IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 April 2024, accepted 27 May 2024, date of publication 3 June 2024, date of current version 10 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3408221

RESEARCH ARTICLE

PPAC-CDW: A Privacy-Preserving Access Control Scheme With Fast OLAP Query and Efficient Revocation for Cloud Data Warehouse

SOMCHART FUGKEAW, (Member, IEEE), AND LYHOUR HAK
School of ICT, Sirindhorn International Institute of Technology, Thammasat University, Khlong Nueng, Pathum Thani 12000, Thailand

Corresponding author: Somchart Fugkeaw (somchart@siit.tu.ac.th)

This work was supported by the Office of the Permanent Secretary, Ministry of Higher Education, Science, Research and Innovation (OPS MHESI), Thailand Science Research and Innovation (TSRI), and Thammasat University under Contract RGNS 65-110.

ABSTRACT Achieving privacy-preserving analytical query with fine-grained access control for cloud-based data warehouse (CDW) through the use of online analytical processing (OLAP) tool is a real challenge. This is because the access control must be enforced differently to multiple users while the OLAP query should be excelled from the encrypted DW and the query results are delivered through the public network. Existing solutions employ encryption solutions to apply on DW. However, they mostly overlooked fine-grained access control enforcement to different users and efficient OLAP query performance when there is a large number of users. In this paper, we proposed a PPAC-CDW scheme, a fine-grained and privacy-preserving access control with efficient query processing for OLAP queries for CDW. Our proposed scheme is based on the integration of ciphertext-policy attribute-based encryption, an extended model of materialized view scheme of MOLAP, and the hybrid cloud system. Our proposed scheme enjoys fast query performance based on encrypted pre-computed cube and our proposed B+Tree model. In addition, we introduced an efficient and traceable user revocation mechanism based on proxy re-encryption and blockchain with optimized cost of ciphertext retrieval. Finally, we conducted experiments demonstrating that our scheme offers more efficient data access performance compared to related works.

INDEX TERMS Access control, B+Tree, cloud computing, CP-ABE, data warehouse, revocation.

I. INTRODUCTION

Data warehouse (DW) is a system used to support data analysis and business intelligence applications. To formulate the data warehouse system, several data sources are pulled to an ETL tool used to extract, transform, and load the unified data onto the data warehouse. Generally, the volume of data to be stored in the DW is huge. Hence, OLAP tool is introduced to leverage the data from DW and big data for supporting the decision making. It is usually employed to model the DW schema. There are three major OLAP models including multidimensional OLAP (MOLAP), relational OLAP (ROLAP), and Hybrid OLAP (HOLAP). In MOLAP system, multidimensional cubes are constructed from pre-computation of all possible views called materialized views. ROLAP is based on the star schema which is a kind of relational database schema in which data are stored in the fact table and several dimension tables. HOLAP is a hybrid scheme of MOLAP and ROLAP. Implementing an enterprise data warehouse is non-trivial. Adequate computational resources, including powerful servers, storage, and network infrastructure, are necessary to accommodate extensive amounts of data and handle complex analytical queries effectively. Due to the emerging technology of cloud computing providing ubiquitous, resilient, and on-demand service, many enterprises tend to use the cloud service as a major platform for their system deployment. Here, data warehouse and big data can be implemented on virtualization machines and cloud storage where the users can connect to consume the services flexibly through any endpoint systems such as computer and mobile devices. While the cloud offers numerous advantages, the primary concern for most

The associate editor coordinating the review of this manuscript and approving it for publication was Siddharth Tallur.

enterprises when considering its adoption is the management of privacy and security. In particular, the data stored in a data warehouse or OLAP system typically consists of cleansed and strategic information that is considered sensitive and highly valuable. As a result, any instances of data leakage or compromise within these systems are considered critical.

Generally, cloud service providers (CSPs) provide basic security systems such as strong authentication and encryption to support the secure access control and data privacy of the system or data located in their platforms. However, CSPs can be considered honest but curious. If the encryption key is not fully managed by the tenants, the privacy of data cannot be fully guaranteed. In addition, the overheads of both computation and key management to support user queries over the encrypted data are big problems. Traditional encryptions such as symmetric encryption and public key encryption are not efficient to be directly applied for outsourced data and queries in the cloud environment. This is because the key distribution cost of symmetric encryption is not impractical for multiple cloud users while the public key encryption needs multiple copies of the ciphertexts for each user.

Basically, data warehouse users interact with its based on the analytical query made over the data warehouse by using the OLAP tool. Here, the query over the encrypted data techniques has gained more momentum and thus has been introduced by a few works [1], [2], [3] to achieve the privacy preserving of database privacy and query.

In addition to the query processing over the encrypted data approach, order preserving encryption (OPE) [4] and homomorphic encryption [5] are applied by some works [6], [7] to support the privacy of query processing in cloud data warehouse and big data [8], [9]. The OPE is an encryption scheme whose encryption function preserves the numerical ordering of the plaintexts. In homomorphic encryption, it allows computation such as mathematical operations such as addition and multiplication to be performed directly on encrypted data.

However, dealing with the query over encrypted database, OPE and homomorphic encryption share common problems related to computation complexity and their performance issue. Also, the segregation of user rights to make the different access rights cannot be fully done by such techniques. For example, Alice and Bob can only make a query over the encrypted DW for viewing the sales performance that belongs to his/her responsible store only. This can be only done by limiting the access in the database configuration or writing additional functions to check the query request done over the encrypted dimensional data and fact to respond to the query. Therefore, managing the privilege of users and providing encryption mechanism are required separately and the cost of such tasks is non-trivial especially when there are a high number of users using the cloud data warehouse (CDW).

To date, a ciphertext policy attribute-based encryption (CP-ABE) is considered an effective solution to support fine-grained and privacy-preserving access control for outsourced data sharing. Many cloud-based access control solutions [10], [11], [12], [13] use CP-ABE as their cryptographic construct. In CP-ABE, the data is encrypted with the access policies constructed from the set of attributes logically determined by the mathematical operators such as $<, >, =$ and gate operators AND, OR, MoN. To decrypt the ciphertext, the user needs to have a secret key consisting of a set of attributes that satisfy the access policy used to encrypt such ciphertext. Therefore, CP-ABE is a cryptographic-based access control mechanism that supports one-to-many encryption and fine-grainedness. However, CP-ABE is not suitable to directly support data warehouse security because of two major reasons due to its expensive pairing and exponentiation operations and lack of revocation support which is the essential security requirement for managing users in data warehouse setting where different user groups can join and leave the system. The cost of revocation generally includes ciphertext re-encryption and key update or key re-generation.

Recently, blockchain technology has been employed by many works [14], [15], [16] to support scalable and traceable data access control. Thanks to its nice characteristics related to decentralized network, tamper-resistance, and fault-tolerance, blockchain has been employed by many industries such as financial, healthcare, supply chain, transportation, etc. to enable their business transaction processing to be done in a more accessible, traceable, and secure manner. Considering the cloud-based access control for big data and data warehouse where the high number of users from several enterprise units or different domains can access or make a query deliberately, adopting blockchain to automate and control core access control functions and user management life cycle (register, authentication, revocation) as well as support data query is promising.

Existing data warehouse security solutions [1], [8], [17], [18], [19], [20] generally share the common shortfalls as follows. They generally focus on encrypting the dimension and/or fact data while the access control method is separately managed. They also ignore the issues related to transaction tractability and revocation. In addition, the performance of encrypted data or query mostly rely on the traditional search where the associated dimension and fact data are exhaustively search for the query.

Consequently, it is a real challenge to entail the privacy-preserving access control for OLAP query with fast query performance and efficient revocation for CDW. In this paper, we proposed a secure and fine-grained and privacy-preserving access control scheme for shared DW query results over the MOLAP system. Major building blocks of our proposed scheme consists of the core cryptographic protocols based on a symmetric encryption and CP-ABE, the proposed B+Tree for encrypted MV retrieval, and proxy re-encryption (PRE) and blockchain to support user revocation management. To the best of our knowledge, our work is the first attempt that extend the capability of B+Tree modeled in role-based and blockchain technology in association with the encrypted dimension and fact data for

TABLE 1. Functionality comparison.

Scheme	Access Control Mechanism	Data Warehouse (DW)/Big Data (BD)	Encryption Technique	Encryption Object	Ciphertext Query Method	Revocatoin Support
[7]	ABAC	DW	Homomorphic and OPE	Measures & Descriptive Attributes	SSB	No
[8]	RBAC	BD	CP-ABE	Big Data	No	Yes
[17]	NA	DW	AES128	Document	No	No
[18]	NA	DW	AE and AES-129	Dictionary	Big data	No
[19]	NA	DW	Homomorphic	Dimension & measure	No	No
[20]	NA	DW	Paillier-PCR	Measure data	Private Block Index	No
[21]	NA	BD	No	No	No	No
Ours	RBAC	DW	AES256 and CP-ABE	Cube	B+TREE	Yes

serving both privacy-preserving access control and fast query response in CDW setting. We summarize the contributions of our proposed scheme as follows.

1) We proposed a new cryptographic-based access control scheme called PPAC-CDW for outsourced data warehouse system where the dimension and fact data are encrypted. The query results are returned as the encrypted materialized view based on CP-ABE. This enables both privacy-preserving OLAP query and a fine-grained access control for users in CDW.
2) We implemented a novel approach to querying data cubes using B+Tree indexing combined with role-based modeling, which is then applied to an encrypted cube. This enables efficient OLAP queries to be conducted over an encrypted data warehouse. Our proposed MV retrieval technique significantly optimizes the query time and enables ease and scalable management of the access control to the group of organizational users.
3) We devised a traceable and efficient user revocation management protocol to enable optimized cost of ciphertext re-encryption based on proxy re-encryption. Our scheme leveraged blockchain and smart contract to support user authentication and user revocation checking making the system accountable and robust for managing users in the large-scale data warehouse. This also improves the scalability and system fault-tolerance due to the decentralization and data replication of blockchain.
4) We conducted the comparative analysis to demonstrate the computation cost and performance of our proposed scheme and related works.

