A SECURE APPROACH TO FINANCIAL DATA MANAGEMENT: A METHOD FOR CONSTRUCTING ENCRYPTED INDEXES TO SUPPORT EFFICIENT QUERYING WITHOUT REVEALING ORDER INFORMATION

Canh Ngoc Hoang[1], Thuy Thu Thi Nguyen[1,*], Danh Kim Le Tran[1], Huy Quang Vu[1]

DOI: http://doi.org/10.57001/huih5804.2024.344

ABSTRACT

Securing digital data in the process of retrieving financial information can be seen as important requirement today. This paper focuses on developing a secure encryption (SE) scheme that supports efficient querying on encrypted data in financial databases. The core of the proposed scheme is the design of a specially encrypted index vector that works as a representative for the digital data. This index has high security, supporting the range query mechanisms without revealing any information about the plaintext data or the query pattern. Additionally, the querying performance of the scheme can be seen as the strong proposed point, as the comparison functions on the encrypted index vectors are designed to be minimalistic and do not return redundant records. Furthermore, the paper also proposes a DAS-Proxy model that allows for the effective and secure deployment of financial databases on any server system (in-house server, cloud server, etc.).

Keywords: *Financial Database, Searchable Encryption, Proxy, Encryption Index Vector* .

[1]Thuongmai University, Vietnam

[*]Email: thuynguyenthithu@tmu.edu.vn

Received: 18/5/2024

Revised: 25/7/2024

Accepted: 28/11/2024

1. INTRODUCTION

In the digital age, data has become the most valuable asset of financial organizations. However, along with the high value of data come with the significant security challenges. The increase in cyber attacks and the increasingly stringent data security regulatory requirements have created an urgent need for advanced security solutions. Searchable Encryption (SE) has emerged as a critical tool that helps organizations and enterprises protect sensitive data while still maintaining the ability to query and analyze the data.

Customer information, transactions, and contracts are data that are frequently accessed to support the management and operation of financial systems. This data is often represented in the form of characters or numbers. Accordingly, all sensitive data will be encrypted before deployment in the database (DB). As a result, the data remains secure against external or internal threats, regardless of whether the DB is deployed on a cloud server or the organization's internal server. However, exploiting and querying encrypted data using standard algorithms like AES and DES [1, 2] is not possible because the encrypted data no longer retains the original data characteristics, such as comparison, ordering, and arithmetic operations.

In recent years, some research has proposed Searchable Encryption (SE) [3-8] that allow direct querying on the index instead of the encrypted data generated by traditional encryption algorithms [1, 2]. However, these solutions still have some issues regarding security, query diversity, performance, and feasibility.

In this paper, we focus on developing an SE scheme based on an encrypted index representing clear numerical data. The proposed index allows concealing the original data information, does not reveal the index order, but still supports a secure mechanism for accurate comparison without the need for decryption. In addition, the paper also proposes the DAS-Proxy model to support the deployment of the proposed SE scheme efficiently.

2. LITERATURE REVIEW

Due to the inherent characteristics of numerical data, most SE (Searchable Encryption) schemes that support queries over encrypted numerical data are designed based on indexes and allow the execution of range queries through these indexes. Some research works such as [3-7, 9, 10] utilize the idea of transforming clear numerical data into encrypted indexes while still supporting comparison mechanisms.

Hacigümüs et al. [4] as well as other studies like [3, 5, 9] propose SE schemes using index-based approaches that partition the clear data into Buckets. Each Bucket is identified by a representative value called the BucketID. The query process on the database is entirely performed on the BucketIDs, and it returns the records containing numerical values belonging to these Buckets. The common drawback of this solution is that the result set may contain redundant records (false positives), which increases the decryption burden on the end-user and the transmission overhead, affecting the overall system performance.

Damiani et al. [11] propose a searchable index based on a one-way hash function. This idea is also used in several other studies [6, 16]. In this approach, each clear numerical value or character is mapped through a one-way hash function to create a representative index. However, due to the deterministic nature of the hash function, the index values are fixed, which only supports equality comparison and is vulnerable to statistical attacks. Additionally, the use of hash functions risks hash value collisions when applied to a large number of records.

Some similar studies [11, 13, 14] use the additional B+ Tree structure to save the index in the storage and create a new concept of Index based on B+ Tree. However, the downside of this solution lies in the number of data communication cycles between the user and the database. When the number of records is large, the number of elements in each Node increases and the height of the tree is also adjusted, leading to an increase in the number of data exchange cycles, which increases the transmission and decoding costs, putting pressure on the network system and user devices.

