The Journal of Supercomputing (2022) 78:15749–15785
https://doi.org/10.1007/s11227-022-04467-7

New attacks on secret sharing-based data outsourcing: toward a resistant scheme

Peyman Rahmani[1] · **Seyed Mostafa Fakhrahmad**[1] · **Mohammad Taheri**[1]

Accepted: 18 March 2022 / Published online: 25 April 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

In this paper, two new practical attacks on some secret sharing-based data outsourcing schemes are first introduced, and several other security and performance issues with the existing schemes are also explored. The existing and new attacks exploit the information about the share range boundaries or the correspondences between the secret values and shares. A range expansion technique is then proposed to thwart one of the attacks. It expands the ranges in every range predicate in the submitted queries in order to hide the share range boundaries from any query observer. Next, a mapping method is proposed to thwart the other attacks. It maps each secret value to a mapping value using a secret one-to-many mapping with a finite set of linear mapping rules so that the tuples of shares are generated from the mapping values rather than directly from the secret values. The proposed mapping method works as an additional layer of security and addresses any attack based on the correspondences between the secret values and shares. At the same time, it preserves the homomorphism property of secret sharing. Finally, a new secure data outsourcing scheme is elaborated on secret sharing, the proposed mapping method, and the proposed range expansion technique. The proposed scheme is resistant to various attacks and also some inferences. It supports the fully server-side or a partially server-side query execution of most types of queries. The experimental results confirm that the proposed scheme is quite practical and efficient.

Keywords Data outsourcing · Data confidentiality · Secret sharing · Query processing · Additive homomorphism · Searchability

✉ Mohammad Taheri
motaheri@shirazu.ac.ir

[1] Department of Computer Science and Engineering, Shiraz University, Shiraz, Iran

1 Introduction

Data outsourcing, or database as a service (DBaaS), is one of the most critical services provided by cloud computing in which a database owner can store his/her database on one or multiple database service providers, allowing him/her to access it at any time and from anywhere. Ensuring the confidentiality of outsourced data is a crucial aspect of such a service. In the field of data outsourcing security, there are three types of confidentiality: content confidentiality, access confidentiality, and pattern confidentiality [1], of which the first one is the most important. Data encryption, data fragmentation, and secret sharing are three popular approaches to providing content confidentiality of outsourced data. Some studies [2, 3] provide not only confidentiality of outsourced data but also data integrity verification. However, protection of outsourced data using a privacy-preserving method may bring some challenges in the management of and access to the sensitive outsourced data, such as secure data deduplication [4], privacy-preserving machine learning and data mining [5–8], and keyword searchability over encrypted data [9–12].

Data encryption is the most popular approach to providing content confidentiality of outsourced data. The problem with traditional encryption methods is that most types of queries cannot be executed on the encrypted data at the server side, where the servers that host data are untrusted and should not acquire the decryption key. To address this problem, some schemes [13–15] store some metadata, such as indices, at the server side. The server-side metadata makes such schemes susceptible to some inferences. Furthermore, in some of these schemes, query results returned by the data server usually contain some false hits. Although false hits can be eliminated for exact-match and range queries at the client side, this is not the case for aggregate queries and delete and update statements. That is, a scheme with the problem of false hits does not support the server-side execution of aggregate queries and delete and update statements with a predicate on a sensitive attribute. There are some specialized encryption schemes available in the literature to support efficient server-side query execution of certain types of queries. Order-preserving encryption (OPE) schemes [16–18] were proposed to efficiently evaluate range predicates on ciphertexts. Partially homomorphic encryption schemes [19–21] can support only a limited number of operations on ciphertexts. On the other hand, fully homomorphic encryption schemes [22, 23], which support most types of operations on ciphertexts, are impractical because of their inefficient computations [24].

A data fragmentation scheme [25] splits a relation into several fragments and then distributes the fragments over distinct data servers. Where the data servers could communicate and collude with each other, data fragmentation schemes are insecure. Furthermore, the lack of straightforward and efficient methods for processing queries in which different fragments are involved limits its application.

Secret sharing as a computationally efficient security approach can provide confidentiality, integrity, and availability with many applications to cloud computing. A secret sharing scheme splits a secret into some shares and distributes

the shares to multiple shareholders such that the secret can be recovered only by the collaboration of some predefined authorized subsets of the shareholders. Some applications of secret sharing include secure multi-party computation (e.g., federated learning [26] and matrix multiplication [27]), threshold cryptography [28], e-voting [29, 30], and privacy-preserving data storage [31]. To generate shares from a secret value by Shamir's threshold secret sharing scheme [32], a random polynomial with a y-intercept of that secret value is first constructed. The shares are then obtained as the vertical coordinates of some points on the polynomial. By having a sufficient number of shares, one can reconstruct the polynomial and subsequently recover the secret value. The additive homomorphism property of Shamir's secret sharing scheme makes it suitable for the application of secure database outsourcing. To the best of our knowledge, all the existing secret sharing-based data outsourcing schemes are based on Shamir's secret sharing scheme. In what follows, we refer to Secret Sharing-based Data Outsourcing as SSDO. Note that when the term SSDO (in general) is used, it refers to polynomial-based threshold SSDO. However, the original Shamir's secret sharing scheme generates random shares, which are non-searchable. There are two main approaches to supporting the evaluation of exact-match and range predicates on the shares of a sensitive attribute directly. One approach is to utilize server-side metadata, and the other is to impose restrictions on the set of possible generated shares from each original value in the domain of the sensitive attribute on each data server such that the associated sets with different original values on each data server are non-overlapping. In what follows, schemes based on the first and second approaches are referred to as metadata-based SSDO and restriction-based SSDO schemes, respectively.

The SSDO schemes proposed in [33, 34] store indices on the protected sensitive attributes on an index server. To obtain the result of a query, the client needs to interact with the index server as well as with the data servers several times. The SSDO scheme proposed in [35] stores private indices on the searchable attribute at the server side in order to utilize them in the server-side evaluation of exact-match and range predicates. This scheme suffers from the problem of false hits. The model proposed for non-communicating data servers in [36] stores OPE ciphertexts of secret values along with two tuples of shares on the data servers. An adversary could infer some information about the sensitive outsourced data from the OPE ciphertexts. In general, in metadata-based SSDO schemes, an adversary could infer some information from the server-side metadata.

In the restriction-based SSDO studies in [37–39], to construct a so-called *distribution polynomial* for a secret value of a sensitive attribute, the set of coefficients of the polynomial is computed based on the secret value deterministically. That is, for a given secret value (in any tuple of the relation), a unique set of coefficients is selected, and therefore, a unique tuple of shares is generated. In other words, the mapping of secret values to tuples of shares is one-to-one. As will be discussed later, there are some security issues with such a one-to-one mapping. In the studies in [40, 41], a range of coefficients is assigned to each original value in the domain of a sensitive attribute. Then, the coefficient of the distribution line that is constructed to generate shares from each secret value is randomly selected

from the range of coefficients assigned to that value. As a result, there is a predetermined range for the possible generated shares from each original value in the domain of the sensitive attribute. In what follows, the SSDO schemes proposed by Emekci et al. [39] and Hadavi et al. [41] are referred to as EMAA and HJEC, respectively, which are their authors' initials. Also, the SSDO scheme proposed by Ghasemi in [42], which improves the security of HJEC based on a Fake Tuple Insertion method, is referred to as FTI.

In this paper, two new practical attacks on HJEC and some similar SSDO schemes are first introduced, and several other security and performance issues with the existing SSDO schemes are explored. The existing and new attacks exploit the information about the share range boundaries or the correspondences between the secret values and shares. A technique called *range expansion* is then proposed to thwart the first introduced attack. The proposed range expansion technique expands the ranges in every range predicate in the submitted queries in order to hide the share range boundaries of the original values from any query observer. Next, a mapping method is proposed to thwart the other attacks, including the second introduced attack and two existing attacks. The proposed mapping method maps each secret value to a mapping value using a secret one-to-many mapping with a finite set of linear mapping rules so that the tuples of shares are generated from the mapping values rather than directly from the secret values. The proposed mapping method works as an additional layer of security and addresses any attack based on the correspondences between the secret values and shares. At the same time, it preserves the homomorphism property of secret sharing. A server-side-partial-aggregation mechanism is designed to be used with the proposed mapping method for processing aggregate queries. Finally, a new secure data outsourcing scheme is elaborated on secret sharing, the proposed mapping method, and the proposed range expansion technique. The proposed scheme is resistant to various attacks and also some inferences. It supports the fully server-side or a partially server-side query execution of most types of queries.

The rest of this paper is organized as follows. In Sect. 2, the essential background of the field is given, and some issues with the existing SSDO schemes are explored. In Sect. 3, the two new attacks are first introduced. The proposed range expansion technique, the proposed mapping method, and the proposed secure data outsourcing scheme are then presented. Next, the security of the proposed secure data outsourcing scheme is analyzed. Section 4 discusses the query processing in the proposed scheme. Section 5 is devoted to the evaluation of the proposed scheme with some experimental results and discussions. Finally, Sect. 6 concludes the paper.

2 Preliminary discussions and related work

In this section, the preliminaries are explained in Sects. 2.1, 2.2, and 2.3. Then, EMAA, HJEC, FTI, and two existing attacks on SSDO schemes are reviewed in Sects. 2.4, 2.5, 2.6, and 2.7, respectively.

2.1 Shamir's threshold secret sharing scheme

A (K,N)-threshold secret sharing scheme ($K \leq N$) shares a secret among N shareholders such that every K shareholders can recover the secret while any $K-1$ or fewer shareholders cannot. Shamir's threshold secret sharing scheme [32] employs random polynomials to generate random shares. Shamir's scheme with a (K,N)-threshold generates a set of shares from a given secret value v in the ith data item as follows. A random polynomial of degree $K-1$ of the form $P_i(x) = a_{i,K-1}x^{K-1} + a_{i,K-2}x^{K-2} + \cdots + a_{\mathrm{i},1}x + v$ is first constructed by considering a set of random coefficients $a_{i,K-1}$, $a_{i,K-2}$,…, and $a_{\mathrm{i},1}$. Then, N distinct horizontal coordinates x_1, x_2,…, and x_N (in the original paper, $x_n = n$ for $1 \leq n \leq N$) are used to generate N shares as the vector $\vec{s}_i = \left[s_{\mathrm{i},1}, s_{\mathrm{i},2}, \ldots, s_{i,N}\right]^{\mathrm{T}}$, where $s_{i,n} = P_i\left(x_n\right)$ for $1 \leq n \leq N$. Finally, each share is delivered to the corresponding shareholder. Anyone in possession of any K distinct points on a polynomial of degree $K-1$ can reconstruct the polynomial by any linear regressor (e.g., Lagrange interpolation or Gauss elimination method) and then obtain the secret value by evaluating the polynomial at 0. The scheme can use modular arithmetic rather than ordinary arithmetic.

2.2 Models and terminologies

The data outsourcing models considered in this research are as follows. A database owner outsources his/her relational database in which the values of some attributes are confidential to one or multiple *honest-but-curious* data servers. The database owner or any authorized client can submit queries and data manipulation statements over the database to the data servers. An honest-but-curious data server executes every query received from the authorized clients honestly but is interested in extracting knowledge from the confidential data. Two models are considered, referred to as single-server and multi-server models. In the single-server model, the shares are all stored on a single data server. In contrast, in the multi-server model, shares are distributed across multiple data servers where the data servers could communicate and collude with each other. In our discussions, an *adversary* may be any data server or anyone who has access to the stored data or to the communication channel.

When the processing of queries and/or data manipulation statements is considered, the term "queries" is used instead of "queries and/or data manipulation statements." For a particular query, the query set refers to the set of tuples involved in that query (i.e., the set of tuples that satisfy the predicate of that query). An attribute is considered to be searchable if there is a need to evaluate exact-match and range predicates on that attribute or to carry out grouping operations on that attribute.

In order to ensure content confidentiality of non-searchable attributes, the original Shamir's secret sharing scheme can be utilized, while for searchable attributes, a mechanism should be designed to evaluate predicates at the server side efficiently. In all SSDO schemes, including the scheme proposed in this paper, for each sensitive attribute, a constant vector of horizontal coordinates (referred to as *distribution vector*) is selected to generate shares from every secret value of

that attribute. With the use of a constant distribution vector, the additive homomorphism property is satisfied, and the need to store separate vectors of horizontal coordinates is also eliminated. The distribution vector is kept secret and shared with only authorized clients. As a result, even if the data servers stack their shares together, they cannot recover the secret values. Since modular arithmetic violates the restrictions, all restriction-based SSDO schemes (including the proposed scheme in this paper) work only with ordinary arithmetic.