The rest of the paper is structured as follows. Section II discusses related works. Section III presents the background theories used in our proposed system. Section IV presents our proposed scheme. Section V gives the security analysis. Section VI describes the implementation and evaluation. Finally, the conclusion and future work are given in section VII.

II. RELATED WORK

Recently, there are several works [7], [8], [9], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26] focusing on the security and privacy of data warehouse and big data implemented in the cloud environment. Here, access control models featured encryption techniques [3], [8], [17], [18] are employed to achieve the privacy-preserving solution of outsourced data and query. Specifically, this paper discusses the work dedicated to access control and DW or OLAP data security.

In [7], the authors proposed an encryption method for securing the data warehouse and the related OLAP system. The proposed algorithm supports queries over encrypted DW data hosted in the cloud. The proposed system performs several encryption tasks based on the statistical properties of target DW data. The authors also conduct experiments to test the performance of OLAP queries done over the encrypted DW. However, several encryption states used to render the expensive cost in practice.

In [21], the authors investigated an implementation and assessing a privacy-preserving OLAP framework namely SPPOLAP, which is a system emphasizing the privacy notion for aggregated OLAP query instead of data cube cells. The authors applied a privacy-preserving OLAP perturbation-based technique which uses the privacy grid to combine partition domains of the cube and the value is thus indistinguishable. Nevertheless, this approach does not provide the access control featured with the fine-grained privilege to the users and the cost of perturbing the aggregated data is expensive as it needs to be computed for all queries.

In [17], the authors introduced an effective sensitivity analysis method using approximate query processing for classifying documents to limit sensitive information leakage. The leakage assessment and parameter extraction algorithm were invented for cloud data warehouses based on the connection network and attribute network construction to evaluate approximate query processing. However, this paper does not entail the privacy-preserving solution for CDW.

In [19], the authors proposed a CloudWar system using a homomorphic encryption algorithm for securing and querying a data warehouse hosted in the cloud. For homomorphic privacy, all cell values are converted into perturbation values. Also, the weighted value for answering range query is introduced to reduce time complexity and communication overhead. However, the complexity of homomorphic key generations and their encryption cost are the core overheads when the system is accessed by a large number of users.

In [20], the authors proposed a privacy-preserving OLAP query based on private cell retrieval from a data warehouse and the Paillier cryptosystem. In this scheme, the client can securely perform OLAP operations on the data warehouse and retrieve the cube cells without disclosing any information. The proposed scheme encrypts all measured value by using the system's public key while dimension attributes are not encrypted. To decrypt the measured value, the authorized user needs to request the server for decryption. The cost for the client to query the data warehouse is thus propositional to the number of decryption queries. However, this work provides a limitation on the server dependency and the communication cost is high when there are a high number of decryption requests.

As shown in Table 1, only scheme [8] and ours used RBAC access control mechanism in which users are assigned to access shared data with specific privilege based on their role. Regarding the data privacy-preserving technique, our scheme applied AES-256 to encrypt the large volume of data and used CP-ABE to encrypt the symmetric key while scheme [18] used AES to only encrypt the data dictionary. Scheme [8] applied CP-ABE to support privacy preserving big data. Scheme [7] and [19] used homomorphic encryption to encrypt measures and dimension attributes. To provide the data retrieval method, only ours, [7], and [20] that provide full privacy-preserving over the measures or dimensional data provide the way to retrieve encrypted query result. Finally, only scheme [20] and ours support user revocation. However, scheme [20] worked on the actual big data and did not provide the query method.

Recently, we proposed a secure and verifiable Boolean keyword searchable encryption over encrypted CDW [28] based on the integration of bitmapping and inverted indexing techniques and blockchain technology. However, this work did not focus on fine-grained access control with the support of user revocation in CDW setting.

To the best of our knowledge, there are no works supporting both fine-grained access control and practically efficient OLAP query over the encrypted data warehouse outsourced in the cloud. Most schemes separate encryption and access control mechanisms in different stages of implementation. Rather, querying encrypted DW requires the assistive use of a searchable encryption technique which deals with index encryption and decryption cost. Lastly, the user revocation issue in most cryptographic-based access control models require re-encryption and key update of all non-revoked users in order to satisfy forward and backward security. However, the cost of revocation is even more costly due to a large number of encrypted cubes or encrypted queries and users.

In this paper, we tackled all the above issues by engaging CP-ABE, blockchain, and B+ tree to support secure, efficient and practical privacy-preserving OLAP query with efficient revocation in cloud data warehouse. Our proposed scheme provides fast encrypted cube which is a resulting of OLAP query based on our proposed indexing mechanism and blockchain without the exhaustive search over all data cubes in the entire data warehouse.

III. BACKGROUND

This section describes the background of materialized views, bilinear maps, and access tree used in our system model.

A. MATERIALIZED VIEWS

In data warehouse, a materialized view (MV) is a pre-computed view result comprising aggregated and/or joined data from fact and possibly dimension tables. In MOLAP, a DW is modeled in the multidimensional space where multiple dimensions are formed and associated with the measure attribute. The precomputed view can be calculated from the possible aggregation operations of the dimensions and measured in a cube.

Definition 1 (Multidimensional Space): Let Ω be the space of all dimensions. For each dimension D_i, there exists a set of levels, denoted as levels(D_i). A dimension is a lattice (H, <) of levels. Each path in the lattice of a dimension hierarchy, beginning from its least upper bound and ending in its greatest lower bound is called a dimension path. For example, the dimension path [day, week, month, year] is represented as day<week<month<year.

Definition 2 (Dimensional Level Space): Let Ψ be the space of all dimension levels. We can find the dimension where a dimension level (DL) belongs to, through the operator h: $h(DL_i) - D$ if $DL_i \in$ levels(D). For each dimension level, there is a set of values belonging to it (e.g. dimension level "city" has "Bangkok", "Tokyo", "London", "NewYork" as values). We define dom(DL_i) as the set of all the values of a dimension level DL_i.

Definition 3 (Base Cube): A base cube C_b as a 3-tuple< D, L, R> where

- $D = <D_1, D_2, \ldots, D_n, M>$ is a list of dimensions (D_i, M $\in \Omega$). M is a measure of the cube.
- $L = <DL_1, DL_2, \ldots, DL_n, *ML>$ is a list of dimension levels (DL_i., *ML$\in$ Ψ). ML is the dimension level of the measure of the cube.
- R is a set of cell data formed as a tuple $x = (x_1, x_2, \ldots, x_n, *m)$ where I in [1,...,n], $x_i \in$ dom(DL_i) and *m $\in$ dom(*ML).

In our model, we assume that materialized view represents all possible views of the base cube c. Each view is computed from the set of aggregation operations including sum, avg, count, max, min, rank(n). Each one of the operations results in a new cube c' or a materialized view (MV). Table 2 shows an example of a simple base cube for the loan data warehouse of a banking system.

As shown in Table 2, a base cube $C_b = <$ D, L, R> where D = Time, Customer, Branch, Loan Account Type, Amount, Loan, Risk, Compliance, Teller, Auditor, L = Day, Customer Name, Location, Loan Amount, Team, R is shown in the above Table. The query is done over the cube through operations such as roll-up, drill down, slice, dice, and navigate.

TABLE 2. Example of base cube for loan data warehouse.

Time	Customer	Branch	Loan Account Type	Amount (USD)	Position
Time	Customer	Branch	Loan Account Type	Amount (USD)	Position
01-01-2021	John W.	B001	Type A	50,000	Teller
01-02-2021	Alice C.	B002	Type B	75,000	Loan
31-03-2021	Kevin B.	B003	Type A	40,000	Teller
01-01-2022	Bob T.	B001	Type C	65,000	Auditor
15-05-2021	Timmy	B004	Type B	80,000	Teller
15-01-2021	Sarah J.	B003	Type C	100,000	Loan
30-09-2022	Bob T.	B002	Type A	230,000	Compliance
15-10-2022	Alex F.	B004	Type B	85,000	Risk

B. BILINEAR MAP

Let G_0 and G_1 be two multiplicative cyclic groups of prime order p. Let g be a generator of G_0 and e be a bilinear map, e: $G_0 \times G_0 \rightarrow G_1$. The bilinear map e has the following properties:

- Bilinearity: $\forall_{u,v} \in G_0$ and a, b $\in \mathbb{Z}_p$, $e(u^a, v^b) = e(u,v)^{ab} = e(u^b, v^a)$
- Non-degeneracy: $e(g, g) \neq 1$
- Computability: $\forall_{u,v} \in G_0$, an efficiently computation of e(u,v) exist

Definition 4 (Access Structure): Let a set $\{P_1, P_2, \ldots, P_n\}$ be a given set of attributes. A collection $A \subset 2^{\{P_1,P_2,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in A$ and $B \subset C$, then $C \in A$. An access structure is a monotone collection A of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e., $A \subset 2^{\{P_1,P_2,\ldots,P_n\}} \setminus \{\varnothing\}$.

Definition 5 (Access Tree T): Let T be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children, and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq \text{num}_x$. When $k_x = 1$, the threshold gate is an OR gate, and when $k_x = \text{num}_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$. If the k-of-n gate is allowed in T, in this case, $k_x = k$ where k is the threshold value determined in the k-of-n gate. In addition, while having an AND gate from the above definition as a right-node root node, we have another branch of a left-child root node that is another OR gate representing the role nodes. A role node is an OR gate which consists of two parameters: position and node key value (NKV). Each user has been assigned a unique NKV that is associated with his position in the overall system. In addition, all ACPs have the Proxy's IP address connected to OR gate to allow the delegation of proxy's decryption capability. In our scheme, access tree T is called as access control policy (ACP).

