This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

An Anonymity Retaining Framework for Multi-party Skyline Queries Based on Unique Tags

Dola Das, Kazi Md. Rokibul Alam, and Yasuhiko Morimoto

Abstract—While finding out dominant objects over multiple competing parties' sensitive datasets via multi-party skyline queries (MSQs), upholding ample data privacy with integrity, data and data owners' (*i.e.*, parties) anonymity, verifiable deeds of entities, the robustness of framework, etc., is crucial. The existing works consider these criteria trivially or partially and may sacrifice data integrity due to malleable cipher data. This paper proposes a framework for MSQs that incorporates these requirements together and eventually makes a ranking of parties' data-objects. To enrich data privacy, it allocates separate decryption keys among mutual entities. To confirm data integrity, it unites encrypted unique tags (*UT*s) to parties' encrypted datasets individually. To assure data and parties' anonymity, it re-encrypts and shuffles encrypted datasets, and uses anonymous credentials, respectively. These qualities and publicly exposed data reinforce the framework and assure the correct deeds of entities. Finally, evaluation results, analyses, comparisons with state-of-the-art works, etc., prove the efficacy of the proposed framework.

Index Terms—Multi-party skyline queries, ElGamal encryption, Commutative cryptosystem, Homomorphic encryption, Mix-net

1 Introduction

Among multiple commercial organizations with a mutual interest, *e.g.*, resorts, real estate, hotels, hospitals, etc., the information on well-served or dominant objects bears a huge interest for the concerned customers. For business purposes, these organizations intend to defeat one another, but at the same time, they prefer not to disclose their datasets publicly. Herein, secrecy-retaining multi-party skyline queries (MSQs) [1] retrieve dominance relation sets from sensitive datasets of the involved organizations. Since here the safeguarding of data and data owners (*i.e.*, parties) is of the highest priority, retaining adequate data privacy and data integrity together, data and parties' anonymity, auditable deeds of entities, the robustness of the framework, etc., is essential. While dealing with sensitive datasets, it is dicey to consider entities involved in MSQs frameworks to be trustworthy. Because, by retaining apparent data privacy, an evil entity may avail or endow illegal benefits by altering the cipher data, *i.e.*, by breaching data integrity. Hence, upholding data integrity is vital to data privacy. In addition, when data and parties' anonymity are not retained, the links between the encrypted data and their decrypted forms, the party as well as its encrypted dataset, etc., can be traceable. Thus, coercers can identify the link between the retrieved data-objects and their parties. Besides, to ensure robustness, *i.e.*, to protect the framework from disruption, publicly exposing the deeds of entities are advantageous.

Existing works of secure MSQs usually do not satisfy all these requirements together. Besides, trusting a single entity to conduct privacy-related operations lead to gaining feeble data privacy or data security that can cause security flaw [25]. Namely, many secure MSQs hugely assign a single decryption key for every party to retrieve their secure datasets and thus ensure feeble privacy [24]. Also, running various operations by engaging the involved parties rather than the external entities yield the framework design impractical [25]. Nonetheless, many works rarely appoint external entities for these purposes [24]. In addition, while multiple untrustworthy entities deal with cipher data among them, the evil usages of the malleable property of many cryptosystems [18] may render incorrect decryption results via a breach of data integrity [6, 15, 22]. Alongside, many secure MSQs sensibly do not consider data anonymity plus parties' anonymity [5]. Moreover, many MSQs put petty or unrealistic attention regarding verifiability [5, 23]. Although some works assure the deeds of entities as verifiable via zero-knowledge proofs (ZKPs) [22], ZKPs are complicated and expensive. Due to these reasons, secure MSQs' many existing frameworks are not practical and efficient enough.

To eliminate these limitations, this paper proposes a framework for secure MSQs [3] that is capable enough to satisfy the pre-mentioned requirements. Specifically, it bestows ample data privacy, data integrity, data as well as parties' anonymity, the auditable deeds of entities, and ultimately, a ranking among parties' retrieved data-objects. Elaborately, it (*i*) allocates separate decryption keys among mutual entities to enrich the data privacy; (*ii*) attaches encrypted unique tags (*UT*s) with parties' encrypted datasets separately to confirm data integrity; (*iii*) executes re-encryption and shuffle operations over encrypted datasets to ensure data anonymity; (*iv*) deploys anonymous credentials to attain parties' anonymity; (*v*) exposes the deeds of entities publicly to assure auditability; and (*vi*) introduces a two-stage procedure to make a ranking among parties' decrypted data-objects. Thus, the developed framework becomes robust, practical, and efficient enough.

- *Dola Das and Kazi Md. Rokibul Alam are with the Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna 9203, Bangladesh.*
  E-mail: dola.das@cse.kuet.ac.bd, rokib@cse.kuet.ac.bd.
- *Yasuhiko Morimoto is with the Graduate School of Advanced Science and Engineering, Hiroshima University, Higashi-Hiroshima 739-8521, Japan (e-mail:morimo@hiroshima-u.ac.jp).* *He is supported by KAKENHI (20K11830) Japan, which partially supported this work.*

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

The remaining portion of this paper is organized as follows. Section 2 reviews the related works; Section 3 describes the required cryptographic tools; Section 4 illustrates an overview of the secure MSQs, the system model, the role of entities, and the distinct stages of the proposed framework; Section 5 presents the evaluation of the framework; and finally, Section 6 concludes the paper.

2 Related Works

Major categories of existing skyline queries along with their variants are based on: the number of parties (*e.g.*, two-party [10], multi-party [11]); the mode of authority (*e.g.*, no authority [24], trusted [23], semi-honest [5]); the cloud environment (*e.g.*, centralized [6, 26], distributed [27]); the data model (*e.g.*, incomplete [12], complete [21], big data [14], dynamic data [13]); targeted inquiry (*e.g.*, particulars of a system [5, 27], mobile crowd-sensing [4]), etc. While assessing MSQs for confidential datasets, due to the intrinsic demands, cutting-edge research puts attention to privacy and securty-related aspects.

For example, by viewing both single and multiple servers as a unit server and thus offering a centralized cloud environment, a group of secure MSQs has been proposed. Namely, the queries in [6] treat encrypted data, appoint a semi-honest authority, and deploy a cloud platform consisting of two non-colluding servers. Thus, it estimates the dominance relationships valid for both skyline and other queries. Queries in [19, 20] though considering the encrypted data, provide only an informal security guarantee. The query in [26] designs lightweight predicate encryption schemes to support user-defined queries over encrypted data and employs a single cloud server to eliminate communication costs. The query in [21] chooses publicly accessible data, engages an untrusted server, and uses the neighbor relationships among the points of interest to propose a verifiable location-based query.

Likewise, by setting multiple distinct servers to run the tasks of different entities, the distributed environment of the skyline queries is arranged. For instance, queries in [14-16] particularly use the homomorphic cryptosystem. Alongside, the skyline in [14] develops a vector dominance protocol between two encrypted vectors. The query in [17] uses the Paillier cryptosystem to develop an online-based medical decision support system. The MSQs in [27] utilize lightweight cryptography to support subsequent secure query processing through the hosting of two separate servers. Instead of secure MSQs, the skyline in [2] processes the query over the distributed databases. It prunes the redundant tuples to filter the data and computes the global skyline result across parallel computing.