In a $(2,N)$-threshold SSDO scheme, the term *a pair of shares* refers to an ordered pair of shares generated from a secret value (i.e., the vertical coordinates of the two points on a specific line constructed for a secret value with any two horizontal coordinates from the distribution vector). Also, in a (K,N)-threshold SSDO scheme with any value of K in general, the term *a tuple of shares* refers to a K-tuple of shares generated from a secret value with K out of N distinct horizontal coordinates from the distribution vector. In a restriction-based SSDO scheme, the set of possible generated shares from each original value v on the data server DS_n $(1 \leq n \leq N)$ is referred to as the share range of v on DS_n.

Shamir's secret sharing scheme and SSDO schemes with a constant distribution vector are $(+,+)$-homomorphic. This means that in such schemes, for any two secret values v and v', the secret value $v+v'$ can be recovered from the virtual shares obtained by the sum of the corresponding shares generated from v and v' (with the same distribution vector); i.e., we have:

$$v + v' = \Psi_K^N\big(\vec{s}(v)\big) + \Psi_K^N\big(\vec{s}\big(v'\big)\big) = \Psi_K^N\big(\vec{s}(v) + \vec{s}\big(v'\big)\big), \tag{1}$$

where $\Psi_K^N\big(\vec{s}(v)\big)$ denotes the secret recovery function using K out of N shares generated from v represented by the vector $\vec{s}(v)$. It can be easily shown that the $(+,+)$-homomorphism property implies the linear property [36], which means that for every value v and constants p and q, we have:

$$v \times p + q = \Psi_K^N\big(\vec{s}(v)\big) \times p + q = \Psi_K^N\big(\vec{s}(v) \times p + q\big). \tag{2}$$

It should be noted that $(+,+)$-homomorphism is not the only form of additive homomorphism but is the only form of additive homomorphism supported by Shamir's secret sharing scheme and SSDO schemes. The additive homomorphism property is utilized to execute linear updates and the aggregate functions SUM and AVG on shares directly.

In a data outsourcing scheme, it is said that a query is executed *partially server-side* if after processing that query at the server side, the returned result should be processed at the client side. A "secret sharing"/"encryption" scheme is order preserving if the order of the "secret values"/"plaintexts" is preserved in their corresponding "shares or metadata"/"ciphertexts".

It is assumed that the domain of the sensitive attribute to be protected is a finite subset of the integer numbers. The domain of an attribute of any other data type is converted to a finite subset of the integer type before outsourcing. In what follows, the notation $[l \ldots u]$ indicates the set of integer numbers in the range l to u.

2.3 Inferences from outsourced data

In this study, four types of inferences are defined as follows:

(1) *Identicalness inference* In a metadata-based or restriction-based SSDO scheme, if the same tuple of shares (with a constant distribution vector) appears in some different tuples of a relation, an observer adversary can infer that such tuples belong to the same secret value. Also, in a metadata-based SSDO scheme, if the same metadata appears in some different tuples of a relation, an observer adversary can infer that such tuples belong to the same secret value.
(2) *Equality inference* In a restriction-based SSDO scheme, if all the original values in the domain of a sensitive attribute have share ranges of the same size on a data server, based on the domain of the sensitive attribute and the domain of shares, an adversary can compute the share range boundaries. Then, he or she can infer the equality of the secret values corresponding to distinct shares that belong to the same share range. In the worst case, with an order-preserving share range assignment, the correspondences between the original values and share ranges are exposed. As a result, the adversary can discover the secret value corresponding to every stored share.
(3) *Ordering inference* In an order-preserving metadata-based/restriction-based SSDO scheme, the order of the secret values of a sensitive attribute is preserved in their corresponding metadata/shares. In such a scheme, the order of the unknown secret values of every sensitive attribute in the tuples of the relation is exposed to anyone who observes the metadata/shares.
(4) *Distribution inference* In a metadata-based/restriction-based SSDO scheme, if the distribution of the secret values of a sensitive attribute is reflected in the distribution of their corresponding metadata/shares, an adversary who has prior knowledge about the distribution of the secret values may be able to guess some of the correspondences between the secret values and metadata/shares.

The original Shamir's secret sharing scheme is resistant to all the above types of inferences, but the existing metadata-based and restriction-based SSDO schemes are vulnerable to some of them. In metadata-based SSDO schemes [35, 36], the server-side metadata is the source of inference. It should be noted that most encryption-based schemes are also vulnerable to such types of inferences. For example, deterministic (one-to-one) encryption schemes are vulnerable to identicalness and distribution inferences, and OPE schemes are vulnerable to ordering inferences. Thus, the scheme in [36], which stores the OPE ciphertexts of the secret values at the server side, is vulnerable to identicalness, ordering, and distribution inferences. Vulnerabilities of the restriction-based SSDO schemes proposed in [39, 41] to some inferences will be explored in Sects. 2.4 and 2.5, respectively.

2.4 The EMAA scheme

The EMAA scheme [39] is based on Shamir's secret sharing scheme with a (K, N) threshold. It chooses $K-1$ coefficient domains, each one associated with one of the coefficients used to construct distribution polynomials of degree $K-1$. Let V and $|V|$ be the domain of the sensitive attribute to be protected and its size, respectively. Each coefficient domain is divided into $|V|$ equal-size partitions. Each original value in V is assigned $K-1$ coefficient partitions from the $K-1$ coefficient domains (each coefficient partition from its corresponding coefficient domain) in an order-preserving manner. A tuple of shares is generated from a given secret value v of the sensitive attribute (in any tuple of the relation) as follows. The distribution polynomial $P_v(x) = a_{v,K-1}x^{K-1} + a_{v,K-2}x^{K-2} + \cdots + a_{v,1}x + v$ is first constructed, where each coefficient $a_{v,k}$ $(1 \leq k \leq K-1)$ is computed using a hash function to map v to a value within the assigned coefficient partition. The same as Shamir's secret sharing scheme, N shares are then generated and stored on the data servers, where nth share $s_n(v)$ is equal to $P_v(x_n)$, and $\vec{x} = \left[x_1, x_2, \ldots, x_N\right]^{\mathrm{T}}$ is a predetermined distribution vector. A client who knows the distribution vector and receives at least K out of N shares can recover the secret value through any linear interpolation.

EMAA is easy to implement and supports server-side query execution of most types of queries. However, it has several drawbacks. In particular, since the mapping of secret values to sets of coefficients is done by using predetermined and fixed hash functions, a unique tuple of shares can be generated from each secret value. Similar to any one-to-one mapping scheme, it is vulnerable to identicalness and distribution inferences. Furthermore, this scheme is vulnerable to ordering inferences and also to the attack introduced by Dautrich and Ravishankar [43].

2.5 The HJEC scheme

The HJEC scheme [41] is based on Shamir's secret sharing scheme with the $(2, 2)$ threshold. Let T be the relation to be outsourced containing the searchable attribute O to be protected with the domain V. For the attribute O, a coefficient domain C is selected and divided into more than or equal to $|V|$ coefficient partitions. Also, a distribution vector $\vec{x} = \left[x_1, x_2\right]^{\mathrm{T}}$ is selected. Each value v from V is assigned one or multiple coefficient partitions from C, denoted by $C_v = \bigcup_{m=1}^{M_v} C_v^m$ (M_v denotes the number of coefficient partitions assigned to v). Suppose that the tuple T_i $(1 \leq i \leq |T|)$ should be outsourced in which the secret value of the attribute O is v. A pair of shares denoted by $(s_{i,1}, s_{i,2})$ is generated from v as follows. A random coefficient (slope) a_i is first selected from C_v. Then, the distribution line $P_i(x) = a_i x + v$ is constructed. Finally, $s_{i,1} = P_i(x_1)$ and $s_{i,2} = P_i(x_2)$ are computed and shared with DS_1 and DS_2, respectively. Clearly, for each coefficient partition $C_v^m = \left[\alpha_v^m, \beta_v^m\right]$, there is a bounded range of possible generated shares from v on each data server DS_n $(n \in \{1, 2\})$, denoted by $S_{v,n}^m = \left[d_{v,n}^m, e_{v,n}^m\right]$ (i.e., $d_{v,n}^m = \alpha_v^m x_n + v$ and $e_{v,n}^m = \beta_v^m x_n + v$). The randomness property of the coefficient selection results in the randomness of the generated pairs of shares. It provides a degree of resistance to identicalness

inferences. However, in order to have searchability over shares, each exact-match predicate on an original sensitive attribute is translated into one or multiple pairs of range predicates on the share columns associated with the original attribute.

HJEC suggests three partitioning methods to partition the coefficient domain: simple partitioning, weighted partitioning, and secure partitioning. Also, a coefficient partition assignment function is defined to assign one or (in the secure partitioning method) multiple coefficient partitions to each original value in the domain of the sensitive attribute to be protected. This assignment function can be either order preserving or order obfuscating. With an order-preserving coefficient partition assignment, the order of the original values is preserved in their assigned coefficients and, in turn, in their corresponding share ranges. In contrast, with an order-obfuscating coefficient partition assignment, the order of the original values is preserved in neither their assigned coefficients nor their corresponding share ranges.

In the simple partitioning method, C is divided into $|V|$ equal-size partitions. Then, a coefficient partition assignment function, which can be either order preserving or order obfuscating, is defined to assign one coefficient partition to each value from V. The simple partitioning method is vulnerable to equality and distribution inferences. In the weighted partitioning method, each value from V is assigned a partition from C with a size proportional to the relative frequency of that value in the sensitive attribute. The coefficient partition assignment function defined with the weighted partitioning method can also be either order preserving or order obfuscating. Both the simple and weighted partitioning methods with an order-preserving assignment are vulnerable to ordering inferences. In the secure partitioning method, C is divided into some equal-size partitions such that the number of coefficient partitions is several times greater than $|V|$. Then, a coefficient partition assignment function is defined by which the number of coefficient partitions assigned to each value from V is proportional to the relative frequency of that value in the sensitive attribute. For the secure partitioning method, the coefficient partition assignment should be order obfuscating.

Every query with predicate(s) and/or aggregate function(s) on a sensitive attribute should be translated, and then, the translated queries are submitted to the data servers. For example, suppose that an authorized client wants to issue a query with a predicate of the form $O = v$, where O is a searchable attribute, $C_v^1 = \left[\alpha_v^1, \beta_v^1\right]$ is the only coefficient partition assigned to v, and $\vec{x} = \left[x_1, x_2\right]^{\mathrm{T}}$ is the distribution vector for O. For the sake of brevity, in what follows, the only coefficient partition assigned to v is denoted by $C_v = \left[\alpha_v, \beta_v\right]$. A query of the form Query 1 is translated into two queries of the form Query 2 for $n \in \{1, 2\}$, where O_n^* denotes the share column associated with O on DS_n. Then, each of the translated queries is submitted to its corresponding data server DS_n. Recovering the secret values at the client side is done through any linear interpolation.

Query 1

SELECT $*$
FROM T
WHERE $O = v$

Query 2

SELECT $*$
FROM T
WHERE $\alpha_v x_n + v \leq O_n^* \leq \beta_v x_n + v$

A problem with HJEC is that the order-obfuscating assignment technique of HJEC (for all three partitioning methods) is not applicable. The reason is that with an order-obfuscating assignment, there will usually be some overlapping share ranges among the share ranges of all the original values in the domain of a protected sensitive attribute. Overlapping share ranges could lead to false hits in the result of queries with a predicate on the sensitive attribute. As a result, aggregate queries and delete and update statements in which there is a predicate on a sensitive attribute cannot be executed on the shares at the server side. Furthermore, HJEC is vulnerable to all the attacks (including the two attacks introduced in this paper and two other attacks introduced in [42, 43]) regardless of the adopted partitioning method and coefficient partition assignment function.

2.6 The FTI scheme

In [42], after introducing an attack on some $(2, N)$-threshold SSDO schemes such as HJEC, Ghasemi presented a method to address that attack as follows. Without loss of generality, suppose a relation with one sensitive attribute should be outsourced to a single data server. Any $(2, N)$-threshold SSDO scheme such as HJEC can be used to generate N share columns from the sensitive attribute. Some fake tuples with random shares for the sensitive attribute and random values for the other attributes are inserted into the relation. An auxiliary column *Tag* is added to the relation. The tag value (in the column *Tag*) for each fake/actual tuple is filled with a value obtained by encrypting $0/1$ using the encryption function of the Paillier cryptosystem [19], which is a one-to-many homomorphic encryption scheme. A sufficient number of fake tuples should be inserted to thwart the attack.

