
[26] V. H. Chandana, "Design and implementation of encryption/ decryption architectures for BFV homomorphic encryption scheme," *International Journal for Research in Applied Science and Engineering Technology*, vol. 12, no. 7, pp. 1098–1102, Jul. 2024, doi: 10.22214/ijraset.2024.63724.
[27] Y. B. Wiryen, N. W. A. Vigny, M. N. Joseph, and F. L. Aimé, "A Comparative study of BFV and CKKs schemes to secure IoT data using TenSeal and Pyfhel homomorphic encryption libraries," *International Journal of Smart Security Technologies*, vol. 10, no. 1, pp. 1–17, Nov. 2023, doi: 10.4018/ijsst.333852.
[28] A. Tsuji and M. Oguchi, "Comparison of FHE schemes and libraries for efficient cryptographic processing," in *2024 International Conference on Computing, Networking and Communications, ICNC 2024*, Feb. 2024, pp. 584–590, doi: 10.1109/ICNC59896.2024.10556382.
[29] Y. Yang, H. Lu, and X. Li, "Poseidon-NDP: practical fully homomorphic encryption accelerator based on near data processing architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 12, pp. 4749–4762, Dec. 2023, doi: 10.1109/TCAD.2023.3292211.
[30] Y. Su, B. Yang, C. Yang, and L. Tian, "FPGA-based hardware accelerator for leveled RIng-LWE fully homomorphic encryption," *IEEE Access*, vol. 8, pp. 168008–168025, 2020, doi: 10.1109/ACCESS.2020.3023255.
[31] S. Hashim and M. Benaissa, "Integer based fully homomorphic DSP accelerator using weighted-number theoretic transform," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 30, no. 3, pp. 362–371, May 2023, doi: 10.37934/araset.30.3.362371.
[32] J. Zhang, X. Cheng, L. Yang, J. Hu, X. Liu, and K. Chen, "SoK: fully homomorphic encryption accelerators," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–32, Dec. 2024, doi: 10.1145/3676955.
[33] M. Zheng, L. Ju, and L. Jiang, "CoFHE: Software and hardware Co-design for FHE-based machine learning as a service," *Frontiers in Electronics*, vol. 3, Jan. 2023, doi: 10.3389/felec.2022.1091369.

BIOGRAPHIES OF AUTHORS

Abdelilah Belhaj In 2013, he earned his Master's degree in applied informatics and Offshoring from Faculty of Sciences, Mohammed V University in Rabat, Morocco. Currently, he is pursuing his Ph.D. studies in the same faculty and works as a computer science teacher in a high school in Temara, Morocco. His research interests lie in cryptography, big data, and neural networks. He can be contacted at email: abdelilah_belhaj@um5.ac.ma.

Soumia Ziti is a full professor and researcher at the Faculty of Sciences of Mohammed V University in Rabat since 2007. She obtained her Ph.D. in computer science specializing in graph theory from the University of Orleans in France, along with a diploma in advanced studies in fundamental computer science. She also holds a Baccalaureate in Mathematical Sciences and completed her undergraduate studies in mathematics and physics, specializing in mathematics, at Hassan II University in Morocco. Furthermore, she earned a master's degree in science and technology in computer science from the same institution. Her research interests encompass a wide range of topics including graph theory, information systems, artificial intelligence, data science, software development, database modeling, big data, cryptography, and numerical methods and simulations. She can be contacted at email: s.ziti@um5r.ac.ma.

Souad Najoua Lagmiri She received her diploma for doctor degree from the Mohammadia School of Engineers Rabat. Currently Director of the Higher Institute of Management, Administration and Computer Engineering Rabat Morocco. She has published several scientific articles in international journals and conferences. His field of research focuses on encryption algorithms, cryptography, and information security based on complex mathematical equations. She can be contacted at email: snajoua.lagmiri@gmail.com.

Karim El Bouchti In 2020, he obtained a Doctor of Computer Science degree from the Faculty of Sciences of Mohammed V University in Rabat Morocco. Since 2023, he has been working as an associate professor at the Faculty of Sciences Semlalia, University Cadi Ayyad Marrakech, Morocco. With 10 years of professional experience in CNESTEN, the National Center for Energy Sciences and Nuclear Technology, his field of research focuses on cyber security, big data analytics. He can be contacted at email: k.elbouchti@uca.ac.ma.