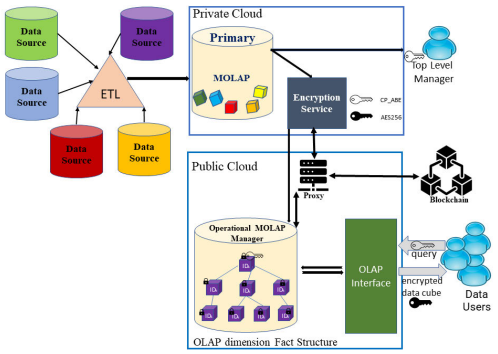

FIGURE 1. Our proposed system model.

IV. OUR PROPOSED SCHEME

This section presents the system model of our proposed MEMV scheme and provides the details of its system components. Figure 1 represents our proposed system model.

There are the following entities constituted in our systems:

1) **Data sources** refer to multiple sources of data that are heterogeneous in their formats, volume, and locations. ETL tool is a system responsible for normalizing the data by extracting data from sources, transforming the multiple data formats into the common schema able to be processed by the data warehouse and OLAP tool, and loading the data to be stored in the warehouse.
2) **Private Cloud** stores the first stage of pre-computed views after the ETL process. In our system, we assume that private cloud is an isolated environment where the access control is accessible by only one tenant or organization. We also locate the encryption service in the private cloud to support data encryption before it is sent to the public cloud for supporting OLAP query to the data users.
3) **Public cloud** stores encrypted cubes which connects to the OLAP interface and blockchain where the query and access control are performed respectively.
4) **Data Users** are the entities authorized to access and make a query over the data warehouse. Each user is assigned a decryption key to decrypt the encrypted query or data cube. In our scheme, users making the query are managed in the role-based of which the set of qualified MVs is bound to.
5) **Blockchain** retains access transactions and smart contracts supporting authentication, and user revocation. In our model, there are three smart contracts including (1) authentication contract which verifies identity and validation of requested users, (2) revocation list updation contract which adds the revoked user into the revocation list, and (3) attribute & NKV validation contract that checks the attribute hold by the revoked user and the existing node key value.
6) **Proxy** is a semi-trusted server that is responsible for ciphertext re-encryption when there is a revocation case.

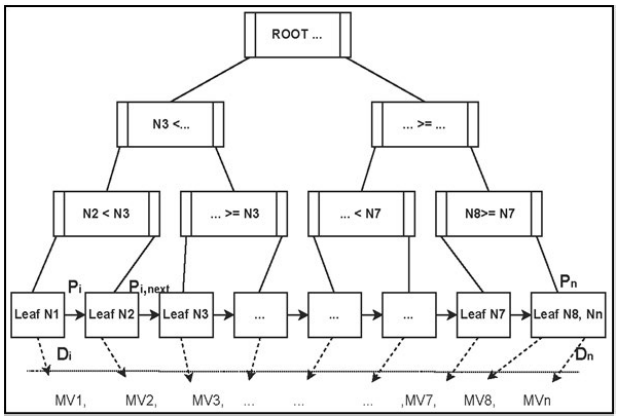

FIGURE 2. Generic B+Tree structure.

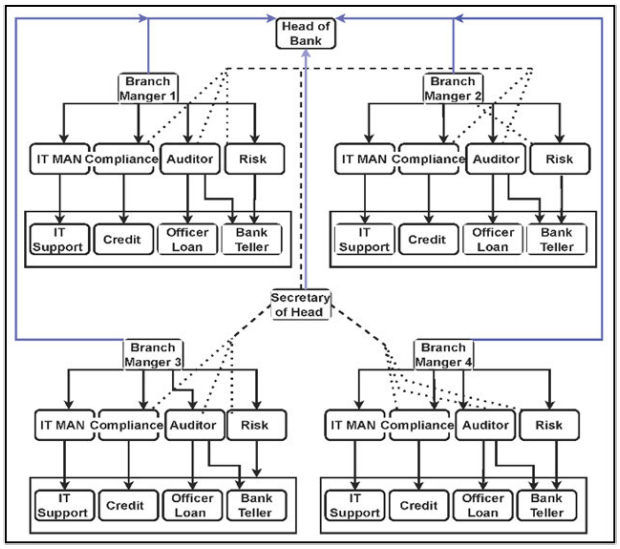

FIGURE 3. B+Tree structure of roles for loan data warehouse.

In our scheme, we applied the B+Tree model [27] with some existing related papers [29], [30], [31] to structurally formulate the index of a set of encrypted MVs to support efficient query retrieval done by the OLAP query.

A. OUR PROPOSED ROLE OF USERS AND B+TREE BASED MV MAPPING

Let B+Tree be the top level of a binary search tree in which the search operation supports fast multi-dimensional views at multiple levels of indexing. Let h be the height of the tree, N be the number of node key values (NKV), and M be the maximum number of child nodes in the tree.

The minimum number of NKVs in each non-root node is $\lceil \frac{M}{2} \rceil$, and the maximum number of NKVs is $M - 1$. The minimum number of children in each internal node is $\lceil \frac{M}{2} \rceil + 1$. In our system, each role has a unique NKV. We separate the structure of the tree into two categories: internal nodes represented by 'IntNode' and external nodes or leaf nodes represented by 'OutNode'.

For the 'IntNode' structure, every internal node structurally has $N_1 < N_2 < N_3 \leq N_4 < \ldots < N_7 \leq N_8 < N_9 < N_n$, where N of n is the unique NKV of each node and $n \in \mathbb{Z}_p$. The symbol placed before and after each NKV is a tree pointer P_i that corresponds to each node and is used to point to another node in the tree.

For the 'OutNode' structure, each leaf node has both an NKV and a data pointer D_i for each particular node, where D_i points to the encrypted material views of the data warehouse. All leaf nodes and their NKVs are at the same level and are sorted in ascending order starting from $N_1 < N_2 < N_3 < \ldots < N_8 < N_9 < N_n$, where $n \in \mathbb{Z}_p$. Figure 2 above shows the generic structure of our proposed B+Tree model.

As for the running example for the loan data warehouse of the banking system, the structure of the B+Tree can be represented in Figure 3. We build the B+Tree data structure as follows:

1) There are 4 branches, and each branch has 9 roles including a branch manager. We have another 2 roles as the top-level managers known as the Secretary of Head and the Head of Bank.
2) Head of Bank: can access everything from the 4 branches.
3) Secretary of Head: can access all sub-managers of all branches except the IT managers.
4) Branch Manager: can access all sub-managers such as IT, compliance, auditor, and risk managers.
5) IT Manager: can only access data of IT support staff.
6) Compliance Manager: can only access data of credit analyst staff.
7) Auditor Manager: can only access data of bank tellers and loan officers.
8) Risk Manager: can access data of every staff member to evaluate and mitigate risk at all costs.
9) IT Support, Credit Analyst, Bank Teller, and Loan Officers: can only access the information within their own team.

Based on this role division, we can extend the number of staff for each team accordingly. Specifically, we define each role with a unique range of NKVs in a list of lists. This simply refers to a role, such as "Bank Teller", for the list of all lists. For instance, (['10', '20', '30', '40'] is the first list in the NKV lists that represents the role of "Bank Teller". Then, we map each list in the NKV lists into four separate lists of all lists with branch numbers. For example, ['10', '20', '30', '40'] will be mapped with ['B001', 'B002', 'B003', 'B004'] respectively. We now have a new list of lists: [['10', 'B001'], ['20', 'B002'], ['30', 'B003'], ['40', 'B004']].

Next, we map the role and NKV together into a list of lists in which we append each index to the branch list. Therefore, when Data Users (DUs) make a query that contains the NKV range, it goes directly to that index and returns the NKV. This will call the bplustree.find($MV_{[i][ii]}$ search function as illustrated in Algorithm 1.

Algorithm 1 B+Tree Structure of Bank System

```
Result: When we give any NKV as an input such as "50", we will get {'LoanOfficer', 'B001'} and so on. Everything is stored on MVRole[i].

Role = {'BankTeller', 'LoanOfficer', 'ITSupport', 'CreditAnalyst', 'ComplianceManager', 'ITManager', 'Auditor', 'RiskManager', 'BranchManager', 'Secretary', 'HeadOfBank'}

NKV = {{'10','20','30','40'},{'50','60','70','80'},{'85','90','95','100'},{'102','104','106','109'},{'110','112','114','116'},{'118','120','122','124'},{'128','132','136','140'},{'142','144','146','148'},{'150','152','154','156'},{'157','158'}}

Branch = {'B001', 'B002', 'B003', 'B004'}

MVfi = dict(zip(Role, NKV))
for i = 0; i < 10; i = i + 1 do
    for value in 0; value <= branch; value++ do
        MVRole[i] = []
        if i < 6 then
            indexes = MVfi.index(value)
            MVRole[i].append(indexes)
            i = i + 1
        else
            MVRole[i] = MVfi[i]
            i = i + 1
        end if
    end for
end for
```

Algorithm 2 Search Function and Retrieve Data

```
input: MVRole[i], ..., MVRole[N]
output: mappedMV(CTMV[i])
function bplustree.find(MVRole[i],[ii])
    current-node = self.root
    while current-node.node.check_leaf = False do
        temp2 = current-node.values
        for i in range(len(temp2)) do
            if value ≤ temp2[i] then
                current-node = current-node.keys[i+1]
            end if
            if value < temp2[i] then
                current-node = current-node.keys[i]
            end if
            if i + 1 = len(current-node.values) then
                current-node = current-node.keys[i+1]
            end if
        end for
    end while
    return current-node
end function
return mappedMV
while True do
    MV = input("makematerializedqueryhere")
    if MV in MVRole[i] then
        bplustree.find(MVRole[i][ii])
    end if
end while
```

The pseudocode of the above example of constructing B+Tree from the banking hierarchy tree is formulated as shown in the Algorithm 1 while the searching and retrieving function done over encrypted MVs through NKV value are shown in the Algorithm 2.