Ghasemi claimed that his proposed scheme utilizes the homomorphism property of the Paillier cryptosystem to perform aggregate queries at the server side. Consider a non-sensitive attribute Q stored in plaintext. Ghasemi assumed that the sum of the values of Q in the actual tuples of a query set with any predicate can be obtained by decrypting the sum of $Q \times Tag$ for all the tuples of that query set using the decryption function of the Paillier cryptosystem. As the Paillier cryptosystem is $(\times, +)$-homomorphic, this assumption is incorrect. Consequently, the methods used by FTI to perform the aggregate functions COUNT, SUM, and AVG do not work correctly. According to the $(\times, +)$-homomorphism properties of the Paillier cryptosystem, to perform the aggregate function SUM on the attribute Q by FTI, the data server should compute the multiplication of the values Tag^Q in all the tuples in the query set (with modular arithmetic). Then, the client can decrypt the result using the decryption function of the Paillier cryptosystem to obtain the sum of the plaintext values of Q in the actual tuples in the query set. The methods to perform the aggregate functions COUNT and AVG are similar to that to perform the aggregate function SUM. There are several serious issues

with FTI. The most important ones are as follows. First, this scheme does not address the other vulnerabilities of its previous schemes. In particular, it is vulnerable to the two attacks introduced in this paper and the attack introduced in [43]. Second, SQL does not supply any multiplication aggregate function, and the database owner should request the database service provider to implement the multiplication aggregate function (with modular arithmetic) on the data servers. Third, high computational costs of the encryption and decryption functions of the Paillier cryptosystem (used in, respectively, the share generation and secret recovery procedures) lead to extensive client-side computations for most types of queries. In particular, to process an exact-match or range query, the client should decrypt the tag values in all tuples in the query result in order to eliminate fake tuples. Fourth, the execution of aggregate queries with a MIN/MAX function on any sensitive or non-sensitive attribute needs several interactions between the client and the data servers to obtain the actual minimum/maximum, especially when the aggregate function should be performed with grouping on an attribute.

2.7 The existing attacks on SSDO schemes

In [43], Dautrich and Ravishankar introduced an attack on the SSDO schemes developed in [33, 35, 37, 38]. They showed that in a (K, N)-threshold SSDO scheme (with a constant distribution vector), if an adversary knows the secret values corresponding to K (or $K + 2$, if the scheme performs over a finite field with an unknown prime number of elements) tuples of shares, he or she can compute the distribution vector and subsequently recover the secret value corresponding to every available tuple of shares. Some other SSDO schemes published in the literature, including [34, 39–42], are also vulnerable to this attack.

In [42], Ghasemi introduced an attack on HJEC and similar $(2, N)$-threshold SSDO schemes. Suppose that two distinct pairs of shares are generated from the same secret value v by two distribution lines $P_i(x) = a_i x + v$ and $P_{i'}(x) = a_{i'} x + v$. The differences between the generated shares on the data servers DS_1 and DS_2 will be $(a_i - a_{i'})x_1$ and $(a_i - a_{i'})x_2$, respectively, where $\vec{x} = [x_1, x_2]^{\mathrm{T}}$ is the distribution vector. That is, the difference between every two shares generated from the same secret value on DS_1/DS_2 includes the factor x_1/x_2. Based on this observation, the attack is performed as follows. The adversary sorts the shares on DS_1/DS_2 and then obtains the multiset Med_1/Med_2 consisting of the differences between every two consecutive shares on DS_1/DS_2. Next, he or she computes the greatest common divisor (GCD) of every two elements in Med_1/Med_2. Among the obtained GCDs from Med_1/Med_2, the most frequent one is x_1/x_2.

3 Methodology

The organization of this section is as follows. In Sect. 3.1, the two new attacks on some SSDO schemes are introduced. In Sect. 3.2, the proposed range expansion technique is explained. In Sect. 3.3, the proposed mapping method and the proposed secure data outsourcing scheme are presented. In Sect. 3.4, the security of the proposed secure data outsourcing scheme against the attacks and inferences is analyzed.

3.1 Two new attacks on some SSDO schemes

This subsection introduces the two new attacks on HJEC and similar $(2, N)$ -threshold SSDO schemes. The targets of these attacks include the schemes in which some share range boundaries could be revealed, or distinct pairs of shares could be generated from the same secret value. Without loss of generality, such schemes are considered with the $(2, 2)$ threshold.

In a $(2, 2)$-threshold SSDO scheme, suppose that the pair of shares stored for the secret value v in the tuple T_i on the data servers is $(s_{i,1}, s_{i,2})$, which may be accessed on the data servers or captured during transferring along the communication channel in a tuple in the result of a query returned by the data servers. In order to recover the secret value, the horizontal coordinates x_1 and x_2 are needed to obtain the coefficient a_i as the slope of the line fitting this pair of shares. Then, the y-intercept of this line is equal to v. The relations can be expressed as:

$$\vec{s}_i^T = a_i.\vec{x}^T + v.\vec{1}^T, \tag{3}$$

where $\vec{s}_i = \left[s_{i,1}, s_{i,2}\right]^{\mathrm{T}}$ is the vector of the shares in the tuple T_i, and $\vec{x} = \left[x_1, x_2\right]^{\mathrm{T}}$ is the distribution vector. As can be seen, scaling $\vec{x}$ by any real value γ can be compensated by scaling a_i by $1/\gamma$ such that the line again fits the shares with no change in the secret value. Hence, the adversary only needs to know the ratio of x_2 to x_1 (denoted by ρ) and then to assume that the distribution vector is $[1, \rho]$ to compute the secret value v by:

$$v = \left(\rho s_{i,1} - s_{i,2}\right)/(\rho - 1). \tag{4}$$

Two main sources of the information leakage exploited in the two new attacks are the submitted queries and the results of queries. With respect to the submitted queries, some share range boundaries may be revealed, and with respect to the results of queries, some distinct pairs of shares generated from the same secret value may be obtained. In the following two subsections, the two new attacks are described. The first attack, which exploits Share Range Boundaries, is referred to as the SRB-based attack. The second attack, which exploits two Distinct Pairs of shares generated from the Same secret Value, is referred to as the DPSV-based attack.

3.1.1 The SRB-based attack

As a simple case in HJEC, suppose that an authorized client wants to issue an exact-match query on the sensitive numeric attribute O from the relation T of the form Query 1. The query is translated into two queries of the form Query 2 for $n \in \{1, 2\}$ (see Sect. 2.5). Then, each of the translated queries is submitted to its corresponding data server DS_n. An adversary, who may have access to either the data servers or the communication channel, could exploit the fact that this pair of queries is potentially corresponding to an exact-match query.

The set of possible generated shares from $v \in V$ using the coefficient partition C_v^m on the data server DS_n ($n \in \{1,2\}$) is denoted by $\left[d_{v,n}^m, e_{v,n}^m\right]$. For the sake of brevity in notations, in what follows, it is assumed that only one coefficient partition is assigned to v. Let $\overrightarrow{c_v} = [\alpha_v, \beta_v]^{\mathrm{T}}$ be the vector of the endpoints for the coefficient partitions assigned to v. The share range of v on DS_n is denoted by $[d_{v,n}, e_{v,n}]$ ($n \in \{1,2\}$). In Fig. 1, the solid lines with the slopes equal to the two endpoints of the coefficient partition represent the share range boundaries for any horizontal coordinate. For an adversary who observes the queries, only the share range boundaries $d_{v,1}$, $e_{v,1}$, $d_{v,2}$, and $e_{v,2}$ are known. There is a system of four nonlinear equations and five unknowns as:

$$\begin{cases} d_{v,1} = \alpha_v x_1 + v \\ d_{v,2} = \alpha_v x_2 + v \\ e_{v,1} = \beta_v x_1 + v \\ e_{v,2} = \beta_v x_2 + v \end{cases} \Rightarrow \overbrace{\begin{bmatrix} d_{v,1} & d_{v,2} \\ e_{v,1} & e_{v,2} \end{bmatrix}}^{\boldsymbol{R}} = \overbrace{\begin{bmatrix} \alpha_v \\ \beta_v \end{bmatrix}}^{\overrightarrow{C_v}} \cdot \overbrace{\begin{bmatrix} x_1 x_2 \end{bmatrix}}^{\vec{x}^T} + v \overbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}}^{\boldsymbol{J}} \Rightarrow \boldsymbol{R} = \overrightarrow{c_v}.\vec{x}^T + v.\boldsymbol{J}. \tag{5}$$

In other words, the system is a system of bilinear added with a linear form of variables. The above system of equations has not a unique solution because the number of equations is smaller than the number of variables and scaling $\vec{x}$ by any factor γ can be compensated by scaling $\overrightarrow{c_v}$ by $1/\gamma$. However, the value of ρ, which is required in (4) for computing v, can be easily obtained as follows. Equation (6) is a system of two nonlinear equations obtained from (5).

$$\begin{cases} d_{v,1} - e_{v,1} = (\alpha_v - \beta_v) x_1 \\ d_{v,2} - e_{v,2} = (\alpha_v - \beta_v) x_2. \end{cases} \tag{6}$$

By dividing the second equation of (6) to the first one, ρ can be obtained. Then, x_2 is written with respect to x_1 as follows:

$$x_2/x_1 = (d_{v,2} - e_{v,2})/\left(d_{v,1} - e_{v,1}\right) = \rho \Rightarrow x_2 = \rho x_1. \tag{7}$$

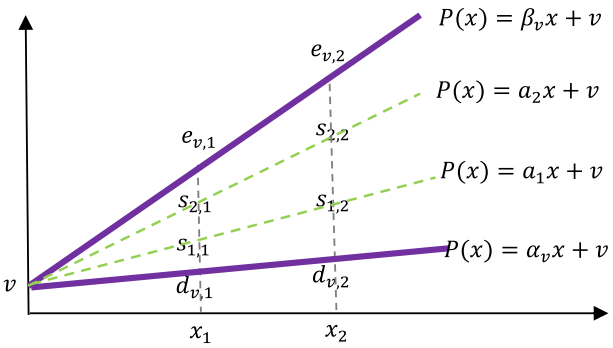

Fig. 1 Share ranges and two pairs of shares generated from a value in a (2,2)-threshold SSDO scheme

By discovering ρ, all the secret values of the sensitive attribute can be recovered by (4) if there is access to both data servers. However, even with no access to the data servers, the value of v can be found as follows. It is enough to consider $(d_{v,1}, d_{v,2})$ or $(e_{v,1}, e_{v,2})$ as a pair of shares generated from v. By using (4) and (7), the original value v is obtained by:

$$v = \frac{\rho d_{v,1} - d_{v,2}}{\rho - 1} = \frac{\rho e_{v,1} - e_{v,2}}{\rho - 1} = \frac{d_{v,1}e_{v,2} - d_{v,2}e_{v,1}}{(d_{v,1} + e_{v,2}) - (d_{v,2} + e_{v,1})} = \frac{|\boldsymbol{R}|}{2tr(\boldsymbol{R}) - \vec{1}^T \boldsymbol{R}\vec{1}}, \tag{8}$$

where $|\boldsymbol{R}|$ and $tr(\boldsymbol{R})$ are the determinant and trace of the share range matrix $\boldsymbol{R}$, respectively. If the share ranges have no overlap, the denominator of the fraction in (8) cannot be 0, and thus, v can be computed. In addition, for any nonzero value v, the determinant of $\boldsymbol{R}$ is not zero, and thus, $\boldsymbol{R}$ is invertible. Considering the distribution vector $\vec{x}$ as $x_1.[1, \rho]^{\mathrm{T}}$, by multiplying (5) by $\boldsymbol{R}^{-1}$ from right, the coefficient vector $\overrightarrow{c_v}$ can also be computed with respect to x_1 with an inverse correlation. However, computing $\overrightarrow{c_v}$ may be easier by transposing (5) and using the same reasoning as in (7) as follows:

$$\boldsymbol{R}^T = \vec{x}.\overrightarrow{c_v}^T + v.\boldsymbol{J} \Rightarrow \frac{\beta_v}{\alpha_v} = \frac{e_{v,1} - e_{v,2}}{d_{v,1} - d_{v,2}} = \theta_v \Rightarrow \beta_v = \theta_v \alpha_v \Rightarrow \overrightarrow{c_v} = \alpha_v.[1, \theta_v]^{\mathrm{T}}. \tag{9}$$

After computing ρ and θ_v by (7) and (9), respectively, and then replacing them in (5), $\overrightarrow{c_v}$ can be computed with respect to x_1 as follows:

$$\boldsymbol{R} = \alpha_v x_1 \begin{bmatrix} 1 \\ \theta_v \end{bmatrix}.[1, \rho] + v.\boldsymbol{J} \Rightarrow d_{v,1} = \alpha_v x_1 + v \Rightarrow \alpha_v = \frac{d_{v,1} - v}{x_1} \;\&\; x_1 = \frac{d_{v,1} - v}{\alpha_v}$$
$$\Rightarrow \overrightarrow{c_v} = \frac{(d_{v,1} - v)}{x_1} \begin{bmatrix} 1 \\ \theta_v \end{bmatrix} \;\&\; \vec{x} = \frac{(d_{v,1} - v)}{\alpha_v} \begin{bmatrix} 1 \\ \rho \end{bmatrix} \Rightarrow \boldsymbol{R} = (d_{v,1} - v) \begin{bmatrix} 1 \\ \theta_v \end{bmatrix}.[1, \rho] + v.\boldsymbol{J}. \tag{10}$$