To eliminate the shortcomings of the B+ Tree index and effectively support range queries on encrypted data, the Order-Preserving Encryption Scheme (OPES) has been introduced by Kadhem et al. [8] as well as in several studies like [10, 15-17]. This scheme requires three stages: "Model", "Flatten", and "Transform" to create the OPES index. However, the OPES index reveals the order of the encrypted values, providing a basis for an adversary to infer information about the underlying plaintext from observing the encrypted records on the server.

Above publications have shown that simultaneously ensuring security, availability, diverse query capabilities, and efficient query performance is a challenge in the problem of querying encrypted data. The goal of building a new encrypted index with high security, strong protection against plaintext information leakage, concealment of order between indexes, and support for an efficient query mechanism can be seen an essential requirement.

3. METHODOLOGY

Using blind indexing to support search on encrypted data is an effective technique applied in many SE (Secure Evaluation) schemes as mentioned above. However, to solve the problem of "secure queries on numerical financial data", we aim to use a special search index representing the numerical data, which allows hiding the order between the indexes, preventing information leakage from the encrypted data, while still supporting comparison mechanisms. The methodology can be summarized as follows:

Firstly, basing on many previous research, we analyze the general solutions for querying on encrypted numerical data and blind index-based querying in particular. Identify the existing issues regarding index construction techniques, index security, index building and storage costs, query efficiency on the index, and the number of data exchange rounds during query execution.

Secondly, the idea of hiding numerical data into vectors and converting numerical comparison expressions into dot products of the corresponding vectors has been used. Continue to add secret parameters and mathematical transformations to make the vectors more secure, called encoded index vectors.

Thirdly, we use mathematical foundations to prove the high complexity of algorithms that can infer information about the numerical data from the index vectors or find the secret parameters used in the index construction and search process.

In addition to the proofs and analysis of the data structures and algorithms related to the index, we also use the standard TPC-H [18] dataset with data sizes from 100,000 to 8 million records to experimentally deploy the

proposed SE scheme on the DAS-Proxy model. Then, we compare and analyze the experimental results with two previous reputable studies to demonstrate the efficiency in terms of query response time.

4. PROPOSE A SCHEME AND IMPLEMENTATION MODEL FOR QUERYING ENCRYPTED DATA

4.1. Problem Statement and Implementation Idea

Problem Statement

- Let the clear relation $R(ID, F_1, F_2,\ldots F_t, \ldots F_n)$ where F_t is the sensitive numeric data field that needs to be protected. Then, R will be replaced by the encrypted relation $R^E(ID, F_1, F_2,\ldots F_t^E, \ldots F_n)$ stored on DB at the Server, and $F_t^E = enc(F_t)$, enc is is a standard data encryption function [1, 2].

- The requirement is to execute a query on the R^E to find the records that satisfy the condition $l \le v_i \le r$, where v_i is the numerical data stored in F_t.

Implementation Idea

- Construct an encrypted search index for the operands v_i and l in the expression "$l \le v_i$. Here, $\overrightarrow{eIv_{field}(v_i)}$ and $\overrightarrow{eIv_{range}(l)}$ are the respective encrypted index vectors representing v_i and l. Similarly, construct the index for the expression $v_i \le r$.

- Construct a searchable index field F_t^{Index}, representative for F_t. This index field contains the encrypted index vector values $\overrightarrow{eIv_{field}(v_i)}$. The updated encrypted relation R^E is then represented as $R^E(ID, F_1, F_2,\ldots F_t^E, F_t^{Index},\ldots F_n)$.

The SQL clear text as follows:

"**Select** $ID, F_1, F_2, \ldots F_t, \ldots F_n$ **From** R **Where** F_t Between l and r" will be transformed to "**Select** $ID, F_1, F_2, \ldots F_t^E, F_t^{Index}, \ldots F_n$ **From** R^E **Where** $ID = Search\left(F_t^{Index}, \overrightarrow{eIv_{range}(l)}, \overrightarrow{eIv_{range}(r)}\right)$"

where Search() is special search function which allows a comparison between $\overrightarrow{eIv_{field}(v_i)}$ in F_t^{Index}, and index vectors $\overrightarrow{eIv_{range}(l)}$ and $\overrightarrow{eIv_{range}(r)}$ without revealing the plaintext information.

4.2. Proposed SE (Searchable Encryption) Scheme to Support Efficient Query on Encrypted Numeric Data

Based on above idea, paper proposes scheme SE including 5 stages as: Gen_SecretMetadata; Build_EncryptionIndexVector; Transform_RangeQueryCondition; SearchOn_ EncryptionIndexVector; and Decrypt_Result. These stages are divided into activities in two areas as the Secure Zone and the Server

Stage 1 "Gen_SecretMetadata"**:** In this stage, the data owner uses a proprietary algorithm to generate and store the MetaData (including keys and secret parameters) in the Secure Zone.