Figure 4 exhibits the example of our constructed B+Tree for the loan data warehouse scenario. For instance, the query is made and the NKV "90" is included. The search function will be executed starting at the root of the tree. The root node has a four keys range (4 data sizes) containing 70, 102, 114, and 128. It searches through each NKV from left to right. It initially compares "90" to "70" since the input NKV is bigger than the compared one, the index of compared NKV moves to the right side for the bigger NKV. This index stays between 70 and 102 which infers the meaning of a value that is starting from 70 up until 102. Then, it goes down to the child node to compare "90" with 85 and it is again smaller than 90. It moves the index to the right side of 85 which is in the middle of 85 and 95. This index indicates the NKV range from 85 up to 95. It goes down and we finally see the leaf node that contains 85 and 90. Since all leaf nodes are linked together as a linked list, the search function finds 90 in that node. Upon the finding of the resulting NKV, the pointer directs the search result to particular encrypted M.

B. USER AUTHENTICATION

We developed a smart contract called "authenticateBC()" to check the identity of the users. This function takes userID, their secret key SKR, and NKV as inputs and outputs the status "True" for successful authentication. It checks whether any DUs are suspicious and unallowed to access our system by taking the input from DUs that corresponds to encrypted ciphertext $CTMV_{[i]}$ storing in our B+Tree data structure. The pseudocode of this smart contract is written below.

As presented in the Algorithm 3, if every input is valid, DUs are allowed to take further action in our proposed system. This smart contract will be executed whenever a DU makes a request to OLAP interface and proceed to the decryption phase.

C. CRYPTOGRAPHIC CONSTRUCT

In this section, we describe the cryptographic construct of our proposed model. Basically, the cryptographic of our system is based on symmetric encryption AES-256 and Ciphertext-Attributed based Encryption CP-ABE. There are

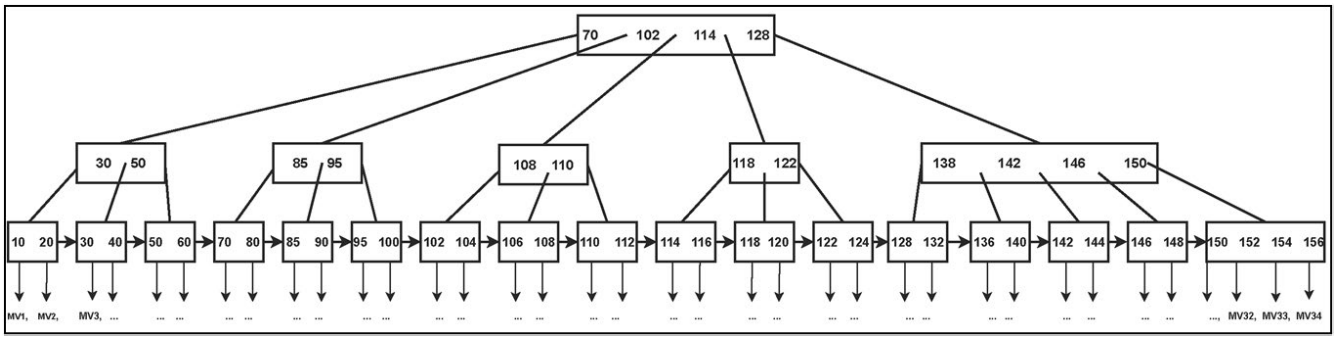

FIGURE 4. Example of B+Tree structure of a bank system.

Algorithm 3 BC Authentication (Smart Contract)

```
input: userID, SK_R, NKV
output: True
function authenticateBC(userID, SK_R, NKV)
    if userID is True then
        if SK_R not in Revol then
            if NKV in bplustree.find(MVRole[i][ii] then
                return True
            end if
        end if
    else
        return False
    end if
end function
```

TABLE 3. Notation.

Notation	Description
ACP	A CP-ABE based access policy
CT_k	An encrypted symmetric key
CT_{MVi}	A ciphertext of MVs encrypted by the symmetric encryption
$mappedMV$	A result from running query that contain both encrypted AES-256 key and encrypted MVs. It can be called a bundle of both $CTMV_i$ and CT_k
MK_k	AA's Master Secret key
$MV_{ROLE[i]}$	A materialized view that belongs to Role i and Attribute as a list that has index i corresponding to the NKVs
NKV	A key value to be used in B+Tree
PK_k	AA's Public key
R	A random value generated by CSPRNG to be used for AES256 encryption.
nonce	Unique random value
$Revol$	A revocation list that contains the list of users whose SKR is eligible for key updating.
S_A	A set of attributes that can be issued to DU
SK_R	A secret key constructing from SA associated user's role.
SK'_R	A new secret key for whom share the same role to a revoked user.
SK_{Proxy}	A secret key for semi-trusted proxy constructing from its IP address and security parameters.

five phases including Setup, Keygen, Encryption, Decryption, and Revocation. To ease of describing our proposed cryptographic algorithms, we present the notations used in our scheme in Table 3.

1) PHASE1: SETUP PHASE

CreateAuthenticatedAuthority(AA)->PK_k, MK_k. The setup algorithm considers security or system parameters and returns the pubic key PK_k and master key MK_k.

2) PHASE2: KEYGEN

There are three key types: symmetric key, user secret key (CP-ABE key), and proxy secret key (CP-ABE Proxy) used in our system. Each key type is generated through three algorithms: systemKeygen, duRoleAttributeKeygen, and proxyAttributeKeygen accordingly.

1) ***systemKeygen(keyGen)*** $\rightarrow$ ***symKey***
This algorithm takes keyGen as an input where `keyGen = CSPRNG.selectRandomKey()` and `keySize=256`. It returns the `symKey` for AES-256. The AA then sends the `symKey` to DUs.
2) ***duRoleAttributeKeygen(PK, MK, SA)*** $\rightarrow$ ***SK_R***
This algorithm is run by the AA. It takes as input `PK`, `MK`, and `SA`. The `SK` of the DU is created by the algorithm using a randomly selected $r \in \mathbb{Z}_p$, and each attribute $j \in S$ will be represented by randomly selecting $r \in \mathbb{Z}_p$, resulting in the following:

$$SK_R = \left(D = g^{(\alpha+r)/\beta}, j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j}\right)$$

The AA then sends the SK_R to the DUs.
3) ***proxyAttributeKeygen(PK_k, MK_k, Proxy's IP)*** $\rightarrow$ ***SK_{Proxy}***
This algorithm is run by the AA. It takes as inputs `PK`, `MK`, and the IP address of the proxy. It outputs the proxy secret key, SK_{Proxy}. Before AA sends SK_{Proxy} to the proxy, AA has to encrypt the key using random encryption. Here, the random encryption is done through the cryptographically secure pseudorandom number generator: `CSPRNG.secureSelectRandomValue()` and `keySize=256`. Then, AA encrypts the SK_{Proxy} by the following:

$$Enc_{CSPRNG.secureSelectRandomValue} = EncR\_SK_{Proxy}$$

Then, R is divided into two parts and they are shuffled through the following functions:

$$Divide(R) \rightarrow R_1 \| R_2$$
$$Shuffle(R_1 \| R_2) \rightarrow R_2 \| R_1 = R'$$

Then, R' is sent to the BC and the Enc_SK_{Proxy} is forwarded to the proxy.

3) PHASE3: ENCRYPTION

We perform dual encryption based on the symmetric encryption AES-256 and CP-ABE, which are managed by our encryption service located in the private cloud. The details of the encryption steps are presented as follows:

1) MVs Encryption
The algorithm takes a symmetric key symKey to encrypt the MVs as the input and outputs the encrypted view $CTMV_i$.

$$Enc(MVs, \texttt{symKey}) \rightarrow \texttt{CTMV}_i$$

Then, the encrypted materialized views $CTMV_i$ are sent to store in the public cloud storage.

2) symKey Encryption
This algorithm takes AA's public key PK_k, the symmetric key symKey, and access control policy ACP as inputs. Then it outputs the ciphertext of the encrypted symmetric key CT_k.

$$Enc\text{-}CP\text{-}ABE(\texttt{PK}_k, \texttt{symKey}, \texttt{ACP}) \rightarrow \texttt{CT}_k$$

Then, CT_k is forwarded to store in the public cloud and bundled together with the corresponding $CTMV_i$ in the B+Tree structure.

4) PHASE4: DECRYPTION PHASE

The decryption phase is activated upon the OLAP query made by the DUs. There are three stages as follows:

1) CT_k and $CTMV_i$ Retrieval
We make a query that includes the NKV to execute the function `bplustree.find(MVRole[i][ii])` and it will return the bundle of the ciphertext and CT_k.

2) symKey Decryption
This algorithm takes the secret key SK_R and outputs the symmetric key symKey. This algorithm is run by the cloud.

$$Dec\text{-}CP\text{-}ABE(\texttt{SK}_R, \texttt{CT}_k) \rightarrow \texttt{symKey}$$

Once the NKV search is found and matches the existing one in the B+Tree function, the algorithm is run to decrypt the CT_{symKey} and get the symKey. Then, DUs who made the query are returned with the encrypted views $CTMV_i$, via the OLAP interface.