3.1.2 The DPSV-based attack

In restriction-based SSDO schemes such as HJEC, from the query results returned by the servers and in metadata-based SSDO schemes such as [35], according to the structure of the stored indices or the returned query results, the adversary may be able to find at least two tuples with the same unknown secret value of a sensitive attribute. Without loss of generality, assume that the adversary finds T_1 and T_2 as two tuples with the same unknown secret value v. The stored shares for these two tuples on the data server DS_n ($n \in \{1, 2\}$) are denoted by $s_{1,n}$ and $s_{2,n}$. Let the slope of the lines corresponding to these pairs of shares in T_1 and T_2 be a_1 and a_2, respectively. In Fig. 1, each of the two dashed lines is corresponding to the line that fits one of these pairs of shares. In this case, the adversary can assume that there is a coefficient partition $[a_1, a_2]$ that generates share ranges $[s_{1,n}, s_{2,n}]$ on DS_n. Thus, (5) can be used again by redefining the matrix $\boldsymbol{R}$ and the vector $\overrightarrow{c_v}$ as:

$$\boldsymbol{R} = \overrightarrow{c_v}.\vec{x}^T + v.\boldsymbol{J} \text{ subject to } \boldsymbol{R} = \begin{bmatrix} s_{1,1} & s_{1,2} \\ s_{2,1} & s_{2,2} \end{bmatrix} \text{ and } \overrightarrow{c_v} = \left[a_1, a_2\right]^{\mathrm{T}}. \tag{11}$$

Now, inspired by (7), ρ can be computed as:

$$\rho = \left(s_{1,2} - s_{2,2}\right)/\left(s_{1,1} - s_{2,1}\right). \tag{12}$$

Also, inspired by (8), the secret value v can be computed as:

$$v = \left(s_{1,1}s_{2,2} - s_{1,2}s_{2,1}\right)/\left(\left(s_{1,1} + s_{2,2}\right) - \left(s_{1,2} + s_{2,1}\right)\right). \tag{13}$$

3.1.3 An example of the attacks on HJEC

Here, an example of the introduced attacks on HJEC is given. Suppose that a part of the information used by the database owner is as follows. The distribution vector is $\vec{x} = \left[x_1, x_2\right]^{\mathrm{T}} = [2, 5]^{\mathrm{T}}$. An original value for which an exact-match query is issued is $v = 7$, and its corresponding coefficient partition is $[6, 8]$. Also, a secret value for which a pair of shares has been stored in the tuple T_5 is $v' = 13$, and its corresponding coefficient partition is $[14, 18]$.

In the issued query, the values of the share range boundaries include $d_{7,1} = 6 \times 2 + 7 = 19$, $e_{7,1} = 8 \times 2 + 7 = 23$, $d_{7,2} = 6 \times 5 + 7 = 37$, and $e_{7,2} = 8 \times 5 + 7 = 47$. Suppose the random coefficient that has been used to generate a pair of shares in T_5 is $a_5 = 17$. Therefore, the stored shares for v' in T_5 are $s_{5,1} = 17 \times 2 + 13 = 47$ and $s_{5,2} = 17 \times 5 + 13 = 98$. An adversary who observes the values of $d_{7,1}$, $e_{7,1}$, $d_{7,2}$, $e_{7,2}$, $s_{5,1}$, and $s_{5,2}$ can perform the attack as follows. First, the value of ρ. is obtained by using (7) as:

$$\rho = (37 - 47)/(19 - 23) = 2.5.$$

Then, the value of v is obtained by using (8) as:

$$v = (2.5 \times 19 - 37)/(2.5 - 1) = 7.$$

In the same way, the value of v' can be obtained by (4) as:

$$v' = (2.5 \times 47 - 98)/(2.5 - 1) = 13.$$

3.2 The proposed range expansion technique

The SRB-based attack exploits the information about the share range boundaries of the original values in the domain of the sensitive attribute. In contrast, the three other attacks, including the DPSV-based attack and the two attacks in [42, 43], all exploit the information about the correspondences between the secret values and shares (but in different ways). Also, the sources of the information leakage exploited in the SRB-based and the three other attacks are different. As a result, the ways to thwart the attack are also different. This subsection addresses only the SRB-based attack.

As mentioned earlier, the source of the information leakage exploited in the SRB-based attack on HJEC and similar $(2, N)$-threshold SSDO schemes is the queries submitted to the data servers. In order to address this attack, the information leakage with respect to the coefficient partition assignment should be controlled. To this end, a range expansion technique is proposed that expands the ranges in every range predicate on the protected searchable attributes in the translated queries in order to hide the share range boundaries of the original values from any query observer. For example, consider an exact-match query of the form Query 1 and its translated queries of the form Query 2 in HJEC (see Sect. 2.5). By the proposed range expansion technique, these translated queries are modified to two queries of the form Query 3 for $n \in \{1, 2\}$, where Δd_1, Δe_1, Δd_2, and Δe_2 are four positive random numbers, independent from each other and from the original value involved in the query. Then, each of these modified queries is submitted to its corresponding data server DS_n.

Query 3

SELECT $*$
FROM T
WHERE $\alpha_v x_n + v - \Delta d_n \leq O_n^* \leq \beta_v x_n + v + \Delta e_n$

This technique prevents revealing the original share range boundaries to the data servers or any query observer when such a query is issued. The problem with the range expansion technique is the possibility of false hits. In order to address this problem, the coefficient partition assignment function is modified as follows. As before, let V and C be the domain of the searchable attribute O and the domain of the coefficients, respectively. The coefficient domain C is partitioned into $2|V| - 1$ partitions, including $|V|$ coefficient partitions and $|V| - 1$ gap partitions, such that a gap partition is inserted between every two consecutive coefficient partitions. All the gap partitions are of the same size ε, where ε is a predetermined parameter, while the sizes of the coefficient partitions are determined based on the weighted partitioning method of HJEC. Assuming that the difference between every two consecutive original values in the domain of the sensitive attribute is a constant value, the insertion of these gap partitions provides a gap of a constant size between every two consecutive share ranges. Let the coefficient partitions assigned to the two consecutive values v and v' $(v < v')$ be $C_v = [\alpha_v, \beta_v]$ and $C_{v'} = [\alpha_{v'}, \beta_{v'}]$, respectively. The size of the gap between the share ranges of v and v' on DS_1/DS_2, denoted by G_1/G_2, can be obtained by (14). The size of the gap between the share ranges of every two consecutive values in V is the same.

$$G_n = d_{v',n} - e_{v,n} = \left(\alpha_{v'} x_n + v'\right) - \left(\beta_v x_n + v\right) = x_n \varepsilon + \left(v' - v\right) \text{ for } n \in \{1, 2\}. \tag{14}$$

Therefore, selecting the random values Δd_n and Δe_n to be smaller than G_n (for $n \in \{1, 2\}$) in the translated queries will guarantee that the results do not contain any false hit. HJEC with the proposed range expansion technique is resistant to the SRB-based attack but still vulnerable to the other three attacks.

3.3 The proposed secure data outsourcing scheme

As mentioned earlier, the DPSV-based attack and the two attacks in [42, 43] all exploit the information about the correspondences between the secret values and shares (but in different ways). HJEC and similar $(2, N)$-threshold SSDO schemes are vulnerable to both the DPSV-based attack and Ghasemi's attack [42] because they generate distinct pairs of shares from the same secret value. Also, most existing (K, N)-threshold SSDO schemes are vulnerable to the attack introduced by Dautrich and Ravishankar [43] because they generate each tuple of shares directly from a secret value. In such schemes, when an adversary knows the secret values corresponding to K tuples of shares, he or she will have a system of K equations and K unknowns (which can be solved). Our secure data outsourcing scheme should have the following two characteristics in order to thwart all three attacks: (1) It never generates distinct pairs (tuples) of shares from the same value. (2) The adversary cannot know the values from which the tuples of shares have been generated even if he or she knows the secret values corresponding to some tuples of shares.

In order to achieve the first characteristic, the most intuitive approach is to restrict the share generation procedure such that from each secret value of a sensitive attribute (in any tuple of the relation), a unique tuple of shares is generated, as in EMAA. The problem with this approach is that it is extremely vulnerable to identicalness and distribution inferences. Our solution is to employ a one-to-many mapping of secret values to some mapping values and then to generate shares from the mapping values such that from each mapping value (in any tuple of the relation), a unique tuple of shares can be generated. With this approach, various tuples of shares can be generated from the same secret value of a sensitive attribute (in different tuples of the relation), while distinct tuples of shares are never generated from the same (mapping) value. In order to achieve the second characteristic, our approach is to map each secret value to a mapping value using a secret mapping so that the tuples of shares are generated from the mapping values rather than directly from the secret values. That is, this secret mapping works as an additional layer of security. Unfortunately, the traditional encryption methods such as DES and AES cannot be employed for this purpose because ciphertexts generated by such methods are neither homomorphic nor searchable. Our unified solution to achieve both characteristics and, at the same time, preserve the homomorphism and searchability properties of secret sharing is to employ a secret one-to-many mapping with a finite set of linear mapping rules as the first layer of security. In the rest of this subsection, the proposed secure data outsourcing scheme with the proposed mapping method is described in detail.

Without loss of generality, the proposed scheme is described with the $(2, 2)$ threshold. As before, let the sensitive attribute to be protected and its domain be denoted by O and V, respectively. A mapping domain W is selected for V so that each value from V is assigned an interval of the values from W, which is called a mapping partition. A pair of shares is generated from a secret value as follows. The secret value is first mapped to a mapping value from its corresponding mapping partition using a secret one-to-many mapping. A pair of shares is then generated from that mapping value by constructing

a distribution line with a coefficient selected from a pre-assigned coefficient partition based on a hash function in a similar way as in EMAA.

Since the proposed scheme never generates distinct pairs of shares from the same mapping value, it is resistant to both the DPSV-based attack and Ghasemi's attack [42]. On the other hand, it generates various pairs of shares from each original value, thereby reducing the potential for identicalness inferences. In addition, since each secret value is mapped to a mapping value by a secret mapping, even if the adversary knows the secret values corresponding to some pairs of shares, he or she will not know that each pair of shares has been generated from which mapping value. As a result, this additional layer of security thwarts the attack introduced by Dautrich and Ravishankar [43] and some other potential attacks. Our new secure data outsourcing scheme is based on secret sharing and utilizes the proposed mapping method and the proposed range expansion technique. The proposed range expansion technique thwarts the SRB-based attack. On the other hand, the proposed mapping method works an additional layer of security to thwart the other attacks and provides resistance to identicalness inferences. In addition, the weighted partitioning method provides resistance to equality and distribution inferences.

Figure 2 illustrates the diagram of the proposed secure data outsourcing scheme, including two main components: system setup and query processing. The two components use three procedures: share generation, query translation, and secret recovery. The system setup component and the share generation and secret recovery procedures are presented in the rest of this subsection. The query processing component and the query translation procedure will be presented in Sect. 4.

Before considering the system setup component in detail, the proposed mapping method is considered. In order to preserve the additive homomorphism property, the mapping should fulfill two requirements. One is that every mapping rule should be linear, and the other is that the number of distinct mapping rules should be limited. A parameter L is selected as the size of every mapping partition, which is equal to the number of distinct mapping rules. Then, a mapping domain W is selected such that $|W| = L \times |V|$. Assume the primary key of the relation T is the attribute PK with a domain of integer numbers. The secret value v in the tuple T_i (of the relation T) is mapped to the mapping value w_i by:

$$w_i = v \times L + \zeta\left(PK_i \% L, [0 \dots L-1]\right), \tag{15}$$

where PK_i is the primary key value of the tuple T_i, and $\zeta : [0 \dots L-1] \rightarrow [0 \dots L-1]$ is a one-to-one function that maps the value of $PK_i \% L$ to a value in the range $[0 \dots L-1]$. An easy way to implement the function ζ is to use a pre-generated random permutation of L values in the range $[0 \dots L-1]$. By the mapping of (15), the secret value v in the tuple T_i is mapped to one of the L distinct values in the mapping partition $W_v = [v \times L \dots v \times L + L - 1]$ based on $PK_i \% L$. The mapping rules implemented by the function ζ should be kept secret and shared with only authorized clients. In some aggregate queries, the value of L is submitted to the data servers, as will be seen in Sect. 4. Therefore, the parameter L cannot be considered secret. As an alternative, an auxiliary column of the domain $[0 \dots L-1]$ can be added to the relation, which is filled randomly, to be used instead of $PK \% L$.