Alongside, some other secure queries proposed in [5, 23, 24] adopt various trust models and also exploit homomorphic encryption. The MSQs in [24] deploy a semi-honest authority model and assign a distinct encryption-decryption key pair for every involved party where it engages the parties accordingly to encrypt and decrypt their private datasets. But, visibly it doesn't consider retaining data integrity or performing re-encryption to attain data anonymity. The MSQs in [5] also consider a semi-honest authority, deploy two non-colluding cloud servers, introduce a new method called 'blind-read' to obtain the skyline point, and exploit two separate secret keys hold by two entities for data decryption purposes. Besides, to retain data anonymity it adds random noise and permutes data, but doesn't consider data integrity clearly. However, any unwanted homomorphic encryption operation in these works by involved parties or corrupt entities may cause to breach of the data integrity.

Unlike the above works, the proposed framework fulfills the essential requirements of secure MSQs. To retain ample privacy it adopts a commutative cryptosystem to allocate the decryption keys among mutually independent entities. To confirm data integrity it develops encrypted unique tags (*UT*s) to unite them separately to parties' encrypted datasets. To ensure data anonymity it employs a mix-net consisting of multiple mix-servers to re-encrypt and shuffle the ciphertext of parties' datasets. To attain parties' anonymity it exploits anonymous credentials to enable parties to conceal their identities. To assure auditability it deploys web bulletin boards (*WBB*s) [22] to expose the acts of entities publicly. Lastly, it generates a ranking for all retrieved data-objects.

3 Cryptographic Tools

This section describes major cryptographic tools required to develop the framework, *i.e.*, (*i*) ElGamal encryption-based a commutative cryptosystem, (*ii*) Re-encryption mix-net, (*iii*) Unique tags (*UT*s), (*iv*) Anonymous credentials, etc. In the remainder, it is assumed that P (≥ 2) mix-servers ($M_i \in \{M_1, \ldots, M_P\}$) comprise the mix-net, entity SB is a representative of the mix-net, and N (≥ 2) parties' ($PA_n \in \{PA_1, \ldots, PA_N\}$) possess datasets that will be used to calculate the secure MSQs where the role of involved entities including these will be explained in Section 4.3.

3.1 ElGamal Encryption

The commutative cryptosystem adopted herein is based on the ElGamal cryptosystem that consists of (*i*) key generation, (*ii*) encryption, and (*iii*) decryption phases. Thus, the sender can encrypt its message m, and later on, the receiver can retrieve m. Since the ElGamal cryptosystem is already presented in [3], it is not described further.

TABLE 1
ORGANIZATION OF KEYS OF THE COMMUTATIVE CRYPTOSYSTEM

Mutual mix-servers	$M_1, \ldots, M_P$
Private decryption keys	$X_1, \ldots, X_P$
Corresponding public keys	$Y_1, \ldots, Y_P$
Combined encryption key	$Y_* = Y_1 Y_2 \ldots Y_P$

3.2 ElGamal-based Commutative Cryptosystem

A commutative cryptosystem [8] is akin to the decryption mix-net [9] and consists of mutually independent mix-servers. ElGamal-based commutative cryptosystem enables each mix-server to own a separate private decryption key, and other mix-servers are unable to know it. Generally, it performs the data encryption through a single key same as the ElGamal cryptosystem. But while retrieval, it executes the repeated decryption operations by mix-

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

servers through their separate decryption keys. Thus it enriches data privacy. Here, the phases are identical to the ElGamal cryptosystem and are described below.

1) Key generation phase: At first, the mix-net declares a large prime p and a generator g of the multiplicative group Z_p^* of the integers modulo p. Now, each mix-server M_i ($1 < i \le P$) chooses its private decryption key X_i and calculates the corresponding public key $Y_i = g^{Xi} \pmod p$ to disclose it publicly. Later on, the combined encryption key Y_* is calculated as $Y_* = Y_1Y_2\ldots Y_P \pmod p = g^{X1+\ldots+XP} \pmod p$, and they are summarized in Table 1.

2) Encryption phase: Now using a secret random integer k ($1 < k < p-2$), the data owner encrypts its message m ($0 < m < p-1$) as $E_{Y*}(k, m) = (C_1, C_2)$, where $\{C_1 = g^k \pmod p, C_2 = m.Y_*^k \pmod p\}$ to send (C_1, C_2) to the mix-net.

3) Decryption phase: To retrieve the message m, mix-servers decrypt (C_1, C_2) as $m = \{C_2 \times C_1^{[-(\sum_{i=1}^{P} X_i)]}\} \pmod p$, by using their decryption keys $\{X_1, \ldots, X_P\}$ repeatedly. In the remainder, notation (mod p) will be excluded.

TABLE 2
AN OVERVIEW OF THE RE-ENCRYPTION MIX-NET

Input	A group of the encrypted message
Operations	Re-encryption and shuffling
Consequence	Messages in new form and position
Output	A group of the re-encrypted message

3.3 Re-encryption Mix-net

Usually, a re-encryption mix-net [9] accepts a group of encrypted messages as the input, then re-encrypts and shuffles them to change the fashion and re-arrange the positions, respectively. Thus, it maintains untraceability between incoming and outgoing messages, *i.e.*, attains data anonymity. For ElGamal-based re-encryption mix-net to perform a re-encryption operation, no knowledge about either the encryption key or decryption keys is required. Table 2 presents an overview of this mix-net. Here, the encryption process is based on Section 3.2-2. The re-encryption operation is performed as below.

As already mentioned, it receives a group of ciphertext. Now, mix-servers re-encrypt every encrypted message, *i.e.*, a $E_{Y*}(k, m) = \{g^k, m.Y_*^k\}$ into $E_{Y*}(r_*, m)$ as $E_{Y*}(r_*, m) = \{(g^k.(g^{r_1}\ldots g^{r_P})), (m.Y_*^k.(Y_*^{r_1}\ldots Y_*^{r_P}))\} = \{(g^k.g^{\sum_{i=1}^{P} r_i}), (m.Y_*^k.Y_*^{\sum_{i=1}^{P} r_i})\}$ by using each M_i's secret random integer r_i. At the same time, each M_i eventually shuffles the re-encrypted messages. Thereby, the last mix-server M_P outputs $E_{Y*}(r_*, m)$ where $r_* = k + \sum_{i=1}^{P} r_i$.

Usually, it does not deploy the decryption phase. But while message retrieval, the decryption process (as will be in Section 4.4.4) integrates Section 3.2-3.

3.4 Unique Tags (*UT*s)

Being inspired by [15, 22], the *UT*s are devised. They are (*i*) unique and registered integers, (*ii*) generated and disclosed publicly in advance, (*iii*) denoted as $\{U_1, \ldots, U_M\}$ where $\{U_{nj} \in (U_1, \ldots, U_M)\}$, and (*iv*) divulged both in plain and encrypted forms. As will be discussed in Section 4.4, by being attached individually to parties' encrypted datasets, allied *UT*s confirm data integrity. Alongside, while the doings of entities along with *UT*s are publicly exposed, they confirm the correctness of the entities' deeds. While encrypting *UT*s, the commutative cryptosystem as well as the re-encryption and shuffle operations discussed in Section 3.2 and Section 3.3, respectively are incorporated. Thereby, no one can identify any link between U_{nj} and its encryption form $E_{Y*}(r_{*nj}, U_{nj})$. The encryption process is as follows, and Table 3 depicts their formations.