Stage 2 "Build_EncryptionIndexVector": In the Secure Zone, each clear-text value v_i stored in field F_t is transformed into two versions: data encrypted using standard encryption algorithms, and an encryption index vector generated by the Build_EncryptionIndexVector() algorithm. This pair of data is then stored in the F_t^E, F_t^{Index} fields, respectively, in the database on the Server.

The basis for constructing the encrypted index vector is based on the following transformation:

$$l \le v_i \Leftrightarrow v_i - l \ge 0 \Leftrightarrow \binom{v_i}{-1} \cdot \binom{1}{l} \ge 0$$

where the arithmetic comparison between the two operands is transformed into a comparison of the dot product of two vectors with a value of 0. Let $\vec{V} = \binom{v_i}{-1}, \vec{L} = \binom{1}{l}$. The goal is to obscure the information of v_i and l in vector $\vec{V}$ and $\vec{L}$. We propose the following steps:

Step 1. Choose 2 noise vector $\overrightarrow{N_v}$ and $\overrightarrow{N_l}$ that are orthogonal to each other, both these vectors having $k-2$ elements $(k \ge 3)$.

$$\overrightarrow{N_v} = \begin{pmatrix} nv_1 \\ nv_2 \\ \vdots \\ nv_{k-2} \end{pmatrix}, \overrightarrow{N_l} = \begin{pmatrix} nl_1 \\ nl_2 \\ \vdots \\ nl_{k-2} \end{pmatrix} \text{ where } \overrightarrow{N_v}.\ \overrightarrow{N_l} = 0$$

Step 2. To add the noise to $\vec{V}$ and $\vec{L}$, we mix the attributes of $\vec{V}$ into $\overrightarrow{N_v}$ and $\vec{L}$ into $\overrightarrow{N_l}$ at secret positions $(p, q) | 1 \le p, q \le k$. Then vectors $\vec{V}$ and $\vec{L}$ can be represented as follows.

$$\vec{V} = \begin{pmatrix} 0 \\ \vdots \\ \mathbf{v_i} \\ \vdots \\ 0 \\ \vdots \\ \mathbf{-1} \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} nv_1 \\ \vdots \\ \mathbf{0_{(p)}} \\ \vdots \\ nv_2 \\ \vdots \\ \mathbf{0_{(q)}} \\ \vdots \\ nv_{k-2} \end{pmatrix} = \begin{pmatrix} nv_1 \\ \vdots \\ \mathbf{v_i} \\ \vdots \\ nv_2 \\ \vdots \\ \mathbf{-1} \\ \vdots \\ nv_{k-2} \end{pmatrix},$$

$$\vec{L} = \begin{pmatrix} 0 \\ \vdots \\ \mathbf{1} \\ \vdots \\ 0 \\ \vdots \\ \mathbf{l} \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} nl_1 \\ \vdots \\ \mathbf{0_{(p)}} \\ \vdots \\ nl_2 \\ \vdots \\ \mathbf{0_{(q)}} \\ \vdots \\ nl_{k-2} \end{pmatrix} = \begin{pmatrix} nl_1 \\ \vdots \\ \mathbf{1} \\ \vdots \\ nl_2 \\ \vdots \\ \mathbf{l} \\ \vdots \\ nl_{k-2} \end{pmatrix}$$

In this step, we use the secret parameters $\overrightarrow{N_v}, \overrightarrow{N_l}, p, q$ to initially obscure the information of the values $\mathbf{v_i}$ and l. Given the assumption that the positions p, q are kept secret, an attacker could still guess the order of the $\mathbf{v_i}$ values through a simple method. To do this, they can choose an arbitrary constant l_c, determine the corresponding vector $\vec{L}$ for l_c, and calculate all the dot products $\vec{V}.\vec{L}$. The dot product result corresponds to the difference $\mathbf{v_i} - l_c$, which forms the basis for the attacker to find the relationship between the $\mathbf{v_i}$values. To solve this problem, the paper proposes further obfuscating the result of the dot product $\vec{V}.\vec{L}$ as in Step 3.