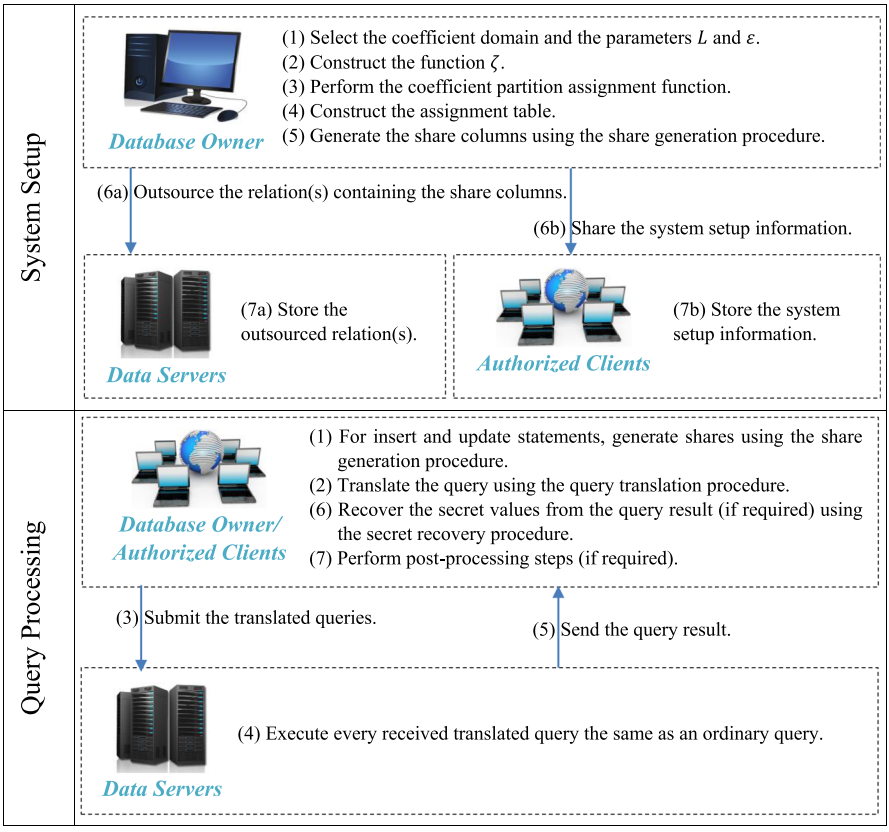

Fig. 2 The diagram of the proposed secure data outsourcing scheme

The proposed scheme performs the system setup for a sensitive attribute with the domain V as follows. The coefficient domain C and the parameters L and ε are selected, and the function ζ is constructed. The domain C is divided into $2|V|-1$ partitions, including $|V|$ coefficient partitions and $|V|-1$ gap partitions. Each coefficient partition is assigned to one of the original values in V, and a gap partition is inserted between each two consecutive coefficient partitions. The coefficient partition assignment function reserves $\varepsilon \times (|V|-1)$ coefficients for the gap partitions (ε is the size of every gap partition). It partitions $|C| - \varepsilon \times (|V|-1)$ coefficients for the coefficient partitions by assigning a coefficient partition to each value with a size proportional to the relative frequency of that value in the sensitive attribute in an order-preserving manner. The insertion of a gap partition between every two consecutive coefficient partitions is utilized to employ the range expansion technique as considered in Sect. 3.2. An assignment table is constructed that consists of the coefficient partition assigned to each original value and its corresponding pair of share range boundaries. The assignment table can be utilized in both the secret recovery and query translation procedures. The share range boundaries corresponding to each original value $v \in V$ are computed as:

$$\begin{cases} d_{v,1} = \alpha_v x_1 + l_v \\ d_{v,2} = \alpha_v x_2 + u_v \\ e_{v,1} = \beta_v x_1 + l_v \\ e_{v,2} = \beta_v x_2 + u_v \end{cases}, \tag{16}$$

where $C_v = [\alpha_v, \beta_v]$ is the coefficient partition assigned to v, $l_v = v \times L$, and $u_v = v \times L + L - 1$ (see Fig. 3).

The share generation procedure of the proposed scheme generates a pair of shares from the secret value v in the tuple T_i as follows. The secret value v is first mapped to the mapping value w_i based on PK_i using (15). The coefficient a_i is then computed based on a hash function that maps w_i to a value in the coefficient partition assigned to v, in a similar way as in EMAA. That is, for each mapping value, a unique coefficient is selected. Finally, the pair of shares is generated by:

$$s_{i,n} = a_i x_n + w_i \text{ for } n \in \{1, 2\}, \tag{17}$$

where $\vec{x} = [x_1, x_2]^{\mathrm{T}}$ is the distribution vector.

The secret recovery procedure of the proposed scheme can be performed by two methods, referred to as the interpolation-based and search-based recovery methods. Let $(s_{i,1}, s_{i,2})$ be the pair of shares in the tuple T_i. In order to recover the secret value v in the tuple T_i, the interpolation-based recovery method needs both shares, while the search-based recovery method needs one of the shares. Let $\vec{x} = [x_1, x_2]^T$ be the distribution vector and $\rho = x_2/x_1$. The interpolation-based recovery method first computes the mapping value w_i using (18) and then applies the reverse mapping to w_i based on PK_i using (19).

$$w_i = \left(\rho s_{i,1} - s_{i,2}\right)/(\rho - 1) \tag{18}$$

$$v = \left(w_i - \zeta\left(PK_i \% L, [0 \ldots L-1]\right)\right)/L. \tag{19}$$

The search-based recovery method conducts a binary search within the assignment table to find the share range to which the searched share belongs. However, the utilization of the additive homomorphism property in the execution of the aggregate functions SUM and AVG on a sensitive attribute protected by the proposed scheme needs to execute the associated translated queries on both data servers and then

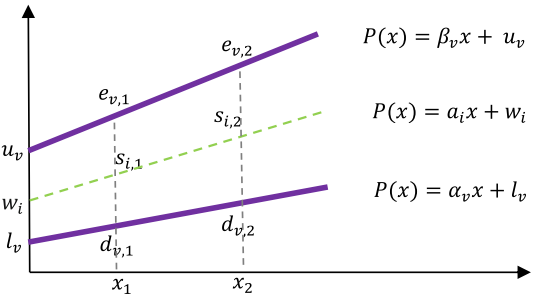

Fig. 3 The share ranges of an original value and a pair of shares generated from a mapping value by the proposed scheme

employ the interpolation-based recovery method. It should be noted that the proposed scheme is applicable with any (K, N) threshold ($K \leq N$).

3.4 Security analysis of the proposed scheme

This subsection analyzes the security of the proposed secure data outsourcing scheme against the attacks and inferences. In the previous SSDO schemes, the attack introduced by Dautrich and Ravishankar [43] can be performed if the secret values corresponding to at least K tuples of shares are known. In the proposed scheme, tuples of shares are generated from mapping values rather than secret values. The function ζ is kept secret. Thus, even if an adversary knows the secret values corresponding to some tuples of shares, he or she cannot find the mapping values corresponding to those tuples of shares. As a result, the proposed mapping method provides resistance to the attack introduced by Dautrich and Ravishankar [43]. With this additional layer of security, the proposed scheme could be resistant to some other potential attacks.

In contrast to HJEC and some other SSDO schemes, the proposed scheme does not generate distinct pairs of shares from the same (mapping) value. As a result, the proposed scheme is resistant to both the DPSV-based attack and Ghasemi's attack [42]. The proposed scheme with a $(2, N)$ threshold without range expansion is considered. Let v be an original value, and $W_v = [l_v \dots u_v]$ and $C_v = [\alpha_v, \beta_v]$ be its corresponding mapping partition and coefficient partition, respectively. Assume an adversary knows the parameter L, as well as the share range boundaries $d_{v,1}$, $e_{v,1}$, $d_{v,2}$, and $e_{v,2}$. He or she will have a system of four linear equations and five unknowns as:

$$\begin{cases} d_{v,1} = \alpha_v x_1 + l_v \\ d_{v,2} = \alpha_v x_2 + l_v \\ e_{v,1} = \beta_v x_1 + l_v + L - 1 \\ e_{v,2} = \beta_v x_2 + l_v + L - 1 \end{cases} \tag{20}$$

From this system, inspired by (7), the ratio of x_2 to x_1 (denoted by ρ) can be obtained by (21). Then, the values of l_v and v can be obtained by (22) and (23), respectively.

$$\rho = (d_{v,2} - e_{v,2} + L - 1)/\left(d_{v,1} - e_{v,1} + L - 1\right) \tag{21}$$

$$l_v = \left(\rho d_{v,1} - d_{v,2}\right)/(\rho - 1) \tag{22}$$

$$v = l_v/L. \tag{23}$$

After obtaining l_v and v, the mapping partition corresponding to every original value in the domain of the sensitive attribute can be computed. After that, the adversary can compute the mapping value corresponding to every available pair of shares by using (18) based on the shares and the value of ρ. Finally, he or she can obtain

the secret value corresponding to every mapping value by finding the mapping partition to which the mapping value belongs. Therefore, employing the range expansion technique in the proposed scheme is essential to resist the SRB-based attack. In summary, the proposed scheme with range expansion is resistant to all four attacks.

The proposed scheme provides a degree of resistance to identicalness inferences. However, because at most L distinct tuples of shares can be generated from each original value, there is the potential for identicalness inferences. In the proposed scheme, there is a tradeoff between the amount of vulnerability to identicalness inferences and the communication and client-side computation overheads (for some types of queries). The larger the value of L, the lower vulnerability to identicalness inferences. On the other hand, the smaller the value of L, the lower the communication and client-side computation overheads. Furthermore, the proposed scheme with the weighted partitioning method is resistant to equality and distribution inferences. However, similar to any order-preserving scheme, the proposed scheme is vulnerable to ordering inferences.

4 Query processing

This section considers the query processing component of the proposed secure data outsourcing scheme with a (K, N) threshold in general. For ease of discussion, queries on an outsourced relation (denoted by T) that contains just one searchable attribute protected by the proposed scheme are considered. The sensitive attribute and its associated share column on DS_n $(1 \leq n \leq N)$ are denoted by O and O_n^*, respectively. Also, the notation Q indicates one or multiple non-sensitive attributes in T, and COUNT(O)/SUM(O)/AVG(O)/MIN(O)/MAX(O) denotes the aggregate function COUNT/SUM/AVG/MIN/MAX on the attribute O. In most cases, the discussions can be generalized for more than one searchable attribute to be protected. The client translates every query into one query, a tuple of queries, or a set of queries (depending on the type of that query and the used execution and secret recovery methods) and then issues the translated queries to the data servers. After receiving the results, the client performs the secret recovery procedure and any other post-processing step (if required) to obtain the final result. In our examples, the notations φ and $Trans(\varphi)$ are used to denote one or multiple predicates in a query and the translation of the predicate φ, respectively.

4.1 Exact-match and range queries

The translation of exact-match and range predicates on sensitive attributes is straightforward. Let for the two original values v and v' of the sensitive attribute O in the relation T, $W_v = \left[l_v \dots u_v\right]$ be the mapping partition corresponding to v, $C_v = \left[\alpha_v, \beta_v\right]$ be the coefficient partition corresponding to v, $W_{v'} = \left[l_{v'} \dots u_{v'}\right]$ be the mapping partition corresponding to v', and $C_{v'} = \left[\alpha_{v'}, \beta_{v'}\right]$ be the coefficient partition corresponding to v'. An exact-match or range query of the form Query 4 (where $v \leq v'$) is translated into N queries of the form Query 5, where $n \in \{1, 2, \dots, N\}$.

Then, each of the translated queries is submitted to its corresponding data server. Here, queries containing a disjunction of multiple exact-match predicates on the same sensitive attribute are categorized as range queries. Exact-match queries are processed without the need to perform the secret recovery procedure. Also, in the secret recovery procedure of a range query, either the interpolation-based or search-based recovery method can be employed. For processing range queries with the interpolation-based recovery method, at least K translated queries should be submitted to their corresponding data servers. In contrast, for processing exact-match queries and range queries with the search-based recovery method, the submission of just one of the translated queries to its corresponding data server is sufficient.

Query 4

SELECT $*$
FROM T
WHERE $v \leq O \leq v'$

Query 5

SELECT $*$
FROM T
WHERE $\alpha_v x_n + l_v - \Delta d_n \leq O_n^* \leq \beta_{v'} x_n + u_{v'} + \Delta e_n$

4.2 Aggregate queries

Aggregate queries in which at least one sensitive attribute is involved are here classified into four categories and denoted as follows:

- *AGG* Queries containing one or more aggregate functions SUM, AVG, MIN, and MAX on non-sensitive attributes or COUNT on any attribute with one or more predicates on sensitive attributes.
- *GROUPING* Queries containing one or more aggregate functions with grouping on sensitive attributes.
- *SUM/AVG* Queries containing the aggregate function SUM or AVG on sensitive attributes.
- *MIN/MAX* Queries containing the aggregate function MIN or MAX on sensitive attributes.