1. The entity *SB* generates and publicly discloses a finite number of *UT*s in plain form, *i.e.*, $U_1, \ldots, U_M$.
2. Using its secret random integer $r_{nj(i)}$, mix-server M_i encrypts $E_{Y*}(r_{*nj(i-1)}, U_{nj})$ received from M_{i-1} into $\{E_{Y*}(r_{*nj(i)}, U_{nj}) = (g^{r*nj(i-1)} g^{r_{nj(i)}}, U_{nj} Y_*^{r*nj(i-1)} Y_*^{r_{nj(i)}})\}$. After shuffling, they are sent to M_{i+1}. Thus, M_P yields $E_{Y*}(r_{*nj}, U_{nj}) = (g^{r*nj}, U_{nj}.Y_*^{r*nj})$ where $r_{*nj} = r_{nj(0)} + r_{nj(1)} + \ldots + r_{nj(P)}$.
3. While re-encryption, M_i uses a secret random integer $x_{nj(i)}$ to re-encrypt into $E_{Y*}(r_{*nj} + x_{*nj(i-1)}, U_{nj})$ received from M_{i-1}. Finally, M_P produces $E_{Y*}(r_{*nj} + x_{*nj}, U_{nj}) = \{g^{(r*nj + x*nj)}, U_{nj}.Y_*^{(r*nj + x*nj)}\}$ where $x_{*nj} = x_{nj(1)} + \ldots + x_{nj(P)}$.
4. Later on, mix-servers decrypt both $E_{Y*}(r_{*nj} + x_{*nj}, U_{nj})$ and $E_{Y*}(r_{*nj}, U_{nj})$ (as will be in Sections 4.4.4 and 4.4.5), respectively to disclose them.

TABLE 3
THE FORMATION OF *UT*S

Forms of *UT*	Depiction
Plain form	$U_{nj} \in (U_1, \ldots, U_M)$
Encrypted form	$E_{Y*}(r_{*nj}, U_{nj})$
Re-encrypted form	$E_{Y*}(r_{*nj} + x_{*nj}, U_{nj})$
No link exists among U_{nj}, $E_{Y*}(r_{*nj}, U_{nj})$ and $E_{Y*}(r_{*nj} + x_{*nj}, U_{nj})$	

3.5 Anonymous Credentials

Although the mix-net hides links between incoming and outgoing datasets, ensuring parties' anonymity is required too. Otherwise, anyone including the entity *SB* can identify them. Hence, the proposed framework adopts the anonymous credential mechanism developed in [7] that works as follows.

1. Through in-person communication, the party PA_n obtains a credential C_n from *SB*. Thereby, any other entity cannot impersonate PA_n.
2. PA_n calculates $AC_n = C_n^{Tn}$ (*mod b*). Here, T_n is a secret integer of PA_n and b is a publicly known appropriate integer. Thereby, no one except PA_n can link between C_n and AC_n and PA_n can preserve its anonymity. Later on, notation (*mod b*) is omitted.
3. PA_n keeps another secret integer W_n, includes it in C_n, and convinces others of its eligibility by computing values that become consistent with AC_n only by integer W_n without disclosing W_n itself.
4. In addition, to make evidence that PA_n had shown AC_n, entities can force PA_n to calculate the seal V^{Wn} using W_n in C_n honestly which is unique to each C_n from a given public integer V.

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

4 Proposed Framework for MSQs

This section develops the framework, *i.e.*, explains an overview of secure MSQs, the system model, the role of entities, and individual stages of the framework. The used notations (*i.e.*, major ones) are summarized in Table 4.

TABLE 4
LIST OF USED NOTATIONS

Notation	Description
N (≥ 2), P (≥ 2)	Number of parties, and independent mix-servers
PA_n, ID_n, C_n	n-th party, PA_n's identity, and PA_n's credential
DA_n, $\underline{DA}_n$	PA_n's dataset in plain and encrypted form
D, L_n, DA_{nj}	The highest dimension of DA_n, the total number of data records of DA_n, and j-th data within DA_n
M_i, SB	i-th mix-server, and the system manager
X_i , $Y_i = g^{X_i}$	Private decryption key, and the public key of M_i
$Y_* = Y_1Y_2…Y_P$	The combined public encryption key
$E_{Y^*}(k, m) = (g^k, m.Y_*{}^k)$	Encryption of message m under ElGamal cryptosystem using Y_* and secret integer k
UT_{nj}, $E_{Y^*}(r_{*nj}, UT_{nj})$	UT for PA_n's j-th data, and its encrypted form
$(k_{nj}, r_{nj(i)})$, (s_i, t_i)	Secret randoms used to encrypt and re-encrypt data
UT_n, $\underline{UT}_n$	UTs required for PA_n in plain and encrypted form
AC_n, T_n, W_n	PA_n's anonymous credential, two secret integers of PA_n to be used in AC_n and to generate seals
V, $\underline{V}$	Two public integers to generate two different seals
V^{Wn}, $\underline{V}^{Wn}$	1st and 2nd seals calculated by PA_n
ds, DS, $\underline{DS}$	The total number of data-pairs, the objects of data-pairs, and anonymized DS
DS_{dec}, $\underline{DS}_{dec}$	Decrypted data-pairs, and the output from DS_{dec}
DP, $\{DA_{nj}, DA_{oq}\}$	Data-pair, plain/decrypted data-objects of a DP
$\{DA_{c1}, DA_{c2}\}$	Rename of $\{DA_{nj}, DA_{oq}\}$
$d_u.DA_{oq} > d_u.DA_{nj}$	A dominance relationship between q-th data of PA_o and j-th data of PA_n concerning u-th dimension
$d_u.PA_n$, $d_u.PA_o$	u-th dimensional data respectively from DA_{nj} of PA_n and DA_{oq} of PA_o
$\underline{SW}$, SW, SL	The lists of final winners, initial winners, and losers

4.1 An Overview of Secure MSQs

The MSQs find out dominance relation sets for N parties. For instance, a legitimate stranger intends to know a ranking of the nearest and best-facilitated hotels from a particular location with lower costs within a city. Let, with D (≥ 2) attributes, *e.g.*, distance, quality, liking, costing, etc., $\{PA_1, …, PA_N\}$ are N arbitrary hotels (*i.e.*, parties) where a smaller value about each attribute is assumed as better. Now, for data-objects (DA_{nj}, DA_{oq}) of the data-pair DP of any two parties (PA_n, PA_o), PA_n is said as non-dominated or winner and PA_o as dominated or loser, *i.e.*, $PA_n \prec PA_o$ if $d_u.PA_n \le d_u.PA_o$ for all d_u of DP and $d_u.PA_n < d_u.PA_o$ for at least one d_u of DP, and vice-versa where $((u, v) \in \{1, …, D\}), (u \ne v)\}$. As the smaller values are better, the object of PA_n would be the skyline. But both PA_n and PA_o are denoted as winners while $d_u.PA_n > d_u.PA_o$ and $d_v.PA_n < d_v.PA_o$ for (DA_{nj}, DA_{oq}). Thus, the MSQs are formulated based on datasets of $\{\bigcup_{n=1}^{N-1}(PA_n), \bigcup_{o=n+1}^{N}(PA_o)\}$. Here, since datasets are sensitive and they can affect parties' commercial benefits, ensuring their correct usage to inhibit dishonest entities from attempting unauthorized modification is essential. That's why the secure MSQs framework design needs to maintain the following criteria.

- *Data privacy:* Employing multiple independent entities rather than trusting a single entity to conduct encryption-decryption tasks to protect against breaching data secrecy.
- *Data integrity:* Taking measures so that the originality of the data (*i.e.*, data accuracy) withstands.
- *Data anonymity:* In order to eliminate the relation, links between parties' encrypted datasets and their decrypted forms must be removed.
- *Parties' anonymity:* In order to conceal links between the party and its retrieved data-objects, parties must disclose their identities anonymously.
- *Auditable deeds:* In order to assure the correct deeds of entities, their deeds must be exposed publicly.