3) Symmetric Decryption
This algorithm is run by the DU. It takes the symmetric key symKey to decrypt the encrypted $CTMV_i$ and outputs the MVs.

$$Dec(\texttt{symKey}, \texttt{CTMV}_i) \rightarrow MVs$$

Then, the resulting MV, which is a query result, is obtained.

5) USER REVOCATION

In this phase, it consists of 4 steps in which the BC and proxy server cooperatively work to complete the revocation process. Basically, the revocation list "Revol" contains the userID and the status of all users. Revol is stored in the blockchain, and all system entities have a uniquely assigned userID. To revoke a user, the data owner must send the revocation request to the proxy server, which will forward the request to the BC. The BC verifies the user who made the request and returns Revol to the proxy with the random value R' to let the proxy run the divide and reshuffle functions as follows:

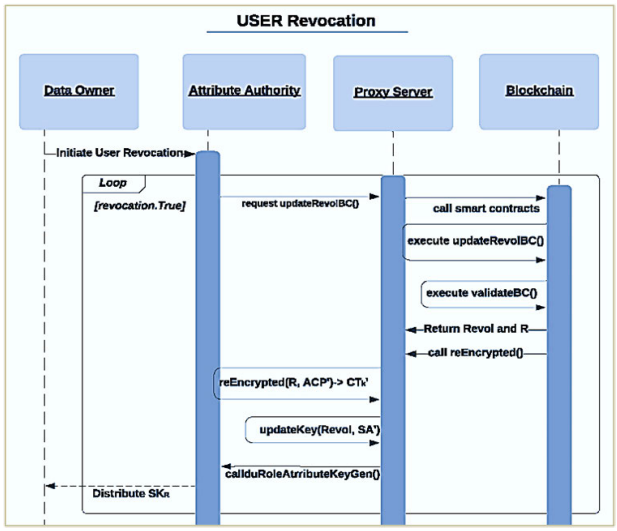

FIGURE 5. Sequence diagram of user revocation process.

$$Divide(R') \rightarrow R_2 \| R_1$$
$$ReShuffle(R_2 \| R_1) \rightarrow R_1 \| R_2 = R$$

The derived R is used to decrypt the encrypted secret key Enc_SK_{Proxy} through the following function:

$$Dec(Enc\_SK_{Proxy}, R) \rightarrow SK_{Proxy}$$

Figure 5 shows the overall user revocation of our proposed system.

The details of the revocation steps are done through the following algorithms.

1) *updateRevolBC(userID, NKV)* $\rightarrow$ *Revol*
This smart contract function takes userID, which is given to every user in the system, and the NKV defined in our B+Tree structure as inputs. It outputs the new Revol list. In this step, the data owner initiates the revocation process by requesting the blockchain to execute this smart contract. This algorithm will add a revoked user's identity and call the Revol List Validation smart contracts to validate the attributes and NKV. The algorithm of `updateRevolBC()` contract is as follows.

2) *ValidateBC(userID, NKV)* $\rightarrow$ *Revol*
This smart contract takes userID and NKV as inputs and outputs the validation of the new Revol. This smart contract is executed by the blockchain. It stores the revocation list of all old users that contain SA and userID belonging to those users to compare with the

Algorithm 4 Revol List Update (Smart Contract)

```
input: userID, NKV
output: Revol
function updateRevolBC(userID, SK_R, NKV)
    Revol = Revol.add(userID, NKV)
    return Revol
end function
```

new updated list for proof of validation. It supports user revocation by checking the `Revol` list to see whether it has stored any revoked user or not. The algorithm of this smart contract is written below.

Algorithm 5 Revol List Validation (Smart Contract)

```
input: userID, NKV
output: Revol
function validateBC(userID, NKV)
    int indexRole = 0
    if oldRevol(userID, NKV) ≠ Revol(userID, NKV) then
        Revol(userID, NKV) = Revol(userID, NKV)
        for indexRole in range 9 do
            for NKV in MVf1[NKV] do
                sameRoleList = [NKV for NKV in MVf1[NKV]]
            end for
        end for
        Revol.update(sameRoleList)
    end if
    return Revol
end function
```

The code checks whether the existing `Revol` and the new `Revol` are the same or not. If not, we update the `Revol` and retrieve all `NKV` that share the same role as the revoked user for further `updateKey()` algorithm.

3) *reEncrypt(R, ACP') → CT$_k$'*
This function takes the random value R to decrypt the encrypted secret key of the proxy from the blockchain and a new access control policy `ACP'` as inputs. It outputs a new encrypted symmetric key `CT_k'` which is bundled with the encrypted ciphertext `CTMV_i` stored in our B+Tree data structure. `ACP` is updated before the re-encryption process starts and it contains two characters: `NKV` and a set of attributes that define the accessibility of the user's privilege. If any `NKV` is changed, the information in `ACP` will also be updated. This algorithm is executed by the proxy located in the cloud once the return status from the blockchain shows the invalid status of any revoked user. The algorithm below shows how the `symKey` is re-encrypted.
The proxy runs the algorithm to decrypt the existing encrypted `symKey CT_k` with its secret key. Then, the `symKey` is re-encrypted corresponding to a particular `CTMV_i` with the new access control list `ACP'`. The

Algorithm 6 Proxy reEncryption

```
input: R, ACP'
output: CT'_k
function reEncrypt(userID, SK_proxy, NKV)
    Dec(Enc_SK_proxy, R) → SK_proxy
    Dec-CP-ABE(CT_k, SK_proxy) → symKey
    Enc-CP-ABE(PK_k, symKey, ACP') → CT'_k
    updateKey()
end function
```

algorithm then outputs a new encrypted `symKey` `CT_k'` that will be re-bundled with the corresponding `CTMV_i` in our B+Tree. Then, the `updateKey()` algorithm is called to update the `Revol`.

4) *updateKey(Revol, SA') → SK$_R$'*
This function takes `Revol` and a new set of attributes `SA'` as inputs. It outputs new secret keys `SK_R'` for all active users that share the same role as the revoked user. This function can be assisted by the B+Tree structure on `NKV` searching as well. Finally, the new updated revocation list `Revol` is obtained. The `updateKey()` algorithm is described as follows:

Algorithm 7 Proxy DU's Key Update

```
input: Revol, SA'
output: SK'_R
function updateKey(Revol, SA')
    removeRevokedUser()
    for NKV in Revol.Revoke do
        duRoleAttributeKeygen(PK, MK, SA')
    end for
    return SK'_R
end function
```

V. SECURITY ANALYSIS

This section discusses the security model and security properties of our proposed system.

A. SECURITY MODEL

In our model, we assume that AA is a trusted authority while private cloud storage is only accessible by the data owner. However, the public cloud is semi-trusted. In our system, all pre-computed cubes are encrypted with the AES encryption and stored in the public cloud. To preserve the confidentiality of the AES key, it is encrypted with CP-ABE method are stored on the cloud. Only authorized users having the secret key issued by the AA can decrypt the encrypted AES key and access the query results.

The security model of our scheme is defined as a game-based in compromising the CP-ABE key to obtain the capability in accessing the encrypted symmetric key. The game-based between an adversary A and a challenger C is defined as follows:

Setup. For uncorrupted authorities AA, the challenger C runs the `CreateAttributeAuthority` algorithm and sends public keys PK to the adversary A. For corrupted authorities AA', the challenger sends both the public key PK and secret key SK_R to adversary A.

Phase 1: The adversary A delivers Sk which is a set of attributes issued by an uncorrupted authority AA_k. The challenger C gives the secret key SK to the adversary A.

Challenge. Adversary A sends two challenge messages m_1 and m_2 to the simulator. The simulator flips a fair binary coin ν, and returns an encryption of m_ν. In this game, the CT_K, which is a ciphertext of the symmetric key encrypted by a CP-ABE method, is computed as follows:

$$CT_K = \Big(T, \hat{C} = m_\nu z, CT_k = h^s, \forall y \in Y : C_y = g^{q_y(0)}, \hat{C}_y = H(\text{att}(y))^{q_y(0)}\Big)$$

where γ is a chosen set of attributes. If $\mu = 0$ then $z = e(g,g)^{\alpha s}$. Therefore, the ciphertext CT_K is a valid random encryption of message m_ν. Otherwise, if $\mu = 1$ then $z = e(g,g)^z$. We now have $\hat{C} = m_\nu e(g,g)^z$. Since z is random, $\hat{C}$ will be a random element of G_1 from the adversary's view and the message contains no information about m_ν.

Phase 2: The simulator does as it did in Phase 1.

Guess Adversary A sends a guess of ν' of ν. The advantage of A in this game is defined as:

$$ADV_A = \Pr[\nu = \nu'] - \frac{1}{2}$$

Definition 3: Our proposed scheme is secure if all polynomial-time adversaries have at most a negligible advantage in the above game.

Theorem 1: Suppose there is no polytime adversary who can break the security of CP-ABE with non-negligible advantage; then there is no polytime adversary who can break our cryptosystem with non-negligible advantage.

Proof: As we have shown how the adversary A has a non-negligible advantage against our scheme. Similar to A, we show how the adversary B is created to break the CP-ABE scheme with non-negligible advantage. The adversary B can play a similar game with the CP-ABE scheme to make private queries during the game to get the private keys in the CP-ABE scheme.

Initialization. The adversary B takes the PK of the authority k, $PK'_k = \{G_0, g, h = g^\beta, f = g^{1/\beta}, e(g,g)^\alpha\}$, and the corresponding secret key (β, g^α) is unknown to the adversary.

Setup. The adversary B gets the public parameters from PK' as $PK_k = \{G_0, g, h = g^\beta, f = g^{1/\beta}, e(g,g)^\alpha\}$, then the public key PK_k is sent to the adversary.