Note that queries of any of these four types can contain predicates on sensitive and non-sensitive attributes and/or grouping on non-sensitive attributes. In fact, AGG queries include aggregate queries in which no sensitive attribute is involved in either the aggregate function(s) or the GROUP BY clause. It should be noted that in the execution of the aggregate function COUNT, the values of the attribute are not involved. Thus, the processing of the aggregate function COUNT on

a sensitive attribute is the same as that on a non-sensitive attribute. In the rest of this subsection, the processing of these four types of queries by the proposed scheme is considered.

In the proposed scheme, the translation of an AGG query is done by translating the predicate of the query in the same way as for exact-match and range queries. Then, the translated query corresponding to just one of the data servers is submitted and executed.

HJEC has not considered GROUPING queries. In HJEC, there may be too many distinct shares generated from each original value in the domain of a sensitive attribute in the outsourced relation. Therefore, the result of a translated query with grouping on a share column associated with a sensitive attribute may include a large number of tuples. In the proposed scheme, two methods are suggested to process GROUPING queries. By both methods, such queries can be executed by retrieving results from just one of the data servers. The first method is to execute the aggregate function for each group separately. That is, a set of queries is issued in which a predicate that is satisfied for one of the groups is appended to each query. However, when the number of groups is large, this method is inefficient. The second method is to execute the aggregate function in the translated query with GROUP BY O_n^* (on DS_n). That is, each group within the result of the translated query is a subgroup of one of the original groups within the final result. After receiving the result of the translated query at the client side, the client aggregates the subgroups into the original groups. In the result received from any data server, the number of tuples (i.e., subgroups) that belong to each original group is at most L because there are at most L distinct tuples of shares generated from each original value.

It should be noted that the second method cannot be used to execute a query containing AVG(Q) with GROUP BY O directly. Such a query is translated into a query to obtain COUNT(Q) and SUM(Q) with GROUP BY O. Then, AVG(Q) for each group of O is computed at the client side. Also, the HAVING clause (if present) is omitted in the translated queries and is applied to the original groups at the client side. In summary, the first method supports the fully server-side execution of GROUPING queries, while the second method supports a partially server-side execution of GROUPING queries.

The proposed scheme satisfies the additive homomorphism property for each set of tuples that use the same mapping rule separately. Here, a server-side-partial-aggregation mechanism is employed to execute the aggregate functions SUM and AVG on a sensitive attribute as follows. A query containing SUM(O) with GROUP BY Q is considered. Each group of Q (as an original group) contains at most L subgroups such that the tuples in each subgroup contain shares generated by the same mapping rule and have the same value of Q. After receiving the sum of the shares in each subgroup from at least K data servers, the client applies interpolation to each K-tuple of sums and then applies the reverse mapping on the obtained value. Finally, the client computes the sum of all the values obtained for every original group. Accordingly, a query of the form Query 6 is translated into N queries of the form Query 7, where $n \in \{1, 2, \ldots, N\}$.

Query 6

SELECT Q, SUM(O)
FROM T
WHERE φ

Query 7

SELECT Q, MOD(PK, L), SUM(O_n^*)
FROM T
WHERE $Trans(\varphi)$
GROUP BY Q, MOD(PK, L)

A query containing AVG(O) is translated into queries to obtain COUNT(O) and SUM(O). Then, AVG(O) is computed at the client side. In summary, the proposed scheme supports a partially server-side execution of queries with the aggregate functions SUM and AVG on a sensitive attribute.

In the translation of a query in which only the minimum or maximum value of the sensitive attribute O is appealed (with or without grouping on one or more non-sensitive attributes), MIN(O)/MAX(O) is translated into MIN(O_n^*)/MAX(O_n^*), where $n \in \{1, 2, \ldots, N\}$. Then, the translated query is submitted to the associated data server. Note that a query with a nested subquery may need to be executed in two rounds. As an example, consider a query of the form Query 8. Such a query is executed as follows. In the first round, the nested query is translated, and the translated query is executed to obtain the maximum value (denoted by max). Then, in the second round, an exact-match query with the predicate $Trans(O = max)$ is submitted to one of the data servers to obtain the tuples with the maximum value of the sensitive attribute.

Query 8

SELECT $*$
FROM T
WHERE O = (SELECT MAX(O) FROM T)

4.3 Data manipulation statements

The data manipulation statements include insert, delete, and update. For insert and update statements, new tuples of shares are generated by the share generation procedure of the proposed scheme. A delete/update statement is translated by translating the predicate of the delete/update statement in the same way as for an exact-match or range query. In restriction-based SSDO schemes, including the proposed scheme, an update statement in which a protected sensitive attribute is updated should not be

performed at the server side because of the problem of restriction violation. Such update statements should be performed on the client side.

5 Experimental results and evaluation

In this section, the results of several experiments conducted to evaluate the performance of the proposed secure data outsourcing scheme are reported, and some other discussions are made. In these experiments and discussions, the baseline scheme refers to the data outsourcing scheme on a single data server without imposing any protection on any attribute. In this section, the computation and communication costs to process various types of queries and the storage costs to outsource a sensitive attribute by the proposed scheme are evaluated and compared to the baseline scheme, EMAA, HJEC, and FTI. In all the experiments, HJEC with the secure partitioning method is considered. Also, the results reported for FTI are based on its original scheme in [42] (i.e., without applying the correction discussed in Sect. 2.6).

In each one of the considered SSDO schemes, including EMAA, HJEC, and FTI, two secret recovery methods similar to those employed in the proposed scheme can be used: interpolation-based and search-based. In this section, the notations EQU and RNG are sometimes used to refer to exact-match (or equality) and range queries, respectively. Also, the notations RNG^{IR} and RNG^{SR} are sometimes used to refer to range queries with processing by the interpolation-based and search-based recovery methods, respectively. Note that secret recovery in both HJEC and FTI needs an extra step to eliminate false hits before the secret values are recovered. In HJEC, the client should obtain the intersection of the results received from all the data servers. In FTI, the client should decrypt the tag value of every tuple in the query result and eliminate the fake tuples (i.e., those with a decrypted tag value 0).

All the schemes were implemented with the $(2, 2)$ threshold by the C# programming language. The implementation of each data outsourcing scheme involves only its client-side application. Three machines with a Core i3 3.6 GHz CPU, 8 GB RAM, and Microsoft Windows 10 64-bit OS were employed: one machine as the client and the two others as the data servers running MS SQL Server 2014 (32 bit). A university database containing a relation *takes* with five attributes *year*, *semester*, *section_id*, *student_id*, and *grade* was designed. In the relation *takes*, its first four attributes were specified as the primary key of the relation, and the two attributes *student_id* and *grade* were considered as the searchable attributes to be protected. Two million artificial tuples were generated and inserted into the relation *takes* with uniform and normal distributions of the attributes *student_id* and *grade*, respectively.

Assume that $C = [1..|C|]$ and $V = [1..|V|]$. The upper bound of the generated shares by the proposed scheme for an attribute with the domain V and the coefficient domain C on each data server DS_n with x_n as its distribution vector element (for $1 \leq n \leq N$) is $|C|.x_n + (|V| + 1).L - 1$. That is, the size of the share domain for such an attribute on each data server DS_n depends on $|V|$, $|C|$, x_n, and L. In general, the coefficient domain is usually selected based on $|V|.L$, and its size should preferably

be multiple times greater than $|V|.L$. Thus, the size of the share domain on the data server DS_n is mainly determined based on $|C|.x_n$.

The domain of the attribute *student_id* is $V_1 = [10,000..29,999]$ (i.e., $|V_1| = 20,000$). Also, the attribute *grade*, which its values originally range from 0.0 to 20.0, was transformed to a column with the integer domain $V_2 = [0 \dots 200]$ (i.e., $|V_2| = 201$) in the original relation. The common parameters in all the SSDO schemes considered in the experiments, including EMAA, HJEC, FTI, and the proposed scheme, are as follows. The selected coefficient domains for the attributes *student_id* and *grade* are $C_1 = [1..10,000,000]$ and $C_2 = [1..200,000]$, respectively. Also, the selected distribution vector for both sensitive attributes in all the experiments is $[29, 81]^{\mathrm{T}}$. Using the selected coefficient domains and the other parameters, the generated shares from both attributes fit into the 4-byte integer data type (i.e., the data type *int* in SQL). However, to perform the aggregate function SUM on the generated shares, it may be needed to cast shares to the 8-byte integer data type (i.e., the data type *bigint* in SQL). The parameters specific to the proposed scheme are as follows. For the attribute *student_id*, the sizes of every coefficient partition and gap partition are 400 and 100, respectively. Also, for the attribute *grade*, the size of every coefficient partition is determined by the weighted partitioning method, and the size of every gap partition is 250. In order to consider the effect of the parameter L on the performance of the proposed scheme, two distinct pairs of relations were generated using two values 50 and 200 of the parameter L (for both sensitive attributes) and outsourced.

Before considering the results of the experiments, it should be noted that the main objective of the proposed scheme is to achieve a high level of security with the support of a wide variety of queries. In fact, it is fully acceptable to employ the proposed scheme rather than a data outsourcing scheme with security drawbacks even if the proposed scheme incurs greater costs (including computation and communication costs of query processing and storage costs) than those data outsourcing schemes with security drawbacks.

5.1 Computation costs

This subsection reports and discusses the results of four experiments conducted to evaluate the computation costs of the proposed scheme and compare them with the other schemes. The first and second experiments consider the execution times of the share generation and secret recovery procedures of the proposed scheme, respectively. The third and fourth experiments evaluate the overall client-side and server-side execution times of query processing by various schemes.

In the first experiment, the pairs of shares corresponding to the secret values of the two attributes *student_id* and *grade* in the two million tuples of the relation *takes* were generated by EMAA, HJEC, FTI, the proposed scheme with $L = 50$, and the proposed scheme with $L = 200$. The hash algorithm used to compute the coefficient for generating each pair of shares in both EMAA and the proposed scheme is SHA-256. Table 1 shows the share generation times of the SSDO schemes for the two attributes in seconds (s). Note that the times reported for FTI include the

Table 1 The share generation times of the SSDO schemes (s)

Attribute	EMAA	HJEC	FTI	Proposed ($L = 50$)	Proposed ($L = 200$)
student_id	2.48	0.09	47.29	2.48	2.46
grade	2.47	0.09	47.83	2.49	2.48

Table 2 The execution times of the two secret recovery methods of the proposed scheme (ms)

Query set size	Interpolation-based recovery		Search-based recovery	
	student_id	*grade*	*student_id*	*grade*
100,000	1.6	1.6	25.3	15.6
200,000	3.1	3.1	49.8	30.9
500,000	7.8	7.8	126.1	76.7
1,000,000	15.5	15.6	248.5	153.4
2,000,000	31.1	31.0	496.6	306.9

times required to generate (1) pairs of shares corresponding to the secret values in the relation, (2) 10% fake tuples, and (3) tag values for all the actual and fake tuples using the encryption function of the Paillier cryptosystem. As can be seen, the share generation times of EMAA, the proposed scheme with $L = 50$, and the proposed scheme with $L = 200$ for the two attributes are nearly the same. HJEC and FTI have, respectively, the fastest and slowest share generation procedures. The high share generation time of FTI is due to the extensive computations of the encryption function of the Paillier cryptosystem. Also, the greater share generation times of EMAA and the proposed scheme than that of HJEC are due to the computations of SHA-256.

The second experiment considers the execution times of the secret recovery procedure of the proposed scheme using the interpolation-based and search-based recovery methods. The results of several range queries with different query set sizes on the shares of each of the two attributes, both with $L = 200$, were recovered by both secret recovery methods. Table 2 reports the average secret recovery times of the two secret recovery methods of the proposed scheme for various query set sizes in milliseconds (ms). As expected, the execution times of the interpolation-based recovery method for the two attributes are nearly the same. On the other hand, the execution time of the search-based recovery method for the attribute *student_id* is greater than that for the attribute *grade*. This is because the assignment table of the attribute *student_id* is about 100 times greater than that of the attribute *grade*. Also, it can be seen that for both attributes, the execution time of the interpolation-based recovery method is significantly lower than that of the search-based recovery method. Repeating this experiment with $L = 50$ showed that there are no meaningful differences between the results obtained for the two values of L. Similar results were obtained for EMAA. The false hit elimination step in the secret recovery procedures of HJEC and FTI significantly increases the client-side execution times of

query processing of these schemes. The client-side execution times of all the SSDO schemes are evaluated and compared in the fourth experiment.

For the next two experiments, several queries were executed for each type of query by the proposed and other schemes, and the client-side and server-side execution times were measured. All the execution times are reported in milliseconds (ms). Here, the client-side computations refer to those related to the outsourced query processing. In the baseline scheme, queries of any type are fully processed at the server side, and it has no client-side computation. In the four considered SSDO schemes, including EMAA, HJEC, FTI, and the proposed scheme, the client-side computations include query translation (in all four schemes, for all the types of queries), recovering the secret values (in all four schemes, for some types of queries), elimination of false hits (in HJEC and FTI, for all the supported types of queries), and aggregation of the subgroups into the original groups (in the proposed scheme, for GROUPING and SUM/AVG queries). In all the schemes, the time required for query translation is negligible. All the executed queries contained one to four predicates on the sensitive and/or non-sensitive attributes. Also, half of the executed SUM/AVG and MIN/MAX queries were with grouping on a non-sensitive attribute, and the other half were not with grouping.