4.2 System Model

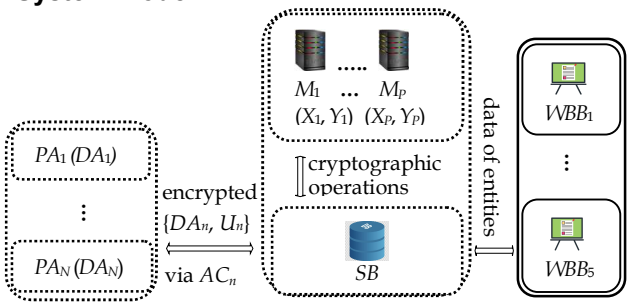

Fig. 1. The system model.

The proposed framework of secure MSQs involve N parties, a mix-net consisting of P mix-servers, a system manager SB and 5 WBBs, *i.e.*, *UT_List*, *PA_List*, *Data_List*, *AnD_List*, and *Res_List* to post various data. Every party PA_n maintains a private dataset $(DA_n \in \{DA_1, …, DA_N\})$ with the highest of D dimensions $(d_u \in \{d_1, …, d_D\})$ and the highest data record L_n. Here, the value of D and the data type (*e.g.*, numeric) must be uniform for all parties, but the number of data records L_n may vary from another party PA_o's records L_o. Again, any particular data DA_{nj} of PA_n within DA_n is represented as $(DA_{nj} \in \{DA_{n1}, …, DA_{nL}\})$. Also, every PA_n has an anonymous credential AC_n to prove its identity ID_n to SB anonymously. Each M_i keeps a decryption key X_i, discloses the corresponding public key Y_i, and Y_* is computed by the Y_i as $Y_* = Y_1Y_2…Y_P$. The SB remains connected with PA_n and M_i while needed, posts the data on various WBBs, and conducts non-cryptographic operations. Fig. 1 depicts the system model of the proposed framework.

The framework undertakes a weaker assumption about trustworthiness, *i.e.*, nothing can make it unreliable as long as a single mix-server remains honest. Other privacy constraints are as below.

- No entity including other parties except PA_n can know PA_n's dataset in plain and encrypted forms.
- While comparing the objects in a DP (in Section 4.4.6), about every dimension the smaller valued object is denoted as the dominant *i.e.*, the winner.
- No party including PA_n can know the number of dominant datasets plus their owners about PA_n's dominated datasets. Similarly, it's also applicable to the remaining parties.

4.3 The Role of Entities

This section explains the roles of the involved entities. In addition, the configuration of WBBs is described below

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

and Fig. 2 depicts them.

Party PA_n: PA_n is defined uniquely by a ID_n, and uses the credential AC_n obtained from *SB* that includes a secret random integer W_n to reveal itself anonymously and to calculate multiple seals, respectively. It keeps its organization's dataset DA_n, encrypts them by using Y_*, and joins with encrypted *UT*s separately before submitting to *SB*, and finally, the MSQs are calculated based on them.

System manager SB: *SB* generates plain *UT*s, registers, gives credentials, and assigns encrypted *UT*s to parties. In addition, it calculates the Cartesian data-pairs of parties' encrypted datasets and MSQs outcomes, posts data on various *WBB*s, checks inconsistency (if any) of decrypted datasets, etc. Besides, *SB* provides integers V and $\underline{V}$ so that every PA_n can calculate V^{Wn} and $\underline{V}^{Wn}$.

Mix-server M_i: Each M_i holds a key-pair (X_i, Y_i) to perform various cryptographic operations along with other mix-servers, namely to form encrypted *UT*s and to decrypt data-pairs calculated from parties' encrypted datasets. Also, for data anonymization, each M_i re-encrypts and shuffles encrypted data-pairs before decryption.

ID	credential
…	…
ID_n	PA_n's signed C_n

a) *PA_List*

encrypted data	seal
…	…
$\langle E_{Y^*}(k_{n1}+r_{*n1}, DA_{n1}U_{n1}), E_{Y^*}(r_{*n1}, U_{n1})\rangle, \ldots,$ $\langle E_{Y^*}(k_{nL}+r_{*nL}, DA_{nL}U_{nL}), E_{Y^*}(r_{*nL}, U_{nL})\rangle$	$\underline{V}^{Wn}$

c) *Data_List*

UT	encrypted *UT*	assigned *UT*	seal
…	…	…	…
U_n	$E_{Y^*}(r_{*n}, U_n)$	$E_{Y^*}(r_{*n1}, U_{n1}), \ldots, E_{Y^*}(r_{*nL}, U_{nL})$	V^{Wn}

b) *UT_List*

anonymized data-pair	decrypted data-pair	queries
…	…	
$\langle\{E_{Y^*}(K_{*nj}, DA_{nj}U_{nj}), E_{Y^*}(R_{*nj}, U_{nj})\}$, $\{E_{Y^*}(K_{*oq}, DA_{oq}U_{oq}), E_{Y^*}(R_{*oq}, U_{oq})\}\rangle$	$\langle\{DA_{nj}U_{nj}, U_{nj}\}$, $\{DA_{oq}U_{oq}, U_{oq}\}\rangle$	skyline objects

d) *AnD_List* e) *Res_List*

Fig.2. Configurations of *WBB*s.

PA_List: It consists of *ID* and credential parts. The *ID* part puts the ID_n of PA_n owing a dataset DA_n for MSQs. After obtaining a credential C_n, the registered PA_n puts a digital signature on C_n by a signature technique to post it on the credential part as shown in Fig. 2 a).

UT_List: As in Fig. 2 b), it holds *UT*, encrypted *UT*, assigned *UT*, and seal parts, there the last two parts are solely for every PA_n. It posts the plain *UT*s on the *UT* part, the encrypted *UT*s on the encrypted *UT* part, the encrypted *UT*s assigned to PA_n on the assigned *UT* part, and the seal V^{Wn} on the seal part to approve PA_n's obtained *UT*s.

Data_List: It comprises the encrypted data and the seal parts. The data part contains PA_n's encrypted dataset with *UT*s and as approval of the posting of this dataset the seal part contains the seal $\underline{V}^{Wn}$ of PA_n as in Fig. 2 c).

AnD_List: It contains the anonymized Cartesian data-pairs of datasets from *Data_List* as shown in Fig. 2 d).

Res_List: It consists of the decrypted data-pair part and the queries part. The data-pair part contains the decrypted data of *AnD_List* and the queries part contains the outcomes of initial and final dominance relationships along with a ranking of the objects as in Fig. 2 e).

4.4 Individual Stages

The proposed MSQs framework consists of six stages, *i.e.*, *UT* formation, registration, data submission, data-pair handling, verification, and skyline outcomes. They are described below and Fig. 3 depicts the dataflow diagram.

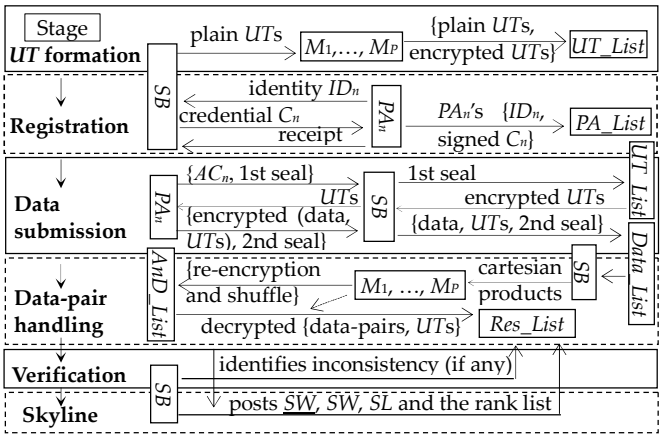

Fig.3. The data flow among entities.

4.4.1 UT formation

The objective of this stage is to generate encrypted *UT*s by involving mix-servers so that no one can link between the plain and the encrypted *UT*s. Here, the major operations are (*i*) generation of plain *UT*s, (*ii*) construction of encrypted *UT*s, and (*iii*) public disclosure of plain and encrypted *UT*s. Here, *SB* and $M_1, \ldots, M_P$ work as follows.