Phase 1. B answers private key queries. Suppose the adversary is given a secret key query for a set of attributes S where S does not satisfy T. Here, B makes a query for obtaining SK for the same set S twice. Then, B obtains two different SKs as follows:

$$SK_k = \Big(D = g^{(\alpha_k + r)/\beta_k}, \forall i \in S : D_i = g^r \cdot H(i)^{r_i}, D'_i = g^{r_i}\Big)$$

$$SK'_k = \Big(D = g^{(\alpha_k + r')/\beta_k}, \forall i \in S : D_i = g^{r'} \cdot H(i)^{r'_i}, D'_i = g^{r'_i}\Big)$$

where i's are attributes from S, and r, r', r_i, r'_i are random numbers in $\mathbb{Z}_p$. With SK_k and SK'_k, B can obtain $g^{(r-r')/\beta}$, and chooses random numbers $t_i, t_{i,j} \in \mathbb{Z}_p$. Let $r^* = t_i - r_i$ and $r'' = t_{i,j} - r'_i$. Then B derives the SK requested by A as

$$SK^* = \Big(D = g^{(\alpha_k + r')/\beta_k}, \forall i \in S : D_i = g^{r^*} \cdot H(i)^{r''_i}, D'_i = g^{r''_i}\Big)$$

Then, the SK is returned to the adversary A.

Challenge. When A decides that Phase 1 is over, it outputs an access policy T and two messages m_1 and m_2, which it wishes to be challenged. B gives the two messages to the challenger and is given the challenge ciphertext CT_K. Then B computes the challenge ciphertext for A from CT_K as CT^*_K. Finally, the challenge ciphertext CT^*_K is returned to the adversary A.

Phase 2. A makes queries not issued in Phase 1. B responds as in Phase 1.

Guess Finally, A outputs a guess $\nu' \in \{1, 0\}$, and then B concludes its own game by generating ν'. According to the above security model, the advantage of the adversary B is:

$$ADV_A = \left|\Pr[\nu = \nu'] - \frac{1}{2}\right| = ADV_B$$

Thus, B has a non-negligible advantage against the CP-ABE, which completes the proof of the theorem. Since our cryptographic construct for accessing the decryption key is based on CP-ABE, the detailed proof can be referred to the original paper [32].

VI. EVALUATION

This section presents the computation analysis of our PPAC-CDW and related works including scheme [17], [19], [20]. We particularly chose works supporting privacy preserving data warehoused through the encryption method. In addition, we conducted experiments to measure the encryption, decryption, DW query, and revocation performance of related works and ours.

A. COMPUTATIONAL COST ANALYSIS

This section analyzes the computation cost of encryption, decryption, and query/search cost of the encrypted query result done over data warehouse. Table 4 displays a comparison between the computation costs of our approach and similar studies. To illustrate the representation of the computation cost for each approach, the following notations are used.

- G_0: Exponential operation in group G_0
- G_1: Exponential operation in group G_1
- E: Bilinear pairing operation
- $|AP|$: Number of attributes in access policy
- $|UA|$: Number of attributes in user secret key

TABLE 4. Comparison of computational cost.

Scheme	Encryption Cost	Decryption Cost	Query/Search Cost
[17]	$2G_0(AES_{Enc1} + XOR) + G_1(\|\|)$	$2G_0 + G_1(\|\|)AES_{Dec1}$	E + $2G_0$ + $\|DC\|+\|NC2\|$
[19]	$(4G_m)G_1$	$2G_m + G_0(G_1)$	$2G_m + 2G_1 + \|DC\| + \|NC\|$
[20]	$PCR_{Enc} + G_1 G_m$	$PCR_{Dec} + G_1 G_m$	$4G_1 G_m + \|DC\| + \|NC\|$
Ours	$(2\|AP\| + 1)G_0 + 2G_1 + AES_{Enc2}$	$(2\|UA\| + 1)E + (2\|AP\| + 2)G_1 + AES_{Dec2}$	H + |DC| + |NC|

- AES_{Enc1}: AES encryption operation of 128 bits
- AES_{Enc2}: AES encryption operation of 256 bits
- AES_{Dec1}: AES decryption operation of 128 bits
- AES_{Dec2}: AES decryption operation of 256 bits
- PCR_{Enc}: Paillier cryptosystem encrypted operation
- PCR_{Dec}: Paillier cryptosystem decrypted operation
- G_m: Multiple arithmetic operations in group G_0
- XOR: XOR operation in 128 bits
- ||: Concatenation operation in 128 bits
- H: Logarithm of the number of keys proportionate to the heights in B+Tree $O(1) - O(\log H)$
- $|DC|$: Number of dimensions in cube
- $|NC|$: Number of generated cubes
- $|NC2|$: Square root of multiple operations of data collections containing number of correlations

Our scheme applied a 2-step encryption consisting of AES and CP-ABE algorithm to encrypt the materialized view and the symmetric key respectively. Since the size of the view or the cube is relatively small, using AES algorithm even provides fast data encryption. Then, the CP-ABE method is applied to encrypt the 256-AES key. For the decryption case, the computation cost is subject to the number of attributes in the policy and the number of attributes contained in the user secret key together with the exponential operation of prime order group G_1 and bilinear pairing operation. All the encrypted MVs (NC) are indexed through B+Tree where the search operation on encrypted cubes MVs is subject to the three height H and the number of cubes generated from the dimensions. Specifically, our search cost does not deal with the prime order group as other works do. This makes our scheme took least query time compared to related works. For scheme [17], the encryption phase takes multiple XOR operations which are required to perform AES encryption of 128 bits in the exponential operation of G_0 and the concatenation operation of prime order group G_1. For the decryption phase, the computation cost is less than encryption phase because it does not need multiple XOR operation, and some operations are constructed when having a search for pair of value. The query cost is high due to the multiple searches of key pair value in N documents which consume NC2 operations and repeat the search for another sub key pairs of two different documents. However, the security of 126-bit AES encryption is considered insecure. In scheme [19], the encryption phase is done through homomorphic function that deals with arithmetic operation with the exponential operation of prime order group G_1. For decryption, it takes higher cost than ours due to the cryptographical construct of Chinese remainder and the exponential operation in group G_0 of G_1. The method of search operation on encrypted cubes is quite weighty because there are couple of arithmetic operations and result comparison. In scheme [20], the encryption and decryption are executed through the arithmetic in group of G_1 and mainly by Paillier cryptosystem which consumes more computational cost than AES-256. Moreover, the query method is more complex than other schemes as it needs to generate and get the response for particular encrypted cube.

B. PERFORMANCE ANALYSIS

We conducted the experiments to evaluate encryption, decryption, query, and revocation performance of our scheme and related works including [17], [19], [20]. The steps of our experiments are done with setting up the environment for cube construction based on Tiny OLAP opensource [33], a cloud proxy for data encryption and decryption, and the blockchain. Then, we used Python to construct cryptographic operations using built-in Python module and art [34] libraries.

The implementation is done via Python's Cryptography and we used Java-Pairing based Cryptography [35] and the Advanced Crypto Software Collection [36], [37] to simulate the cryptographic operations of our scheme. For scheme [17], we used pycryptodomex [38] and pycryptodome [39] libraries, and used numpy [40] with sympy [41] libraries, partially homomorphic encryption PHE for scheme [19] and [20] respectively. The experiments were done on an Intel(R) Xeon(R) E-2336 CPU @ 2.9GHz and 16 GB of RAM server that is running on the Ubuntu 20.04 Operating System. As the library and module are on the Ubuntu Server, we used python code instead of solidity to represent the blockchain. to support user revocation process. In our scheme, we used Ethereum as a blockchain platform for our simulation and used Solidity to develop three smart contracts.

1) **Encryption and Decryption Performance**
To measure the encryption and decryption performance, we compare the time used to encrypt and decrypt the cube or dimension and measure data of our scheme, scheme [17], scheme [19], and scheme [20]. The computation time was then measured by varying the number of MVs while maintaining a constant data size of 250 KB for all executions of all implemented algorithms. Figure 6 and Figure 7 depict the overall encryption and decryption costs for all schemes.
As shown in Figure 6, the comparison of the encryption time of each scheme compared to ours, our scheme's encryption time stays constant and minimal at all variations of the of MVs number, compared to other schemes. This is because we applied AES encryption to encrypt all the constant size of MVs and used the CP-ABE to encrypt the symmetric key. In scheme [19],

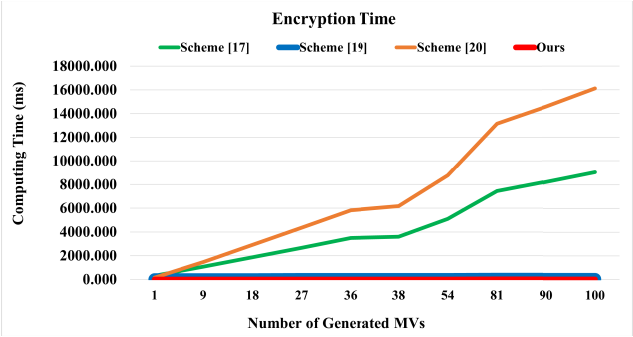

FIGURE 6. Total encryption cost.

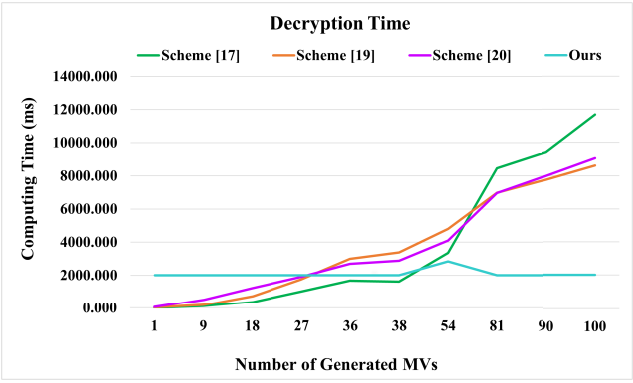

FIGURE 7. Total decryption cost.

the encryption is based on homomorphic method of which the cost is subject to the constant size of data. However, it is approximately three times higher than ours. Scheme [17] used random method of AES128bits with a default deterministic function which cost even higher than scheme [19] and ours. Scheme [20] took the most expensive encryption cost among all as it is executed by PHE partially homomorphic encryption.