In all the SSDO schemes, the following settings were applied. For GROUPING and MIN/MAX queries, the search-based recovery method was used. Although the secret recovery time of range queries by the interpolation-based recovery method is lower than that by the search-based recovery method, the search-based recovery method was adopted for the third and fourth experiments to reduce the communication costs. The type SUM/AVG is the only query type for which it is essential to execute the translated queries corresponding to each query on both data servers and then recover the query result by the interpolation-based recovery method. For SUM/AVG queries, among the server-side execution times on the two data servers, the greater one is taken as the server-side execution time. For any other type of query, the average server-side execution times on the two data servers is taken as the server-side execution time since it is assumed that the client submits just one of the translated queries corresponding to the query to be processed to one of the data servers randomly. In the proposed scheme, all GROUPING queries were processed by the second method (see Sect. 4.2). In the rest of this subsection, the results of these two experiments are considered.

In the third experiment, the average client-side and server-side execution times and the computation overhead of several queries of various types with two query set sizes executed by the proposed scheme on each of the two data servers DS_1 and DS_2 were obtained and compared with each other and with the baseline scheme. Tables 3 and 4 show the results of this experiment on, respectively, the attributes *student_id* and *grade*, each one with both $L = 50$ and $L = 200$. Here, the computation overhead of a query is computed as the overall execution time (i.e., the client-side execution time plus the server-side execution time) required to process that query by the proposed scheme minus that by the baseline scheme to that by the baseline scheme (in percentage). The two attributes have different domain sizes and have outsourced with different settings. EQU and AGG queries are processed without the need to perform the secret recovery procedure. Also, for MIN/MAX queries, the number

Table 3 The execution times (ms) and computation overheads (%) of query processing on the attribute *student_id*

Type of query	Query set size	Baseline	Proposed scheme ($L = 50$)				Proposed scheme ($L = 200$)			
		DS_1	Client	DS_1	DS_2	Overhead	Client	DS_1	DS_2	Overhead
RNG^{SR}	10,000	39	3	40	41	12	3	42	38	10
	50,000	44	13	47	45	34	13	46	48	36
AGG	10,000	37	0	37	36	−1	0	37	36	−1
	50,000	37	0	36	38	0	0	36	39	1
GROUPING	10,000	31	2	43	43	45	3	46	47	60
	50,000	67	4	88	86	36	7	96	98	55
SUM/AVG	10,000	32	2	34	33	13	2	44	42	44
	50,000	46	2	72	69	61	2	76	79	76
MIN/MAX	10,000	34	0	35	35	3	0	34	36	3
	50,000	46	0	46	45	−1	0	43	45	−4

Table 4 The execution times (ms) and computation overheads (%) of query processing on the attribute *grade*

Type of query	Query set size	Baseline	Proposed scheme ($L = 50$)				Proposed scheme ($L = 200$)			
		DS_1	Client	DS_1	DS_2	Overhead	Client	DS_1	DS_2	Overhead
EQU	10,000	26	0	37	36	14	0	35	37	13
	25,000	29	0	38	38	9	0	38	37	7
RNG^{SR}	10,000	39	2	39	38	4	2	40	38	5
	50,000	45	8	46	47	21	8	50	46	24
AGG	10,000	38	0	37	36	−4	0	38	37	−1
	50,000	38	0	37	39	0	0	37	37	−3
GROUPING	10,000	25	1	28	31	22	2	33	32	38
	50,000	61	2	78	77	30	2	83	85	41
SUM/AVG	10,000	23	2	37	35	67	2	39	40	83
	50,000	34	2	67	66	103	2	76	74	129
MIN/MAX	10,000	26	0	26	24	−4	0	23	27	−4
	50,000	36	0	38	34	0	0	36	33	−4

of secret values to be recovered is equal to the number of groups, which is usually a small number. Thus, the client-side execution times of these three types of queries are expected to be close to zero. Some observations in this experiment are as follows:

- There is no meaningful difference between the server-side execution times of the proposed scheme on the two data servers in all cases.
- For GROUPING and MIN/MAX queries, the server-side execution times of both the baseline and proposed schemes on the attribute *student_id* are a little

greater than those on the attribute *grade*. The reason is that in the same number of tuples of the relation *takes*, the number of distinct values of the attribute *student_id* is usually greater than that of the attribute *grade*. Also, for RNG^{SR} and GROUPING queries, the client-side execution times of the proposed scheme on the attribute *student_id* are greater than those on the attribute *grade*. In the other cases, the differences between the execution times of the proposed scheme on the two attributes are negligible.

- For EQU, RNG^{SR}, AGG, and MIN/MAX queries, there is no meaningful difference between both the client-side and server-side execution times of the proposed scheme with $L = 50$ and those with $L = 200$. In contrast, for GROUPING and SUM/AVG queries, both the client-side and server-side execution times of the proposed scheme with $L = 200$ are a little greater than those with $L = 50$. The reason is that the number of subgroups in the translated GROUPING and SUM/AVG queries is proportional to the value of L.
- For RNG, AGG, and MIN/MAX queries, the differences between the server-side execution times of the proposed scheme and those of the baseline scheme are negligible. This is because the translated queries corresponding to these three types of queries are very similar to their original queries. On the other hand, for EQU, GROUPING, and SUM/AVG queries, the server-side execution times of the proposed scheme are greater than those of the baseline scheme. The reasons include: (1) each exact-match predicate is translated into a range predicate, (2) each translated GROUPING query is performed with a larger number of groups compared to the original query, and (3) each translated SUM/AVG query is performed with grouping on an extra attribute.

In this experiment, the distribution of the server-side execution times of queries of each type by the proposed scheme based on the results of queries with the query set size of 50,000 on the shares columns associated with the attribute *grade* with $L = 200$ was considered. Figure 4 shows five box plots; each box plot represents the mean (average), one standard deviation above and one below the mean, and the minimum and maximum of the server-side execution times of one type of query.

In the fourth experiment, the proposed scheme was compared with the baseline and existing SSDO schemes. For each type of query, the same set of queries was executed by all the schemes, and the average client-side and server-side execution times were obtained. Table 5 compares the execution times of the baseline scheme, EMAA, HJEG, FTI, and the proposed scheme for various types of queries and two query set sizes. In Table 5, *DS* denotes the data server with the maximum execution time of SUM/AVG queries or the randomly selected data server for the other types of queries. Because of the one-to-one mapping of the secret values to the tuples of shares in EMAA, the results of this scheme are nearly the same as those of the baseline scheme. In HJEC, every exact-match or range predicate is usually translated into a disjunction of multiple range predicates. The larger the number of range predicates, the greater the server-side execution time. HJEC supports none of the four types of aggregate queries. The large client-side execution time of HJEC is due to the false hit elimination step by performing

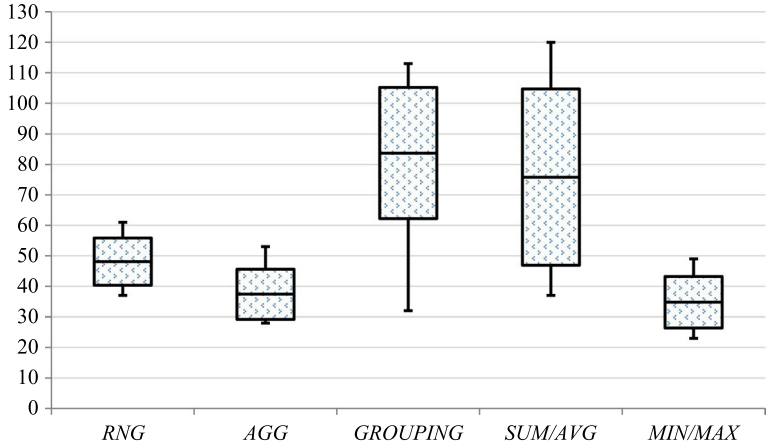

Fig. 4 The distribution of the server-side execution times of the proposed scheme (ms)

Table 5 Comparison between the execution times of the baseline and various SSDO schemes (ms)

Type of query	Query set size	Baseline	EMAA		HJEC		FTI		Proposed scheme	
		DS	Client	*DS*	Client	*DS*	Client	*DS*	Client	*DS*
EQU	10,000	26	0	27	59	166	171	38	0	36
	25,000	29	0	29	93	212	417	39	0	37
RNG^{SR}	10,000	39	2	38	61	245	172	41	2	39
	50,000	45	8	46	167	303	829	47	8	48
AGG	10,000	38	0	39	N/A	N/A	N/A	N/A	0	38
	50,000	38	0	38	N/A	N/A	N/A	N/A	0	37
GROUPING	10,000	25	0	26	N/A	N/A	N/A	N/A	2	33
	50,000	61	0	60	N/A	N/A	N/A	N/A	2	84
SUM/AVG	10,000	23	0	25	N/A	N/A	N/A	N/A	2	40
	50,000	34	0	33	N/A	N/A	N/A	N/A	2	76
MIN/MAX	10,000	26	0	26	N/A	N/A	26	81	0	25
	50,000	36	0	37	N/A	N/A	39	126	0	35

an intersection query on the results received from the two data servers. For FTI, just 10% fake tuples were inserted into the outsourced relations. The original FTI scheme cannot support AGG, GROUPING, and SUM/AVG queries. The large client-side execution time of FTI is due to the false hit elimination step by decrypting the tags of the tuples in the query result. In summary, HJEC and FTI support only limited types of queries with extensive client-side computations. Although the execution times of the proposed scheme are greater than those of both the baseline scheme and EMAA, its computation overhead is not significant, and this little amount of overhead is acceptable for achieving a great level of security.

5.2 Communication costs

In this study, the communication cost of a query is evaluated based on the number of tuples transferred between the client and the data servers to process the query. We define the communication overhead factor of a secure data outsourcing scheme for a query as the ratio of the number of tuples transferred between the client and all the data servers to process the query by that scheme to that by the baseline scheme. Thus, the communication overhead factor of 1 indicates no overhead. Table 6 shows the communication overhead factors of EMAA with a (K, N) threshold, HJEC with a $(2, N)$ threshold, FTI with a $(2, N)$ threshold, and the proposed scheme with a (K, N) threshold for various types of queries in the single-server and multi-server (with N data servers) models. Let the ratio of the actual tuples to the inserted fake tuples in FTI be denoted by $1 : \tau$.

Because of the one-to-one mapping of the secret values to the tuples of shares in EMAA, the communication overhead factor of EMAA with the single-server model for any type of query is 1. Also, the communication overhead factors of EMAA with the multi-server model for queries with the interpolation-based recovery method including RNG^{IR} and SUM/AVG and for data manipulation statements including INSERT and UPDATE are K and N, respectively. The differences between the other schemes and EMAA are as follows. In HJEC, the communication overhead factors with the multi-server model for EQU and RNG^{SR} queries are 2 because of the need for false hit elimination. Also, HJEC supports neither UPDATE statements nor any type of aggregate queries. Because of fake tuples in FTI, the communication overhead factor for every supported type of query in FTI is, on average, $1 + \tau$ times that in EMAA. It should be noted that in addition to the overhead mentioned here, FTI may need several rounds of interaction between the client and data servers to process a MIN/MAX query. The only additional communication cost of the proposed scheme compared to EMAA is that the communication overhead factors for

Table 6 The communication overhead factors of various SSDO schemes

Type of query	EMAA		HJEC		FTI		Proposed scheme	
	Single-server model	Multi-server model	Single-server model	Multi-server model	Single-server model	Multi-server model	Single-server model	Multi-server model
EQU	1	1	1	2	$1+\tau$	$1+\tau$	1	1
RNG^{IR}	1	K	1	2	$1+\tau$	$2+2\tau$	1	K
RNG^{SR}	1	1	1	2	$1+\tau$	$1+\tau$	1	1
INSERT	1	N	1	N	$1+\tau$	$N+N\tau$	1	N
UPDATE	1	N	N/A	N/A	$1+\tau$	$N+N\tau$	1	N
AGG	1	1	N/A	N/A	N/A	N/A	1	1
GROUPING	1	1	N/A	N/A	N/A	N/A	L	L
SUM/AVG	1	K	N/A	N/A	N/A	N/A	L	$K.L$
MIN/MAX	1	1	N/A	N/A	$1+\tau$	$1+\tau$	1	1

GROUPING and SUM/AVG queries in the proposed scheme are L times those in EMAA. The reason is that for such queries, the number of subgroups that belong to each original group in the proposed scheme is at most L.