1. Each party PA_n declares the required number of *UT*s, *i.e.*, U_n equal to L_n (*i.e.*, $U_n \equiv L_n$) in advance. The PA_n will request *SB* to provide these *UT*s in Section 4.4.3.
2. While U denotes the total number of *UT*s as $U = \sum_{n=1}^{N} U_n$ where ($U_n \in \{U_1, \ldots, U_N\}$), $U_n \equiv L_n$, ($U_{nj} \in \{U_{n1}, \ldots, U_{nL}\}$), *i.e.*, U_{nj} (($n \in \{1, \ldots, N\}$), ($j \in \{1, \ldots, L\}$)) denotes *j*-th *UT* of PA_n. Now, *SB* generates the U numbers of plain *UT*s and sends them to $M_1, \ldots, M_P$.
3. $M_1, \ldots, M_P$ encrypt *UT*s, *i.e.*, transform each U_{nj} into $E_{Y^*}(r_{*nj}, U_{nj})$ based on step 2 of Section 3.4 that removes the link between U_{nj} and $E_{Y^*}(r_{*nj}, U_{nj})$. Likewise, $U_n = \{U_{n1}, \ldots, U_{nL}\}$ is transformed into $\underline{U}_n = \{E_{Y^*}(r_{*n1}, U_{n1}), \ldots, E_{Y^*}(r_{*nL}, U_{nL})\}$.
4. Next, *SB* discloses $U_n = \{U_{n1}, \ldots, U_{nL}\}$ as well as $\underline{U}_n = \{E_{Y^*}(r_{*n1}, U_{n1}), \ldots, E_{Y^*}(r_{*nL}, U_{nL})\}$ on *UT_List* publicly.

Security problems of this stage are treated as follows.

- *$M_1, \ldots, M_P$ may encrypt UTs incorrectly:* Registered *UT*s in plain form are disclosed from the beginning. Again, after merging with PA_n's dataset, *UT*s will be retrieved and posted as in Section 4.4.4. If *UT*s are encrypted incorrectly, their initial plain forms cannot be retrieved later on. In addition, this incorrect encryption doesn't bring any benefit to anyone.
- *SB may post encrypted UTs different from plain UTs: SB* assigns encrypted *UT*s to PA_n from *UT_List*. Later on, assigned *UT*s are retrieved and disclosed. If a mismatch exists between the two plain forms of *UT*s, it is publicly identifiable.

4.4.2 Registration

The major objectives of this stage are to (*i*) accomplish the

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

registration of participating parties, and (*ii*) let them appear anonymously while interacting further. To attain these, the major operations are to (*i*) disclosure of parties' identities, (*ii*) assignment of credentials for parties, (*iii*) confirmation of parties' involvement by disclosing their signed credentials, and (*iv*) enable parties to be anonymous. Here, *SB* and PA_n interact as follows.

1. At first, PA_n shows ID_n to *SB* through in-person contact so that a different entity cannot pretend as PA_n.
2. *SB* assigns a credential C_n to PA_n that includes PA_n's secret integer W_n as in Section 3.5. PA_n also signs on C_n to authorize it and sends back $\{ID_n, C_n\}$ to *SB*.
3. *SB* reveals $\{ID_n, C_n\}$ on *PA_List* as shown in Fig. 2 a).
4. PA_n generates a receipt and sends it to *SB* while the C_n was a legitimate one.

The security problem of this stage is treated as follows.

- *PA_n may attempt to obtain multiple credentials:* To obtain a C_n it is a must for PA_n to show its unique ID_n. Alongside, the receipt of PA_n is proof of already having the C_n. Thereby, PA_n cannot obtain multiple C_n.

4.4.3 Data submission

The main objectives of this stage are to (*i*) enable PA_n to submit the encrypted dataset so that data integrity continues while in the successive processing, and (*ii*) allow PA_n to approve its obtained *UT*s and submitted dataset. Here, the major operations are (*i*) authentication of PA_n anonymously by *SB*, (*ii*) approval of obtained *UT*s by PA_n, (*iii*) submission of PA_n's encrypted dataset along with *UT*s, and (*iv*) approval of submitted dataset by PA_n itself. Fig. 4 a) shows PA_n's dataset encryption procedure. Here, PA_n and *SB* act and interact as follows.

1. PA_n calculates AC_n (as in Section 3.5) to prove its identity to *SB* anonymously.
2. After authentication, *SB* delivers the required number of encrypted *UT*s (as in Section 4.4.1) to PA_n.
3. To approve the assigned *UT*s, PA_n submits its 1st seal V^{Wn} to *SB* to be put on *UT_List* as in Fig. 2 b).
4. Now PA_n encrypts $DA_n = \{DA_{n1}, \ldots, DA_{nL}\}$ into $\underline{DA}_n = \{E_{Y^*}(k_{n1}, DA_{n1}), \ldots, E_{Y^*}(k_{nL}, DA_{nL})\}$ based on Section 3.2.
5. PA_n calculates $DAU_n = \{E_{Y^*}(k_{n1}+r_{*n1}, DA_{n1}U_{n1}), \ldots, E_{Y^*}(k_{nL}+r_{*nL}, DA_{nL}U_{nL})\} = \{(E_{Y^*}(k_{n1}, DA_{n1}))(E_{Y^*}(r_{*n1}, U_{n1})), \ldots, (E_{Y^*}(k_{nL}, DA_{nL}))(E_{Y^*}(r_{*nL}, U_{nL}))\}$ by using *UT*s of step 2 of Section 3.4 through homomorphic encryption (*i.e.*, Algorithm 1, step 1, line 1).
6. PA_n submits $\{DAU_n, \underline{U}_n\} = \{[E_{Y^*}(k_{n1}+r_{*n1}, DA_{n1}U_{n1}), E_{Y^*}(r_{*n1}, U_{n1})], \ldots, [E_{Y^*}(k_{nL}+r_{*nL}, DA_{nL}U_{nL}), E_{Y^*}(r_{*nL}, U_{nL})]\}$ to *SB* as its encrypted dataset to be put on *Data_List* (*i.e.*, Algorithm 1, step 1, line 2).
7. After finding $\{DAU_n, \underline{U}_n\}$ on the *WBB*, PA_n sends the 2nd seal $\underline{V}^{Wn}$ to be posted on *Data_List* for approval as shown in Fig. 2 c) (*i.e.*, Algorithm 1, step 1, line 3).

Security problems of this stage are treated as follows.

- *SB may not assign any of the required UTs to PA_n*: Only PA_n holds the credential C_n and can calculate V^{Wn}. If *SB* denies PA_n to issue the required *UT*s, PA_n can prove *SB*'s dishonesty by disclosing V^{Wn} which is not yet posted on the *UT_List*.
- *SB may provide the same UTs to multiple parties: UT*s assigned to PA_n are already approved by V^{Wn}. Thus, *SB* cannot reissue the same *UT*s to other parties.
- *A coercer may obtain PA_n's seal:* It may occur only when PA_n handovers its C_n and W_n to others, or are stolen. Here, PA_n itself would be responsible for this.
- *SB may not post PA_n's dataset publicly:* Same as the first problem of this stage, PA_n can prove this dishonesty by $\underline{V}^{Wn}$ which is not yet posted on *Data_List*.
- *SB may modify the contents submitted by PA_n*: Data posted on any *WBB* are publicly disclosed. Hence, *SB* or any other entity cannot do so.