Step 3. Use a pair of random positive integers (r_v, r_l) to add noise to the elements in the vectors $\vec{V}$ and $\vec{L}$:

$$\vec{V} = r_v * \vec{V} = \begin{pmatrix} r_v * nv_1 \\ \mathbf{r_v} * \mathbf{v_i} \\ r_v * nv_2 \\ \vdots \\ \mathbf{r_v} * (\mathbf{-1}) \\ r_v * nv_{k-2} \end{pmatrix},$$

$$\vec{L} = r_l * \vec{L} = \begin{pmatrix} r_l * l \\ \mathbf{r_l} * \mathbf{1} \\ r_l * l_2 \\ \vdots \\ \mathbf{r_l} * \mathbf{l} \\ r_l * l_{k-2} \end{pmatrix}$$

In this case $\vec{V}.\vec{L} = (r_v * r_l) * (v_i - l) \geq 0$ if $v_i \geq l$. Thus, the result of the dot product of the two vectors $\vec{V}$ and $\vec{L}$ acts as an indicator of whether v_i s greater than l or no, and the result of $\vec{V}.\vec{L}$ does not accurately reflect the exact difference of $v_i - l$.

However, there is a risk of exposing the values of p, q when the attacker analyzes the set of vectors $\vec{V}$ và $\vec{L}$. Specifically, with k elements $(r_v * nv_1, r_v * v_i, r_v * nv_2, \dots, r_v * (-1), r_v * nv_{k-2})$ of the vector $\vec{V}$, the attacker can find the greatest common divisor of r_v and infer the elements with a value of (-1). This allows them to guess the value of p, A similar approach can be used to guess the value of q. To solve this problem, the paper proposes further transforming $\vec{V}$ and $\vec{L}$ into a more secure form called an encoded index vector, as described in Step 4.

Step 4. Use an invertible secret matrix M of size kxk, then create the matrices $M_l = M^T, M_v = M^{-1}$. Perform the following transformations: $\overrightarrow{eiv_{field}(v_i)} = \overrightarrow{M_v \text{ x } V}$ and $\overrightarrow{eiv_{range}(l)} = \overrightarrow{M_l \text{ x } L}$. In this case, the inner product of the index encoding vectors for v_i and l is represented as follows.

$$\overrightarrow{eiv_{field}(v_i)}.\overrightarrow{eiv_{range}(l)} = (M_l \text{ x } L)^T x\ (M_v \text{ x } V)$$
$$= \left(L^T \text{ x } M_l{}^T\right) \text{x } (M_v \text{ x } V) = L^T \text{ x } V$$
$$= \vec{V}.\vec{L} = (r_v * r_l) * (v_i - l)$$

With a set of secret values $\overrightarrow{N_v}, \overrightarrow{N_l}, p, q, r_v, r_l, M$ it is possible to create highly secure index encoding vectors $\overrightarrow{eiv_{field}(v_i)}$ and $\overrightarrow{eiv_{range}(l)}$ This limits the leakage of information about the original values v_i and l but still provides the inner product $\overrightarrow{eiv_{field}(v_i)}.\overrightarrow{eiv_{range}(l)}$ to determine the relationship between v_i and l. For an adversary who gains access to the database (specifically the values in the F_t^{Index}) here will be no mechanism to compare or predict the relationships between the $\overrightarrow{eiv_{field}(v_i)}$ vectors. The Algorithm 1 describes transforming steps of v_i into index encoding vector $\overrightarrow{eiv_{field}(v_i)}$.

Algorithm 1: $\boldsymbol{Transform\_v_i\_To\_\overrightarrow{eiv_{field}}}$

Input: v_i is the value stored in the F_t, which needs to e transformed in to $\overrightarrow{eiv_{field}(v_i)}$; k is the size of the index encoding vector;$\overrightarrow{N_v}$ is a secret noise vector for the vector $\vec{V}$;(p, q) with $1 \leq p, q \leq k$ are the secret positions; r_v is a random positive number;M is a secret invertible matrix with size kxk;

Output: $\overrightarrow{eiv_{field}(v_i)}$ is index encoding vector of v_i;

Implementation steps:

(1) Let $\vec{V} = (v_i, -1)$; Transform $\vec{V} = (0, \dots, v_i, \dots 0, \dots, -1, \dots 0)$ with k elements;

(2) Calculate $\vec{V} = \vec{V} + \overrightarrow{N_v}$;

(3) Calculate $\vec{V} = r_v * \vec{V}$;

(4) Let $M_v = M^{-1}$, calculate $\overrightarrow{eiv_{field}(v_i)} = \overrightarrow{M_v \text{ x } V}$;

(5) Return $\overrightarrow{eiv_{field}(v_i)}$;

Stage 3 "Transform_RangeQueryCondition"**:** in this stage, the range query condition $[l, r]$ on the cleartext data will be transformed into $[\overrightarrow{eiv_{range}(l)}, \overrightarrow{eiv_{range}(r)}]$. This stage is also executed in the secure enclave. Algorithm 2 transforms l, r into the index encoding vectors as follows:

Algorithm 2: $\mathbf{Transform\_l, r\_To\_\overrightarrow{eiv_{field}}}$

Input: l, r are the range query conditions; k is the size of the index encoding vector;

$\overrightarrow{N_l}$ is a secret noise vector for the vectors $\vec{L}, \vec{R}$; (p, q) with $1 \leq p, q \leq k$ are the secret positions; r_l is a

random positive number; M is a secret invertible matrix with size kxk;

Output: $\overrightarrow{\text{eIV}_{\text{range}}(l)}$ và $\overrightarrow{\text{eIV}_{\text{range}}(r)}$;

Implementation steps:

(1) Let $\vec{L} = (1, l)$; Transform $\vec{L} = (0, \dots, 1, \dots 0, \dots, l, \dots 0)$ with k elements; Let $\vec{R} = (1, r)$; Transform $\vec{R} = (0, \dots, 1, \dots 0, \dots, r, \dots 0)$ with k elements;

(2) Calculate $\vec{L} = \vec{L} + \overrightarrow{N_l}$; Calculate $\vec{R} = \vec{R} + \overrightarrow{N_l}$;

(3) Calculate $\vec{L} = r_l * \vec{L}$; Calculate $\vec{R} = r_l * \vec{R}$;

(4) Let $M_l = M^T$; Calculate $\overrightarrow{\text{eIV}_{\text{range}}(l)} = \overrightarrow{M_l \text{ x } L}$ and $\overrightarrow{\text{eIV}_{\text{range}}(r)} = \overrightarrow{M_l \text{ x } R}$;

(5) **Return** $\overrightarrow{\text{eIV}_{\text{range}}(l)}, \overrightarrow{\text{eIV}_{\text{range}}(r)}$;

Stage 4 "SearchOn_ EncryptionIndexVector": this is the stage of directly querying the F_t^{Index} this is the stage of directly querying the $[\overrightarrow{\text{eIV}_{\text{range}}(l)}, \overrightarrow{\text{eIV}_{\text{range}}(r)}]$. This stage is executed entirely on the Server. Specifically, it executes the query: "**Select** ID, F_1, F_2, ... F_t^E, F_t^{Index}, ... F_n **From** R^E **Where** $ID = \text{Search}(F_t^{Index}, \overrightarrow{\text{eIV}_{\text{range}}(l)}, \overrightarrow{\text{eIV}_{\text{range}}(r)})$" . Algorithm 3 describes the simple and efficient operation of the Search() function as follows:

Algorithm 3: Search

Input: $\overrightarrow{\text{eIV}_{\text{field}}(v_i)}$ is F_t^{Index} field value on the record with identifier ID; $\overrightarrow{\text{eIV}_{\text{range}}(l)}, \overrightarrow{\text{eIV}_{\text{range}}(r)}$ are the transformed values of the query range into index encoding vectors;

Output: ID: the identifier of the record containing the value $v_i \in [l, r]$; Or

NULL: the record with ID does not satisfy the query condition;

Implementation:

(1) Calculate $a = \overrightarrow{\text{eIV}_{\text{field}}(v_i)} . \overrightarrow{\text{eIV}_{\text{range}}(l)}$;

(2) Calculate $b = \overrightarrow{\text{eIV}_{\text{field}}(v_i)} . \overrightarrow{\text{eIV}_{\text{range}}(r)}$;

(3) If $(a \geq 0 \text{ and } b \leq 0)$ then

Return ID

Else

Return NULL

Stage 5 "Decrypt_Result": The set of records obtained from Stage 4 will be transferred to the secure enclave to decrypt the data in the F_t^E field and return it to the end-user.

4.3. Proposal of the DAS-Proxy model to support efficient deployment of the proposed SE scheme

The paper proposes to use a secure enclave called Proxy (where services and servers are exclusively controlled by the data owner) in combination with the DAS model [4] to create a new model called DAS-Proxy. The model consists of three components: Client, Proxy, and Server

Client: This is the end-user's device, where the user's search requests are transformed into clear query commands and transmitted to the Proxy.

Proxy: This is a secure enclave that allows the data owner to perform the stages, processes, and functions requiring security properties of the proposed SE scheme, such as Gen_SecretMetadata, Build_EncryptionIndexVector, Transform_RangeQueryCondition, Decrypt_Result.

Server: This is where the encrypted digital data is stored, and the database is installed. This area can be in-house servers or cloud servers, where we do not have absolute guarantee of data security. The process of querying the encrypted data (SearchOn_ EncryptionIndexVector) will be performed in this area, where neither adversaries nor server providers can exploit any information about the plaintext.