Figure 7 demonstrates the comparison in computational time of each scheme compared to ours which includes both decryption cost and searching cost over encrypted MVs. Each scheme has their alternatives method to both decryption algorithms and retrieving the encrypted MVs to be decrypted. The graph depicts that our scheme's decryption time still stays constant although it takes more computing cost at the initial stages where the number of generated MVs are small. However, when the number of generated MVs are higher, the processing time for decryption in our scheme is mathematically less than other schemes due to the optimized search cost over encrypted MVs based on the B+Tree. In [17], the decryption time which is subject to the AES128 bits with the deterministic function and the search cost, initially yielded the least cost among all schemes. However, when the number of MVs increases, the processing time started growing sharply. This is because there were multiple searching times over encrypted data which split them into 4 types

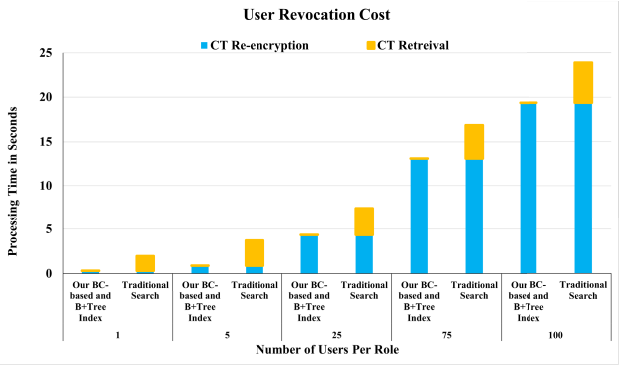

FIGURE 8. Detailed cost of user revocation.

of data. In scheme [19], the decryption cost is subject to the operation of CRT and homomorphic decryption and some searching comparison. In scheme [20], the decryption cost relies on Paillier cryptosystem and the search through complex request and response algorithm which account for more computation cost than scheme [19] and ours.

2) **User Revocation Performance**

We measured the user revocation cost by the addition of the processing time for retrieving the data which basically searching the right node key value from B+Tree data structure on the encrypted MVs and the processing time for re-encrypt the symmetric key by generating SA' and repeat the CP-ABE encryption. In this regard, blockchain is very handy to facilitate and foster the process of the whole user revocation process. Blockchain contains several smart contracts to fulfill the needs for completed user revocation. The first function is to create random and secure user credential information such as userID and password to be used for revoking process. The second contract is to generate Revol list to add and store the revoked user whenever the DO makes a request to OLAP interface. The third smart contract is to validate the attributes and NKV to support the decryption throughput. To this end, we did the experiment to demonstrate the detailed cost of the revocation that consists of the retrieval of the affected ciphertexts and the ciphertext re-encryption. Figure 8 illustrates the detailed cost of user revocation. We measured the revocation time by varying the number of users per role.

As shown in Figure 8, our experiment demonstrates that the performance of our proposed revocation mechanism based on blockchain and B+Tree outperforms the traditional search for retrieving the encrypted MVs to be used for the re-encryption process. With our proposed scheme, we used a node key value representing the pointer to any encrypted MVs where multiple users are only applicable or allowed to access one MVs based on their role. The proposed indexing scheme enables the efficient retrieval of encrypted cubes based on the role-based model. Hence, when the

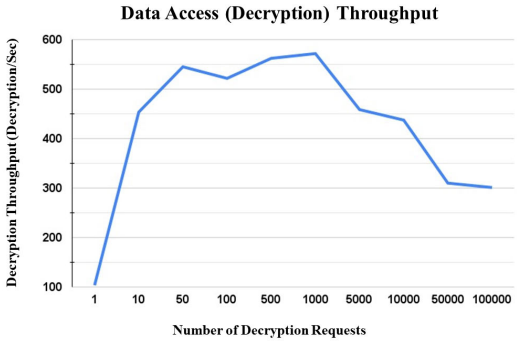

FIGURE 9. Decryption throughput.

number of users per role is huge, the smart contract combines multiple users who share the same role in a separate list for conducting the re-encryption process for each revocation request. Therefore, we only have to perform small search cost to get all users in the same role. In contrast, the solution that do not provide CT retrieval method generally rely on the traditional search where the exhaustive search is executed to search on one-by-one comparison in checking the CT ever accessed by the revoked user.

3) **Data Access (Decryption) Throughput**
Finally, we conducted the experiment to test the decryption throughput to determine how much our scheme accommodates the cube access transactions. The throughput was measured using the generation of concurrent multi-thread requests for supporting the data access request where the proxy runs search operation and re-encryption. Here, we used the fixed size of cube at 250KB, 5-attribute policy, and 5-attribute secret key. We varied the number of decryptions requests up to 100,000 requests and recorded the throughput as shown in Figure 9.
The result shown in Figure 9 indicated that our proposed scheme yielded the highest throughput at 572 in supporting 1,800 concurrent decryption requests per second. The system can still support the number of requests in the range of1000 to 10000 requests before there is a sharp decline when the number of requests exceed 10000 requests. In fact, the throughput performance is based on the hardware used to run the transactions. In real cloud environment, using our proposed scheme would render higher throughput as the resource dynamic provisioning, high computation with load balancing of cloud computing. The optimized cost for searching over encrypted MVs through B+Tree node key value index and CP-ABE decryption over the small size of symmetric key enables the high throughput and high utilization of computation resource.

4) **Processing Cost occurred in Blockchain**
Finally, we evaluate the performance of the smart contracts executed using the Blockchain technology by means of the gas cost. In our experiments, we simulated the network gas fees required by the blockchain to execute smart contracts. These contracts serve the purpose of authenticating users, adding-on a revoke user's identity, and validating the attributes and NKV of users in Revol, respectively; authenticateBC(), updateRevolBC(), and validateBC().
In our experimental setup, we imposed a gas limit of 3,000,000 and defined specific criteria for various smart contracts to enhance user authentication mechanisms. A dataset comprising 1,000 distinct users was synthetically generated, with each record consisting of a unique name, userID, uniqueString, password, and an activity status. During the revocation list update process, a parallel list search algorithm was employed to append additional identities to the existing dataset, ensuring there were no duplicates. The ValidateBC smart contract utilized the keccak256 hashing function alongside encoding techniques to authenticate the integrity of the new and existing revocation lists. The gas costs are shown in Table 5.
As depicted in Table 5, the estimated gas expenditures for executing each smart contract are presented. The gas price, which represents the amount of Ether (ETH) a user is willing to pay per unit of gas, is conventionally expressed in Gwei (1 Gwei = 10^{-9} ETH = 1 billion Wei). The cost of consumption in USD is calculated by multiplying the gas utilized by the gas price, which signifies the actual expense incurred for transaction execution or smart contract operations. Our findings indicate that the smart contracts for list update and validation incurred relatively minimal costs, whereas the authentication contract exhibited significantly higher gas fees due to the complexities involved in managing multiple identities. The incorporation of blockchain technology into our proposed framework did not markedly degrade the system's performance. On the contrary, it substantially augmented the reliability of user requests by ensuring robust authentication and validation, in addition to maintaining the integrity of search outcomes retrieved from public cloud services. Our system adopts a proof of stake consensus mechanism, favoring validators over miners to address challenges related to scalability, security, and fostering a more dynamic decentralized ecosystem.

TABLE 5. Blockchain Cost QUERY COST (Consider Gas Price Per Unit = 0.375USD).

Smart Contracts	Gas Consumption (in Wei)	Cost (USD)
`authenticateBC()`	3,678,747	0.0137948
`updateRevolBC()`	865,400	0.0032452
`validateBC()`	629,011	0.0023587

VII. CONCLUSION

We have proposed the privacy-preserving data warehouse access control scheme enabling secure and fine-grained

access control to both data warehouse outsourced in the cloud and the OLAP query made over the data warehouse. We leveraged CP-ABE and symmetric encryption to encrypt the data warehouse and devised the access policy to be constructed with a set of attributes and roles used to enforce the encryption for users in role-based model. We also introduced B+ Tree technique to support efficient data (cube) retrieval and employed blockchain to handle authentication and assist in the user revocation process. With the proposed solution, we achieve privacy protection of data warehouse with the secure and efficient delivery of the OLAP query. We conducted the experiments to measure the encryption time, decryption time, and revocation time of our scheme and related works. The results demonstrated that our proposed scheme outperforms related works for all experiments. For future works, the investigation on privacy preserving of multi-version of the changes of data warehouse is worth to study. In addition, the cross-domain big data sharing is challenging.

REFERENCES

[1] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure and efficient skyline queries on encrypted data," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 7, pp. 1397–1411, Jul. 2019.

[2] R. Li, Z. Xu, W. Kang, K. C. Yow, and C.-Z. Xu, "Efficient multi-keyword ranked query over encrypted data in cloud computing," *Future Gener. Comput. Syst.*, vol. 30, pp. 179–190, Jan. 2014.

[3] J. Chi, C. Hong, M. Zhang, and Z. Zhang, "Privacy-enhancing range query processing over encrypted cloud databases," in *Proc. Web Inf. Syst. Eng.*, Miami, FL, USA, 2015, pp. 63–77.

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, Jun. 2004, pp. 563–574.

[5] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, May 2009, pp. 169–178.