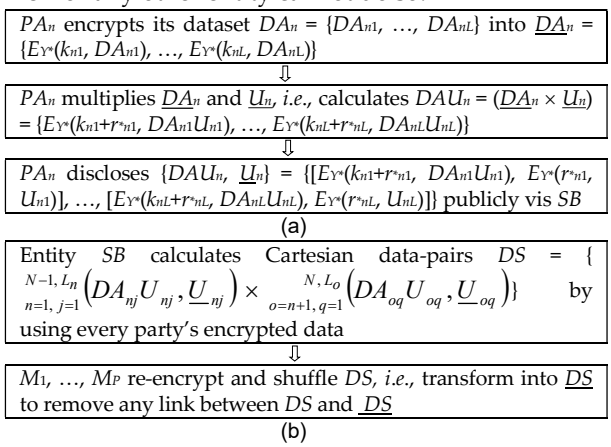

Fig. 4. The procedure of (a) dataset encryption, and (b) data-pair anonymization.

4.4.4 Data-pair handling

The major objectives of this stage are to (*i*) organize submitted datasets to be compatible with MSQs, (*ii*) anonymize data to eliminate links between the incoming and the outgoing ones, and (*iii*) retrieve the data. Here, the major operations are (*i*) formation of data-pairs by Cartesian products, (*ii*) anonymization of data-pairs by re-encryption and shuffling, and (*iii*) retrieval of data-pairs via repeated decryptions. Here, *SB* and $M_1, \ldots, M_P$ interact as follows.

1. Considering $\{DAU_n, \underline{U}_n\}$ of every PA_n, *SB* calculates the Cartesian data-pairs *DS*. For every two parties' every data, *SB* calculates the *ds* number of Cartesian pairs as $ds = \sum_{n=1}^{N-1}(L_n \,.\, \sum_{o=n+1}^{N} L_o)$. Explicitly, considering DA_{nj} $[(n \in \{1, \ldots, N\}), (j \in \{1, \ldots, L_n\})]$ as a data of PA_n, and DA_{oq} $[(o \in \{1, \ldots, N\}$ and $(o \neq n)), (q \in \{1, \ldots, L_o\})]$ as a data of PA_o; *SB* calculates the data-pair $DS_{nojq} = \{[E_{Y^*}(k_{nj}+r_{*nj}, DA_{nj}U_{nj}), E_{Y^*}(r_{*nj}, U_{nj})], [E_{Y^*}(k_{oq}+r_{*oq}, DA_{oq}U_{oq}), E_{Y^*}(r_{*oq}, U_{oq})]\}$. Thus, *SB* calculates $DS = \{ \underset{n=1,\, j=1}{\overset{N-1,\, L_n}{}} \left(DA_{nj}U_{nj}, \underline{U}_{nj}\right) \times \underset{o=n+1,\, q=1}{\overset{N,\, L_o}{}} \left(DA_{oq}U_{oq}, \underline{U}_{oq}\right)\}$ (*i.e.*, Algorithm 1, step 2, line 1).
2. $M_1, \ldots, M_P$ re-encrypt data-objects of data-pairs within *DS* and shuffle them (based on Section 3.3) obtained from *SB* for anonymization. Namely, DS_{nojq} is transformed into $\underline{DS}_{nojq} = \{[E_{Y^*}(K_{*nj}, DA_{nj}U_{nj}), E_{Y^*}(R_{*nj}, U_{nj})], [E_{Y^*}(K_{*oq}, DA_{oq}U_{oq}), E_{Y^*}(R_{*oq}, U_{oq})]\} = \{[[g^{(knj+r^*nj)}.(g^{\sum_{i=1}^{P} s_i}), (DA_{nj}U_{nj}).Y_*^{(knj+r^*nj)}.(Y_*^{\sum_{i=1}^{P} s_i})], [g^{r^*nj}.(g^{\sum_{i=1}^{P} s_i}), (U_{nj}).Y_*^{r^*nj}.(Y_*^{\sum_{i=1}^{P} s_i})]], [[g^{(koq+r^*oq)}.(g^{\sum_{i=1}^{P} t_i}), (DA_{oq}U_{oq}).$

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

$Y_*^{(koq+r^*oq).(\sum_{i=1}^{P} t_i)}$)], [$g^{r^*oq}.(\ g^{\sum_{i=1}^{P} t_i}$), ($U_{oq}$).$Y_*^{r^*oq}.(Y_*^{\sum_{i=1}^{P} t_i})$]]} while using secret integers s_i and t_i of each M_i. Here, {$K_{*nj} = k_{nj} + r_{*nj} + \sum_{i=1}^{P} s_i$, $R_{*nj} = r_{*nj} + \sum_{i=1}^{P} s_i$ } and {$K_{*oq} = k_{oq} + r_{*oq} + \sum_{i=1}^{P} t_i$, $R_{*oq} = r_{*oq} + \sum_{i=1}^{P} t_i$ }. Thus, as in Fig. 4 b), all encrypted data-pairs of *DS* are transformed into the anonymized data-pairs $\underline{DS}$ where due to re-encryption and shuffling no link exists between *DS* and $\underline{DS}$. Now, *SB* posts $\underline{DS}$ on *AnD_List* as in Fig. 2. d) (*i.e.*, Algorithm 1, step 2, line 2).

3. Then, as in Fig. 5 a), M_1, …, M_P decrypt all data-pairs of $\underline{DS}$ as DS_{dec} = {$\underline{DS}_1$, …, $\underline{DS}_P$} (based on Section 3.3) by using X_1, …, X_P consecutively. For example, the decrypted value of $\underline{DS}_{nojq}$ is {[($DA_{nj}U_{nj}$), U_{nj}], [($DA_{oq}U_{oq}$), U_{oq}]}. *SB* puts DS_{dec} on the *Res_List* as shown in Fig. 2 e) (*i.e.*, Algorithm 1, step 2, line 3).
4. *SB* excludes *UT*s attached with data-objects within DS_{dec} through division operation. Namely, from {[($DA_{nj}U_{nj}$), U_{nj}], [($DA_{oq}U_{oq}$), U_{oq}]} *SB* calculates {[($DA_{nj}U_{nj}$) / U_{nj}], [($DA_{oq}U_{oq}$) / U_{oq}]} = {DA_{nj}, DA_{oq}} and continues this operation for remaining data-pairs. Thus, the total output is denoted as $\underline{DS}_{dec}$ = {($DS1_{dec}$ / U^T1), …, ($DSds_{dec}$ / U^Tds)} (*i.e.*, Algorithm 1, step 2, line 4) where (($DSz_{dec} \in \{DS1_{dec}, \ldots, DSds_{dec}\}$), ($DSz_{dec}$ and U^Tz consist of two data-objects and their corresponding attached *UT*s, respectively), and ($z \in \{1, \ldots, ds\}$)). In the remainder, objects DA_{nj} and DA_{oq} are renamed as DA_{c1} and DA_{c2}, respectively.

Security problems of this stage are treated as follows.

- *M_1, …, M_P may re-encrypt and decrypt data dishonestly:* Thereby, incorrect decryption results would be obtained. Then, entities liable for disruption will be identified in the Verification stage (Section 4.4.5) by the disruption detection mechanism similar to [22].
- *M_1, …, M_P may add or delete data:* By these, the number of data records posted on *UT_List* (*i.e.*, assigned *UT*s), *Data_List*, *AnD_List*, and *Res_List* will become unequal which is publicly identifiable.