5. SECURITY AND PERFORMANCE SYSTEM ANALYSIS

5.1. Security Analysis

Security Analysis of the System through Examining the Potential Information Leakage from $\overrightarrow{\text{eIV}_{\text{field}}(v_i)}$ và $\overrightarrow{\text{eIV}_{\text{range}}(l)}$ (similar for $\overrightarrow{\text{eIV}_{\text{range}}(r)}$). We consider the hypothesis of a known-plaintext attack, where the adversary (denoted as A) knows the set of vectors $(\overrightarrow{\text{eIV}_{\text{field}}(v_i)}, \overrightarrow{\text{eIV}_{\text{range}}(l)})$, The goal of A is to analyze the content of each vector or the relationship between the vectors in order to try to infer information related to the plaintext values v_i, l. Some typical methods that A may use include: (1) Summary the frequency of occurrence of the encoded index vectors, based on the abnormal frequencies of some vectors to infer the corresponding v_i, lvalues. For example, in a company there are only 3 people with a salary of 100 million, while there are only three records containing the same $\overrightarrow{\text{eIV}_{\text{field}}(v_i)}$, A will infer that these 3 records correspond to the 3 employees with the special high salaries. However, during the construction of the index vectors, random parameters

r_v, r_l are introduced, so there will not be any identical index vectors even though they represent the same v_i or l value. Therefore, A's attack based on frequency statistics (for $\overrightarrow{eiv_{field}(v_i)}$) or guessing queries pattern (for $\overrightarrow{eiv_{range}(l)}$) is not feasible. (2) A try to use the dot product $\overrightarrow{eiv_{field}(v_i)}.\overrightarrow{eiv_{field}(v_j)}$ to find the relationship between v_i and v_j is infeasible, because only the result of the dot product $\overrightarrow{eiv_{field}(v_i)}.\overrightarrow{eiv_{range}(v_j)}$ can reflect this relationship. To determine $\overrightarrow{eiv_{range}(v_j)}$, faces a major challenge in having to predict a series of secret parameters stored in the Proxy's secure area, such as $\overrightarrow{N_v}, \overrightarrow{N_l}, p, q, r_v, r_l, M$. Among these parameters, M plays the most important role and is of particular interest to A to predict. A considers finding the corresponding M as finding $M_v = M^{-1}$ in the matrix equation $\overrightarrow{eiv_{field}(v_i)} = \overrightarrow{M_v \times V}$ with the favorable hypothesis for A being that $\vec{V}$ and $\overrightarrow{eiv_{field}(v_i)}$. For each $\overrightarrow{eiv_{field}(v_i)}$ sthere will be a similar equation with k^2 variables (the number of elements of M_v), and each equation will have countless solutions that satisfy it. Therefore, finding the exact M_v from the countless solutions mentioned above is a major challenge for A.

5.2. Experiments and Performance Evaluation

In this paper, we experiment with range queries for three query support solutions implemented on the DAS-Proxy model as follows:

- ***Solution 1***: Encrypt all data using standard algorithms, store on the database at the Server. Perform the query by loading all the encoded data to the Proxy, decrypt and then query on the clear data, and finally return the result to the Client.

- ***Solution 2***: Use the Bucket partitioning method proposed by Hacigümüs et al. [4] to transform the clear data into Bucket Index, then store them on the database at the Server. The querying process on the Server will be performed directly on the Bucket Index, and the result returned to the Client will be further decrypted and queried on the clear data.

- ***Solution 3***: Use the SE scheme built on the encoded index vector proposed in this paper. The querying process is performed only once on the F_t^{Index} field in the database at the Server.

Experimental data: Use the L_EXTENDEDPRICE column data from the LINEITEM table in the TPC-H [18] database. Use record scales from 100 thousand, 500 thousand, 1 million, 2 million, 4 million, and 8 million records. Select 30 query ranges $[l, h]$ for each experiment at each record scale.

Measurement indexes in the experiment: the execution time at different steps in the DAS-Proxy model, including: conversion and decryption at the Proxy (T1); re-querying at the Proxy (T2); data transmission between Proxy and Server (T3); query execution at the Server (T4).

The results can be seen in Figure 1.