[6] A. Cuzzocrea, P. Karras, and A. Vlachou, "Effective and efficient skyline query processing over attribute-order-preserving-free encrypted data in cloud-enabled databases," *Future Gener. Comput. Syst.*, vol. 126, pp. 237–251, Jan. 2022.

[7] C. C. Lopes, V. C. Times, S. Matwin, R. R. Ciferri, and C. Dutra de Aguiar Ciferri, "Processing OLAP queries over an encrypted data warehouse stored in the cloud," in *Proc. Dawe*, 2014, pp. 195–207.

[8] S. Fugkeaw and H. Sato, "Privacy-preserving access control model for big data cloud," in *Proc. Int. Comput. Sci. Eng. Conf. (ICSEC)*, Nov. 2015, pp. 1–6, doi: 10.1109/ICSEC.2015.7401416.

[9] M. Kantarcioglu and F. Shaon, "Securing big data in the age of AI," in *Proc. 1st IEEE Int. Conf. Trust, Privacy Secur. Intell. Syst. Appl. (TPS-ISA)*, Dec. 2019, pp. 218–220, doi: 10.1109/TPS-ISA48467.2019.00035.

[10] S. Fugkeaw, "A lightweight policy update scheme for outsourced personal health records sharing," *IEEE Access*, vol. 9, pp. 54862–54871, 2021, doi: 10.1109/ACCESS.2021.3071150.

[11] N. Chen, J. Li, Y. Zhang, and Y. Guo, "Efficient CP-ABE scheme with shared decryption in cloud storage," *IEEE Trans. Comput.*, vol. 71, no. 1, pp. 175–184, Jan. 2022, doi: 10.1109/TC.2020.3043950.

[12] S. Das and S. Namasudra, "Multiauthority CP-ABE-based access control model for IoT-enabled healthcare infrastructure," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 821–829, Jan. 2023, doi: 10.1109/TII.2022.3167842.

[13] S. Fugkeaw, "Secure data sharing with efficient key update for industrial cloud-based access control," *IEEE Trans. Services Comput.*, vol. 16, no. 1, pp. 575–587, Jan. 2023, doi: 10.1109/TSC.2021.3110828.

[14] J. Yu, Y. Hou, S. Li, and Z. Wen, "A high-speed data retrieval model on blockchain," in *Proc. 11th Int. Conf. Inf. Commun. Technol. (ICTech))*, Wuhan, China, Feb. 2022, pp. 101–105, doi: 10.1109/ICTech55460.2022.00029.

[15] J. Bergers, Z. Shi, K. Korsmit, and Z. Zhao, "DWH-DIM: A blockchain based decentralized integrity verification model for data warehouses," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Dec. 2021, pp. 221–228, doi: 10.1109/Blockchain53845.2021.00037.

[16] Z. Zhou, M. Wang, J. Huang, S. Lin, and Z. Lv, "Blockchain in big data security for intelligent transportation with 6G," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9736–9746, Jul. 2022, doi: 10.1109/TITS.2021.3107011.

[17] M. Ahmadian and D. C. Marinescu, "Information leakage in cloud data warehouses," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 2, pp. 192–203, Apr. 2020, doi: 10.1109/TSUSC.2018.2838520.

[18] B. Fuhry, H. A. Jayanth Jain, and F. Kerschbaum, "EncDBDB: Searchable encrypted, fast, compressed, in-memory database using enclaves," in *Proc. 51st Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2021, pp. 438–450, doi: 10.1109/DSN48987.2021.00054.

[19] K. Karkouda, A. Nabli, and F. Gargouri, "CloudWar: A new schema for securing and querying data warehouse hosted in the cloud," in *Proc. 28th Int. Conf. Comput. Theory Appl. (ICCTA)*, Oct. 2018, pp. 6–12, doi: 10.1109/ICCTA45985.2018.9499193.

[20] X. Yi, R. Paulet, E. Bertino, and G. Xu, "Private cell retrieval from data warehouses," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1346–1361, Jun. 2016, doi: 10.1109/TIFS.2016.2527620.

[21] A. Cuzzocrea, V. De Maio, and E. Fadda, "Experimenting and assessing a distributed privacy-preserving OLAP over big data framework: Principles, practice, and experiences," in *Proc. IEEE 44th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2020, pp. 1344–1350, doi: 10.1109/COMPSAC48688.2020.00-69.

[22] S. Pendse, V. Krishnaswamy, K. Kulkarni, Y. Li, T. Lahiri, V. Raja, J. Zheng, M. Girkar, and A. Kulkarni, "Oracle database in-memory on active data guard: Real-time analytics on a standby database," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Apr. 2020, pp. 1570–1578, doi: 10.1109/ICDE48307.2020.00139.

[23] E. Guermazi, M. Ben Ayed, and H. Ben-Abdallah, "Adaptive security for cloud data warehouse as a service," in *Proc. IEEE/ACIS 14th Int. Conf. Comput. Inf. Sci. (ICIS)*, Jun. 2015, pp. 647–650, doi: 10.1109/ICIS.2015.7166672.

[24] K. Yang, X. Jia, and K. Ren, "Secure and verifiable policy update outsourcing for big data access control in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3461–3470, Dec. 2015, doi: 10.1109/TPDS.2014.2380373.

[25] G. Ra, D. Kim, D. Seo, and I. Lee, "A federated framework for fine-grained cloud access control for intelligent big data analytic by service providers," *IEEE Access*, vol. 9, pp. 47084–47095, 2021, doi: 10.1109/ACCESS.2021.3067958.

[26] Y. Reddy, "Big data processing and access controls in cloud environment," in *Proc. IEEE IEEE 4th Int. Conf. Big Data Secur. Cloud (BigDataSecurity), Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2018, pp. 25–33, doi: 10.1109/BDS/HPSC/IDS18.2018.00019.

[27] J. Yoo, H. Cha, W. Kim, W.-H. Kim, S.-S. Park, and B. Nam, "Pivotal B+ tree for byte-addressable persistent memory," *IEEE Access*, vol. 10, pp. 46725–46737, 2022, doi: 10.1109/ACCESS.2022.3170916.

[28] S. Fugkeaw, L. Hak, and T. Theeramunkong, "Achieving secure, verifiable, and efficient Boolean keyword searchable encryption for cloud data warehouse," *IEEE Access*, vol. 12, pp. 49848–49864, 2024, doi: 10.1109/ACCESS.2024.3383320.

[29] H. Shen, L. Xue, H. Wang, L. Zhang, and J. Zhang, "B+-tree based multi-keyword ranked similarity search scheme over encrypted cloud data," *IEEE Access*, vol. 9, pp. 150865–150877, 2021, doi: 10.1109/ACCESS.2021.3125729.

[30] C. Ho, K. Pak, S. Pak, M. Pak, and C. Hwang, "A study on improving the performance of encrypted database retrieval using external indexing system of B+ tree structure," *Proc. Comput. Sci.*, vol. 154, pp. 706–714, Jan. 2019, doi: 10.1016/j.procs.2019.06.110.

[31] W. Zhang, Z. Yan, Y. Lin, C. Zhao, and L. Peng, "A high throughput B+tree for SIMD architectures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 707–720, Mar. 2020, doi: 10.1109/TPDS.2019.2942918.

[32] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE SP*, Berkeley, CA, USA, May 2007, pp. 321–334.

[33] T. Zeutschler. (Jun. 30, 2023). *TinyOlap*. [Online]. Available: https://github.com/Zeutschler/tinyolap

[34] S. Haghighi and S. Sabouri. (Jun. 14, 2023). *Art: ASCII Art Library For Python*. [Online]. Available: https://pypi.org/project/art/

[35] PyCA. (Nov. 6, 2022). *Pyca/Cryptography*. [Online]. Available: https://github.com/pyca/cryptography
[36] J. Bethencourt. (May 2006). *Advanced Crypto Software Collection*. [Online]. Available: https://acsc.cs.utexas.edu/cpabe/
[37] *PBC (Pairing-Based Cryptography) Library*. Accessed: Nov. 6, 2023. [Online]. Available: https://crypto.stanford.edu/pbc/
[38] H. Eijs. (May 19, 2023). *Pycryptodomex: Cryptographic Library for Python*. PyPI. [Online]. Available: https://pypi.org/project/pycryptodomex/
[39] H. Eijs. (May 19, 2023). *Pycryptodome: Cryptographic Library for Python*. PyPI. [Online]. Available: https://pypi.org/project/pycryptodome/
[40] (2009). *NumPy*. [Online]. Available: https://numpy.org/
[41] (Dec. 20, 2020). *SymPy*. [Online]. Available: https://www.sympy.org/en/index.html

SOMCHART FUGKEAW (Member, IEEE) received the bachelor's degree in management information systems from Thammasat University, Bangkok, Thailand, the master's degree in computer science from Mahidol University, Thailand, and the Ph.D. degree in electrical engineering and information systems from The University of Tokyo, Japan, in 2017. He is currently an Assistant Professor with the Sirindhorn International Institute of Technology, Thammasat University. He has published more than 60 papers in refereed journals and conferences. His research interests include information security, access control, cloud computing security, big data analysis, and high performance computing. He served as a Reviewer for several international journals, such as IEEE Access, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Cloud Computing, IEEE Transactions on Big Data, IEEE Transactions on Network Science and Engineering, IEEE Transactions on Network and Service Management, IEEE Transactions on Parallel and Distributed Systems, *Computers and Security*, IEEE Systems Journal, and *ACM Transactions on Multimedia Computing, Communications, and Applications*.

LYHOUR HAK received the bachelor's degree in computer engineering from the Sirindhorn International Institute of Technology, Thammasat University, where he is currently pursuing the master's degree in computer engineering. His research interests include network security, access control mechanism, blockchain, and information security.