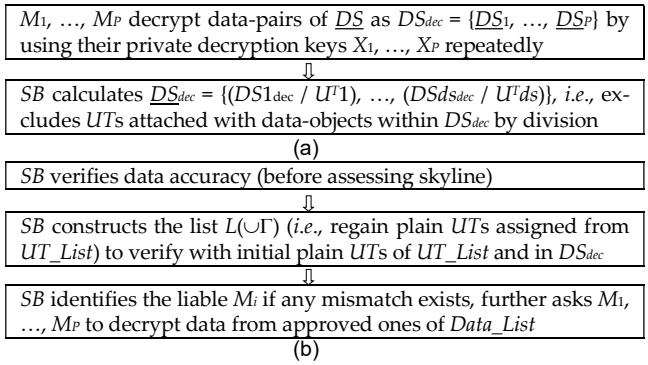

Fig. 5. The procedure of (a) data-pair decryption, and (b) data verification.

4.4.5 Verification

The objective of this stage is to detect any improper data (if exists). To do so, *SB* compares the data on various *WBB*s and detects the liable entities. Fig. 5 b) shows the procedure and it works as follows.

1. *SB* asks M_1, …, M_P to construct a list $L(\cup\Gamma)$ of the plain *UT*s retrieved from the assigned *UT*s of *UT_List* by their decryption operations. Now, *SB* compares the list $L(\cup\Gamma)$ with the initial plain form of *UT*s (*i.e.*, 1st column of Fig. 2 b)) of *UT_List*.
2. Further, *SB* compares the list $L(\cup\Gamma)$ with the *UT*s of DS_{dec} (*i.e.*, 1st column of Fig. 2 e)) in *Res_List* (*i.e.*, Algorithm 1, step 3, line 1).
3. Here, a mismatch may exist because consistent data can be constructed and decrypted by uniting the encrypted data (*i.e.*, 1st column of Fig. 2 c)) of *Data_List*. Similarly, a consistent *UT* can be formed and decrypted by uniting the encrypted *UT*s (*i.e.*, 3rd column of Fig. 2 b)) of *UT_List*. But, these *UT*s are not unique, or not included in $L(\cup\Gamma)$ list, or differ from *UT*s in the plain registered form {U_1, ..., U_M}.
4. Afterward, *SB* asks M_1, …, M_P to show their correct decryption and re-encryption operations. Here, through the disruption detection mechanisms discussed in [22], honest mix-servers can prove the correctness of their operations without disclosing their secrets, and dishonest ones are identified (*i.e.*, Algorithm 1, step 3, line 2).
5. If dishonest mix-server(s) exists, *SB* asks it to perform a further re-encryption and decryption to recover the incorrectly handled data by tracing back the original encrypted data-pairs *DS*, *i.e.*, by using the approved data of *Data_List*. Since unfair entities were already identified once, later on, they must behave honestly.
6. After recovering all plain data, *SB* proceeds to the next stage, *i.e.*, to calculate the skyline outcomes. While there was no mismatch, *SB* assumes that the mix-servers are honest from the beginning.

Here, a security threat is, by conspiring with the coercer and getting the non-anonymous data, the 1st mix-server M_1 can recognize the link between encrypted data and its corresponding decrypted form. But as discussed in step 5, while M_1 re-encrypts and shuffles the cipher further, it is bound to utilize the newly generated secrets, rather than previously used ones. Thereby M_1 cannot continue to preserve the link. Thus, it brings no benefit to M_1.

In an actual sense, the Verification stage aims not to identify dishonesties but rather to assure anyone regarding the honest conduction of entities' operations. Thus, an adversary rarely can succeed to perform improperly.

Algorithm 1 Dealing Encrypted Datasets
Input: $\underline{DA}_n$ and $\underline{U}_n$ of every PA_n.
Output: Retrieved data-pairs $\underline{DS}_{dec}$.
Step 1: Data Submission
1: Calculation of DAU_n = {$\underline{DA}_n$,$\underline{U}_n$} by PA_n;
2: Submission of {DAU_n, $\underline{U}_n$} by PA_n via *SB* to post on *Data_List*;
3: Approval of submitted dataset by the seal $\underline{V}^{Wn}$ of PA_n;
Step 2: Data-pair Handling
1: Calculation of *DS* using (DAU_n, $\underline{U}_n$) of every PA_n by *SB*;
2: Transformation of *DS* to anonymized ones $\underline{DS}$ by M_1, …, M_P;
3: Decryption of $\underline{DS}$ into DS_{dec} by M_1, …, M_P;
4: Obtaining of $\underline{DS}_{dec}$ through the exclusion of *UT*s by *SB*;
Step 3: Verification
1: Comparison between retrieved *UT*s and initial plain *UT*s by *SB*;
2: Disruption detection for M_1, …, M_P and data recovery by *SB*;

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

Algorithm 1 presents the pseudocode of dealing with the encrypted datasets by involved entities.

4.4.6 Skyline outcomes

The objective of this stage is to assess the skyline queries from retrieved data-pairs and rank the objects. Here, the outcomes are calculated through three steps, *i.e.*, (*i*) initial comparisons, (*ii*) final comparisons, and (*iii*) generating a ranking of the objects. The operations proceed as follows.

1. For objects $\{DA_{c1}, DA_{c2}\}$ of a data-pair within $\underline{DS}_{dec}$, *SB* calculates the initial dominance relationships as:
   - DA_{c2} is denoted as the winner, *i.e.*, $DA_{c1} > DA_{c2}$ if $d_u.DA_{c1} \geq d_u.DA_{c2}$ for all d_u and $d_u.DA_{c1} > d_u.DA_{c2}$ for at least one d_u. In contrast, DA_{c1} is the winner.
   - Both DA_{c1} and DA_{c2} are denoted as winners or non-dominated to each other objects, *i.e.*, $DA_{c1} \cong DA_{c2}$ if $d_u.DA_{c1} > d_u.DA_{c2}$ and $d_v.DA_{c1} < d_v.DA_{c2}$ where $u \neq v$.

   In detail, *SB* initializes two variables as $x = 0$, and $y = 0$ to count the winners for each dimension of $\{DA_{c1}, DA_{c2}\}$ (*i.e.*, Algorithm 2, step 1, line 1). Now, *SB* assigns the values of x and y (*i.e.*, Algorithm 2, step 1, line 2) as below.
   - $x \leftarrow x + 1$ when $\{d_u.DA_{c2} < d_u.DA_{c1}\}$; or
   - $y \leftarrow y + 1$ when $\{d_u.DA_{c1} < d_u.DA_{c2}\}$; or
   - Skip d_u when $\{d_u.DA_{c1} = = d_u.DA_{c2}\}$ and forward to d_{u+1};

   Now, the initial winners *SW* (*i.e.*, list of 'the dominant' and 'the non-dominated to each other' objects) and the losers *SL* (*i.e.*, list of 'the dominated' objects) are evaluated and stored (*i.e.*, Algorithm 2, step 1, line 3) as below.
   - $SW \leftarrow DA_{c2}$ and $SL \leftarrow DA_{c1}$ when $\{(x>y)$ and $(y==0)\}$; or
   - $SW \leftarrow DA_{c1}$ and $SL \leftarrow DA_{c2}$ when $\{(y>x)$ and $(x==0)\}$; or
   - $SW \leftarrow \{DA_{c1}, DA_{c2}\}$ when $\{(x==y)$ or $((x\neq0)$ and $(y\neq0))\}$;

   Similarly, considering x, y for each remaining data-pairs separately within $\underline{DS}_{dec}$ *SB* evaluates and stores the lists of winners and losers as $\{SW, SL\} = \{\underline{DS}1_{dec}, \ldots, \underline{DS}ds_{dec}\}$ where $((\underline{DS}z_{dec} \in \{\underline{DS}1_{dec}, \ldots, \underline{DS}ds_{dec}\})$, $(\underline{DS}z_{dec}$ consists of two data-objects also), and $(z \in \{1, \ldots, ds\}))$. Then, *SB* counts the repeated objects as singular objects within both *SW* and *SL* (*i.e.*, Algorithm 2, step 1, line 4). The above queries are regarded as the initial comparisons.
2. If an object exists in both *SW* and *SL*, *SB* removes that common object from *SW* through the set difference (SD) operation to generate the list of final winners $\underline{SW}$ (*i.e.*, Algorithm 2, step 2, line 1). Here, the objects existing in $\underline{SW}$ must not persist in *SL*, and the operations are specified as below.
   - *SB* assigns $\underline{SW} \leftarrow (SW - SL)$.
   - If $(SW - SL)$ is null, *SB* assigns $\underline{SW} \leftarrow SW$.