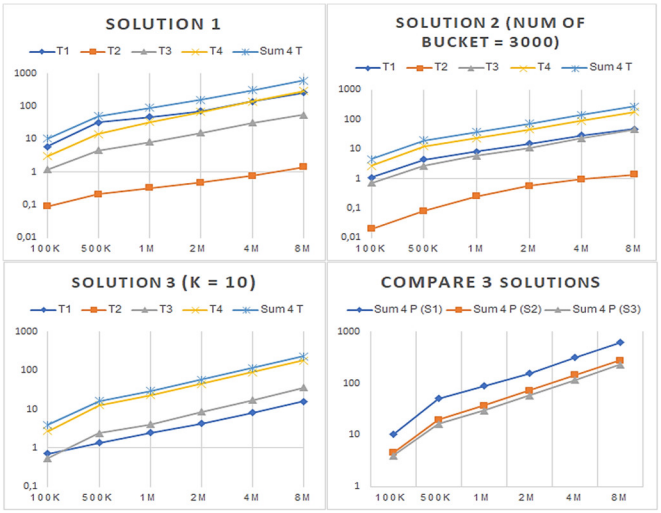

Fig. 1. Evaluate the executive time for 3 query solutions

Fig. 1 shows that the experimental results are completely consistent with the theoretical evaluations. In solution 1, also known as the naive query method, is evaluated as the simplest but worst in terms of performance and feasibility. In this solution, the pressure of loading the entire database code to the Proxy causes the T3 line to rise abnormally higher than Solutions 2 and 3. However, executing the query to return all data on the Server and decrypting all this data on the Proxy causes the memory cost on the Server to rise significantly and increases the computation cost on the Proxy more than all other solutions. This makes the T1 and T4 lines unusually high. In addition, the re-query on the entire dataset also contributes to making T2 higher than the other solutions. Solution 2 allows direct queries on the Bucket Index, helping to reduce some of the records that do not meet the range query conditions on the Server. However, since each Bucket Index represents multiple clear values, the returned result set from the Server contains false positive records (the more Buckets, the lower the false positive rate). Therefore, the T1, T2, T3, T4 lines of this solution are always much lower than Solution 1. Solution 3 supports queries on the encoded vector index field, helping the result set after querying on the

Server to have no false positive records. Therefore, it minimizes the transmission cost and does not need to decode the redundant records or perform re-queries. The T2 line will not exist and the T1, T3 lines will be much lower than Solution 2. On the Server, Solution 3 has the advantage of lower memory cost than Solution 2 due to no redundant records returned. However, Solution 2 has the advantage of easier comparison on the Bucket Index than the dot product operations of Solution 3. Therefore, for the T4 line, we note the similarity in time cost for both Solutions 2 and 3.

From Fig. 1, the comparison of the total execution query time shown, the proposed SE scheme has a superior advantage in terms of performance and execution cost. The essence of the improvement in time cost is that the value of v_i on each record has been represented by a single vector $\overrightarrow{\text{eIV}_{\text{field}}(v_i)}$ and the comparison expression of $l \le v_i \le r$ has been transformed into dot products $\overrightarrow{\text{eIV}_{\text{field}}(v_i)} \cdot \overrightarrow{\text{eIV}_{\text{range}}(l)} \ge 0$ and $\overrightarrow{\text{eIV}_{\text{field}}(v_i)} \cdot \overrightarrow{\text{eIV}_{\text{range}}(r)} \le 0$ with low implementation cost. The result of the dot product accurately reflects the relationship between the value v_i and the query range $[l, r]$. In other words, the query result returned on the Server does not contain any false positive records, which leads to a reduction in the transmission cost from the Server to the Proxy (T3) and eliminates the need for decoding and re-querying at the Proxy (T1, T2) as in previous solutions.

6. CONCLUSION

In this paper, we focus on constructing a special SE (Searchable Encryption) scheme and proposing its deployment on the DAS-Proxy model to effectively address range query requirements on encrypted data in the server's database. The core proposal is a coded index vector structure as a replacement for plaintext data to be stored and support searching on the server. This new structure allows for better concealment of the original information, without revealing the order as in previous OPES or Bucket Index solutions. However, the coded index vector still supports a secure comparison mechanism on the server and does not return redundant data after the query. The security and execution efficiency of the proposed SE scheme are also clarified in the paper through experiments and comparisons with two other popular solutions for querying encrypted data. The results of this research enable the integration of efficient range query functionality on encrypted data into SE (Searchable Encryption) systems. This not only enhances the security of information access in financial systems in particular but can also be extended to apply to any other encrypted data query problem, such as in the fields of healthcare, education, or national population databases.

REFERENCES

[1]. M. J. Dworkin, *Advanced Encryption Standard (AES)*. National Institute of Standards and Technology (U.S.), Gaithersburg, MD, NIST FIPS 197-upd1, May 2023. doi: 10.6028/NIST.FIPS.197-upd1.

[2]. A. Biryukov, C. De Cannière, "Data encryption standard (DES)," in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg, Ed., Boston, MA: Springer US, 129-135, 2005. doi: 10.1007/0-387-23483-7_94.