5.3 Storage costs

In order to outsource a relation by a (K, N)-threshold SSDO scheme, each sensitive attribute is split into N share columns of a domain with usually a larger size. In addition, in a multi-server model, the primary key attribute(s) and the other searchable attributes of the original relation should be replicated on all the data servers. Consider the relation *takes* with the attributes *year*, *semester*, *section_id*, *student_id*, and *grade* of the SQL data types *smallint*, *byte*, *smallint, int*, and *byte*, respectively. Here, the domain of the attribute *grade* is assumed to be [1 … 200]. The SQL data types *byte*, *smallint*, *int*, and *bigint* occupy 1, 2, 4, and 8 bytes of storage, respectively. The storage overhead is defined as the total size of the outsourced relations to the size of the original relation. Here, the size of the indexes created and managed by the database management system is not considered. In the original relation *takes*, each tuple occupies 10 bytes of storage. With the setting at the beginning of this section, the data type *int* is adequate for the share columns associated with each of the sensitive attributes *student_id* and *grade*. The total size of the tuple(s) in the outsourced relation(s) corresponding to each tuple in the original relation *takes* by any SSDO scheme considered in the experiments with a (K, N)-threshold in the single-server or N-server model will be $5 + 8N$ or $13N$, respectively. That is, the storage overhead by a $(2, 2)$-threshold SSDO scheme to outsource this relation in the single-server or two-server model is 110% or 160%, respectively. In FTI [42], the insertion of fake tuples into the outsourced relations and adding the column *Tag* to the outsourced relations should also be considered as other storage overheads.

6 Conclusion

Data encryption, data fragmentation, and secret sharing are the most popular approaches to provide content confidentiality of outsourced data. However, there are some limitations to the use of data encryption and data fragmentation schemes. Secret sharing is computationally efficient and supports server-side query execution of a wide variety of queries. However, as analyzed in this paper, the existing SSDO schemes are vulnerable to some attacks and inferences. In this paper, two practical attacks on some SSDO schemes (referred to as the SRB-based and DPSV-based attacks) were first introduced, and several other security and performance issues with the existing SSDO schemes were also explored. The existing and new attacks exploit the information about the share range boundaries or the correspondences between the secret values and shares. A range expansion technique was then proposed to thwart the SRB-based attack. The proposed range expansion technique expands the ranges in every range predicate in the submitted queries in order to hide the share range boundaries of the original values from any query observer. Next, a

mapping method was proposed to thwart the other attacks. The proposed mapping method maps each secret value to a mapping value using a secret one-to-many mapping with a finite set of linear rules so that the tuples of shares are generated from the mapping values rather than directly from the secret values. The proposed mapping method works as an additional layer of security and addresses any attack based on the correspondences between the secret values and shares. At the same time, it preserves the homomorphism property of secret sharing. Finally, a new secure data outsourcing scheme was elaborated on secret sharing, the proposed mapping method, and the proposed range expansion technique. As the mapping rules are kept secret, even if an adversary knows the secret values corresponding to some tuples of shares, he or she cannot perform the attack introduced by Dautrich and Ravishankar [43]. In addition, since the proposed scheme does not generate distinct pairs (tuples) of shares from the same secret value, it is resistant to both the DPSV-based attack and Ghasemi's attack [42]. Furthermore, it is resistant to some inferences. It was shown that the proposed scheme supports the fully server-side or a partially server-side query execution of most types of queries. The experimental results confirmed that the proposed scheme is quite practical and efficient.

References

1. di Vimercati SDC, Foresti S, Paraboschi S et al (2011) Efficient and private access to outsourced data. In: Proceedings of 2011 31st International Conference on Distributed Computing Systems, pp 710–719. https://doi.org/10.1109/ICDCS.2011.37
2. Hong J, Wen T, Guo Q et al (2019) Privacy protection and integrity verification of aggregate queries in cloud computing. Cluster Comput 22:5763–5773. https://doi.org/10.1007/s10586-017-1521-0
3. Neela KL, Kavitha V (2022) An improved RSA technique with efficient data integrity verification for outsourcing database in cloud. Wirel Pers Commun. https://doi.org/10.1007/s11277-021-09248-8
4. Shynu PG, Nadesh RK, Menon VG et al (2020) A secure data deduplication system for integrated cloud-edge networks. J Cloud Comput. https://doi.org/10.1186/s13677-020-00214-6
5. Hesamifard E, Takabi H, Ghasemi M, Jones C (2017) Privacy-preserving machine learning in cloud. In: Proceedings of the 2017 on Cloud Computing Security Workshop, pp 39–43. https://doi.org/10.1145/3140649.3140655
6. Zhou L, Zhu Y, Castiglione A (2017) Efficient k-NN query over encrypted data in cloud with limited key-disclosure and offline data owner. Comput Secur 69:84–96. https://doi.org/10.1016/j.cose.2016.11.013
7. Kim H-J, Lee H, Kim Y-K, Chang J-W (2022) Privacy-preserving kNN query processing algorithms via secure two-party computation over encrypted database in cloud computing. J Supercomput. https://doi.org/10.1007/s11227-021-04286-2
8. Rong H, Liu J, Wu W et al (2020) Toward fault-tolerant and secure frequent itemset mining outsourcing in hybrid cloud environment. Comput Secur. https://doi.org/10.1016/j.cose.2020.101969
9. Song DX, Wagner D, Perrig A (2000) Practical techniques for searches on encrypted data. In: Proceeding of the 2000 IEEE Symposium on Security and Privacy (S&P 2000), pp 44–55. https://doi.org/10.1109/SECPRI.2000.848445
10. Xu L, Weng C-Y, Yuan L-P et al (2018) A shareable keyword search over encrypted data in cloud computing. J Supercomput 74:1001–1023. https://doi.org/10.1007/s11227-015-1515-8
11. Miao M, Wang J, Wen S, Ma J (2019) Publicly verifiable database scheme with efficient keyword search. Inf Sci (NY) 475:18–28. https://doi.org/10.1016/j.ins.2018.09.067
12. Noroozi M, Eslami Z (2019) Public-key encryption with keyword search: a generic construction secure against online and offline keyword guessing attacks. J Ambient Intell Humaniz Comput 11:879–890. https://doi.org/10.1007/s12652-019-01254-w

13. Hacigümüş H, Iyer B, Li C, Mehrotra S (2002) Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp 216–227. https://doi.org/10.1145/564691.564717
14. Hore B, Mehrotra S, Tsudik G (2004) A privacy-preserving index for range queries. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases-Volume 30, pp 720–731
15. Mei Z, Zhu H, Cui Z et al (2018) Executing multi-dimensional range query efficiently and flexibly over outsourced ciphertexts in the cloud. Inf Sci (NY) 432:79–96. https://doi.org/10.1016/j.ins.2017.11.065
16. Agrawal R, Kiernan J, Srikant R, Xu Y (2004) Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp 563–574. https://doi.org/10.1145/1007568.1007632
17. Popa RA, Li FH, Zeldovich N (2013) An ideal-security protocol for order-preserving encoding. In: Proceedings of the 2013 IEEE Symposium on Security and Privacy, pp 463–477. https://doi.org/10.1109/SP.2013.38
18. Yang C, Zhang W, Yu N (2017) Semi-order preserving encryption. Inf Sci (NY) 387:266–279. https://doi.org/10.1016/j.ins.2016.12.025
19. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Stern J (eds) Advances in cryptology — EUROCRYPT '99. Lecture notes in computer science, vol 1592, pp 223–238. https://doi.org/10.1007/3-540-48910-X_16
20. Ding W, Yan Z, Deng RH (2017) Encrypted data processing with homomorphic re-encryption. Inf Sci (NY) 409–410:35–55. https://doi.org/10.1016/j.ins.2017.05.004
21. Liu X, Choo K-KR, Deng RH et al (2018) Efficient and privacy-preserving outsourced calculation of rational numbers. IEEE Trans Dependable Secur Comput 15:27–39. https://doi.org/10.1109/tdsc.2016.2536601
22. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, pp 169–178. https://doi.org/10.1145/1536414.1536440
23. Brakerski Z, Vaikuntanathan V (2011) Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In: Rogaway P (eds) Advances in cryptology – CRYPTO 2011. Lecture notes in computer science, vol 6841, pp 505–524. https://doi.org/10.1007/978-3-642-22792-9_29
24. Naehrig M, Lauter K, Vaikuntanathan V (2011) Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, pp 113–124. https://doi.org/10.1145/2046660.2046682
25. Aggarwal G, Bawa M, Ganesan P, et al (2005) Two can keep a secret: a distributed architecture for secure database services. In: The Second Biennial Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, California
26. Mondal A, More Y, Ramachandran P, Panda P, Virk H, Gupta D (2022) Scotch: an efficient secure computation framework for secure aggregation. arXiv preprint. arXiv:2201.07730
27. Li J, Makkonen O, Hollanti C, Gnilke OW (2022) Efficient recovery of a shared secret via cooperation: applications to SDMM and PIR. IEEE J Sel Areas Commun. https://doi.org/10.1109/jsac.2022.3142366
28. Yu K, Tan L, Yang C et al (2021) A blockchain-based Shamir's threshold cryptography scheme for data protection in industrial Internet of Things Settings. IEEE Internet Things J. https://doi.org/10.1109/jiot.2021.3125190
29. Liu Y, Zhao Q (2018) E-voting scheme using secret sharing and K-anonymity. World Wide Web 22:1657–1667. https://doi.org/10.1007/s11280-018-0575-0
30. Tejedor-Romero M, Orden D, Marsa-Maestre I et al (2021) Distributed remote E-voting system based on Shamir's secret sharing scheme. Electronics. https://doi.org/10.3390/electronics10243075
31. Framner E, Fischer-Hübner S, Lorünser T et al (2019) Making secret sharing based cloud storage usable. Inf Comput Secur 27:647–667. https://doi.org/10.1108/ics-01-2019-0016
32. Shamir A (1979) How to share a secret. Commun ACM 22:612–613. https://doi.org/10.1145/359168.359176
33. Hadavi MA, Jalili R. Secure data outsourcing based on threshold secret sharing; towards a more practical solution. In: Proceedings of the Very Large Data Bases PhD Workshop 2010, pp 54–59
34. Hadavi MA, Noferesti M, Jalili R, Damiani E (2012) Database as a service: towards a unified solution for security requirements. In: Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops, pp 415–420. https://doi.org/10.1109/COMPSACW.2012.79

35. Tian X, Sha C, Wang X, Zhou A (2011) Privacy preserving query processing on secret share based data storage. In: Yu JX, Kim MH, Unland R (eds) Database systems for advanced applications. DASFAA 2011. Lecture notes in computer science, vol 6587, pp 108–122. https://doi.org/10.1007/978-3-642-20149-3_10
36. Xiang T, Li X, Chen F et al (2016) Processing secure, verifiable and efficient SQL over outsourced database. Inf Sci (NY) 348:163–178. https://doi.org/10.1016/j.ins.2016.02.018
37. Agrawal D, El Abbadi A, Emekci F, Metwally A (2009) Database management as a service: challenges and opportunities. In: Proceedings of the 2009 IEEE 25th International Conference on Data Engineering, pp 1709–1716. https://doi.org/10.1109/ICDE.2009.151
38. Agrawal D, El Abbadi A, Emekci F, Metwally A, Wang S (2011) Secure data management service on cloud computing infrastructures. In: Agrawal D, Candan KS, Li WS (eds) New Frontiers in Information and Software as Services. Lecture Notes in Business Information Processing, vol 74, pp 57–80. https://doi.org/10.1007/978-3-642-19294-4_3
39. Emekci F, Methwally A, Agrawal D, El AA (2014) Dividing secrets to secure data outsourcing. Inf Sci (NY) 263:198–210. https://doi.org/10.1016/j.ins.2013.10.006
40. Hadavi MA, Damiani E, Jalili R, Cimato S, Ganjei Z (2013) AS5: A secure searchable secret sharing scheme for privacy preserving database outsourcing. In: Di Pietro R, Herranz J, Damiani E, State R (eds) Data privacy management and autonomous spontaneous security. DPM SETOP 2012. Lecture notes in computer science, vol 7731, pp 201–216. https://doi.org/10.1007/978-3-642-35890-6_15
41. Hadavi MA, Jalili R, Damiani E, Cimato S (2015) Security and searchability in secret sharing-based data outsourcing. Int J Inf Secur 14:513–529. https://doi.org/10.1007/s10207-015-0277-x
42. Ghasemi R (2019) Resolving a common vulnerability in secret sharing scheme–based data outsourcing schemes. Concurr Comput Pract Exp. https://doi.org/10.1002/cpe.5363
43. Dautrich JL, Ravishankar CV (2012) Security limitations of using secret sharing for data outsourcing. In: Cuppens-Boulahia N, Cuppens F, Garcia-Alfaro J (eds) Data and applications security and privacy XXVI. DBSec 2012. Lecture notes in computer science, vol 7371, pp 145–160. https://doi.org/10.1007/978-3-642-31540-4_12

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.