[3]. E. J. Goh, *Secure Indexes*. 2003. Accessed: Apr. 19, 2024. [Online]. Available: https://eprint.iacr.org/2003/216

[4]. H. Hacigümüs, B. Iyer, C. Li, S. Mehrotra, *Executing SQL over Encrypted Data in the Database-Service-Provider Model*. ACMSIGMOD, Madison, Wisconsin, USA, 2002. doi: 10.1145/564691.564717.

[5]. R. Handa, R. Challa, N. Aggarwal, "An efficient approach for secure information retrieval on cloud," *Journal of Intelligent & Fuzzy Systems*, 34, 1345-1353, 2018. doi: 10.3233/JIFS-169430.

[6]. C. Wang, N. Cao, J. Li, K. Ren, W. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data," in *2010 IEEE 30th International Conference on Distributed Computing Systems*, 253-262, 2010. doi: 10.1109/ICDCS.2010.34.

[7]. L. Liu, J. Gai, "Bloom Filter Based Index for Query over Encrypted Character Strings in Database," in *2009 WRI World Congress on Computer Science and Information Engineering*, 303-307, 2009. doi: 10.1109/CSIE.2009.979.

[8]. H. Kadhem, T. Amagasa, H. Kitagawa, "Optimization Techniques for Range Queries in the Multivalued-partial Order Preserving Encryption Scheme," in *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, A. Fred, J. L. G. Dietz, K. Liu, and J. Filipe, Eds., Berlin, Heidelberg: Springer, 338-353, 2013. doi: 10.1007/978-3-642-29764-9_23.

[9]. P. Karras, A. Nikitin, M. Saad, R. Bhatt, D. Antyukhov, S. Idreos, "Adaptive Indexing over Encrypted Numeric Data," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, Accessed: Apr. 19, 2024. [Online]. Available: https://www.academia.edu/71586868/Adaptive_Indexing_over_Encrypted_Numeric_Data

[10]. K. Li, W. Zhang, C. Yang, N. Yu, "Security Analysis on One-to-Many Order Preserving Encryption-Based Cloud Data Search," *IEEE Transactions on Information Forensics and Security*, 10, 9, 1918-1926, 2015, doi: 10.1109/TIFS.2015.2435697.

[11]. E. Damiani, S. D. C. Vimercati, S. Jajodia, S. Paraboschi, P. Samarati, "Balancing confidentiality and efficiency in untrusted relational DBMSs," in

Proceedings of the 10th ACM conference on Computer and communications security, in CCS '03. New York, NY, USA: Association for Computing Machinery, 93-102, 2003. doi: 10.1145/948109.948124.

[12]. J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, W. Lou, "Fuzzy Keyword Search over Encrypted Data in Cloud Computing," in *2010 Proceedings IEEE INFOCOM*, 1-5, 2010. doi: 10.1109/INFCOM.2010.5462196.

[13]. C. Ho, K. Pak, S. Pak, M. Pak, C. Hwang, "A Study on Improving the Performance of Encrypted Database Retrieval Using External Indexing System of B+ Tree Structure," *Procedia Computer Science*, 154, 706-714, 2019. doi: 10.1016/j.procs.2019.06.110.

[14]. Z. F. Wang, A. G. Tang, W. Wang, "Fast query over encrypted data based on B+ tree," in *2009 International Conference on Apperceiving Computing and Intelligence Analysis*, 132-135, 2009. doi: 10.1109/ICACIA.2009.5361133.

[15]. S. Lee, T. J. Park, D. Lee, T. Nam, S. Kim, "Chaotic Order Preserving Encryption for Efficient and Secure Queries on Databases," *IEICE Transactions*, 92-D, 2207-2217, 2009, doi: 10.1587/transinf.E92.D.2207.

[16]. A. Boldyreva, N. Chenette, A. O'Neill, "Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions," in *Advances in Cryptology - CRYPTO 2011*, P. Rogaway, Ed., Berlin, Heidelberg: Springer, 578-595, 2011. doi: 10.1007/978-3-642-22792-9_33.

[17]. A. Boldyreva, N. Chenette, Y. Lee, A. O'Neill, "Order-Preserving Symmetric Encryption," in *Advances in Cryptology - EUROCRYPT 2009*, A. Joux, Ed., Berlin, Heidelberg: Springer, 224-241, 2009. doi: 10.1007/978-3-642-01001-9_13.

[18]. https://tpc.org/TPC_Documents_Current_Versions/pdf/tpc-h_v3.0.0.pdf. Accessed: Apr. 19, 2024.