   These queries are regarded as the final comparisons.
3. Besides, considering the initial as well as the final comparisons, a ranking of objects is formed (*i.e.*, Algorithm 2, step 3, line 1) as follows:
   - Rank–1 ← the list of objects available only in $\underline{SW}$, *i.e.*, only the final winners;
   - Rank–2 ← the list of objects common in *SW* and *SL*, *i.e.*, the initial winners but not the final winners;
   - Rank–3 ← the list of objects available only in *SL* but never existed in *SW*, *i.e.*, only the losers.

   If there exists no common objects between *SW* and *SL*, then Rank–2 will disappear and Rank–3 will turn into Rank–2, *i.e.*, Rank–2 ← Rank–3.

Dominance relationships		Ranking of objects
Initial	Final	
SW and *SL* contain both common and distinct objects	$\underline{SW}$ and *SL* contain only distinct objects	Rank–1 objects; Rank–2 objects; Rank–3 objects;

Fig. 6. The skyline outcomes.

Finally, *SB* posts the above outcomes (*i.e.*, steps 1 ~ 3) on the 2nd column of the *Res_List*, *i.e.*, Fig. 2 e). In detail, it is shown in Fig. 6. Herein, it is notable that rather than *SB*, any entity or a third party can conduct the above Verification and Skyline outcomes procedure.

Algorithm 2 presents the pseudocode of finding out the skyline outcomes.

Algorithm 2 Skyline outcomes
Input: Decrypted data-pairs $\underline{DS}_{dec}$.
Output: The ranking of objects.
Step 1: Initial Comparisons
1: Initialization of: $x = 0$, $y = 0$ for each $\{DA_{c1}, DA_{c2}\}$;
2: Increment of x or y for queries of each $\{DA_{c1}, DA_{c2}\}$ within $\underline{DS}_{dec}$;
3: Update *SW* and *SL* based on x and y to know the initial results;
4: Count repeated objects as a singular one in both *SW* and *SL*;
Step 2: Final Comparisons
1: Removal of objects from *SW* common in *SW* and *SL*;
Step 3: Ranking of objects
1: Making a rank of objects from the outcomes of Steps 1 and 2;

5 Evaluation of the Framework

5.1 Experimental Setup

A prototype of the proposed framework was developed under a 2.50 GHz 64–bit processor, and 8 GB RAM using synthetic correlated datasets with data dimensions (2 ~ 5), and the length of each data and *UT* were 60-digit and 40-digit integers, respectively. Besides, it was depicted using independent and anti-correlated datasets as in [6]. Moreover, for cryptographic operations it deployed 3 mix-servers, and used Python 3.6.4 with a 1024-bit modulus for coding purposes. Since it was developed in a single computer, during data cast the communication delays were skipped. While computation times were measured, *UT* formation and Verification stages were not considered because the former ended before the parties' involvement, and the latter was required only if entities are dishonest.

TABLE 5
Computation Time of Various Stages

Individual Stage	Processing time
Registration	19ms/party
Data submission	212.3ms/party
Data-pair handling	711.61 sec (all parties all data)
Skyline outcomes	46.52 sec (all parties all data)

5.2 Experimental Results

Table 5 shows the computation time of various stages where the prototype considered 10 parties holding a maximum of 20 data with $2D \sim 5D$ and thus, handled a maximum of 17600 data-pairs, *i.e.*, 35,200 objects. While processing, the registration stage took 19ms for the party PA_n where *SB* required 9.4ms to assign the credential and PA_n

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

needed 9.6ms to produce a receipt. The data submission stage consists of four major operations and for 2*D* data, it took 212.3ms for each PA_n, *i.e.*, to authenticate PA_n *SB* required 19.3ms, to generate seal V^{Wn} PA_n required 8.5ms, to encrypt 20 data and unite them with *UT*s PA_n needed 176ms and to generate seal $\underline{V}^{Wn}$ PA_n needed 8.5ms. The data-pair handling stage that consists of three major tasks, to handle all data it consumed 711.61 sec, *i.e.*, to generate 17600 data-pairs *SB* required 8.4ms, to re-encrypt and shuffle them M_1,…, M_3 took 563.8 sec and to decrypt (here each M_i worked in parallel as the cryptosystem is commutative) by M_1,…, M_3 and to divide them *SB* required 147.8 sec. Regarding the skyline outcomes, it took 46.52 sec, *i.e.*, to generate the rank list, the initial and the final comparisons took 45.27 sec and 1.25 sec, respectively.

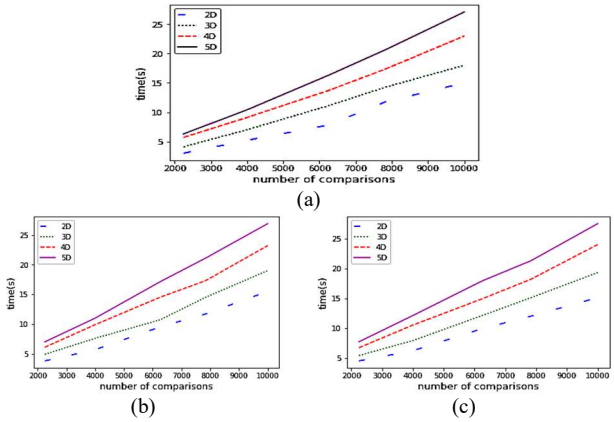

Fig. 7. The computation time to calculate the dominance relationships by the proposed framework for different dimensions of (a) correlated, (b) independent, and (c) anti-correlated data-objects.

Fig. 7 shows the computation time required to calculate the dominance relationships by varying the dimensions where Figs. 7 a), 7 b), and 7 c) present the outcomes for correlated, independent, and anti-correlated data-objects, respectively. It shows that for every dimension, while the number of comparisons increases, the time requirements also increase. Again, for every higher dimension, it takes a higher time than its lower dimension, proportionally.

5.3 Runtime Comparisons and Discussions

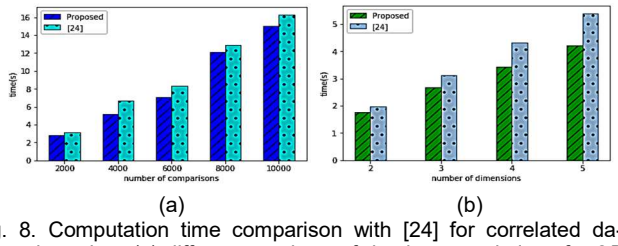

Fig. 8. Computation time comparison with [24] for correlated datasets based on (a) different numbers of dominance relations for 2*D* data, and (b) varying the dimensions for 1000 comparisons.

This section compares the proposed work with [24], [27], and [26] through runtime. Fig. 8 uses 2*D* correlated datasets to compare with [24], *i.e.*, by varying the number of comparisons it plots Fig. 8 a), and by using 1000 comparisons plus changing the data dimension it draws Fig. 8 b). Here for calculating skyline queries, both works considered all party's data and the figure shows that the work in [24] requires a higher time than the proposed one. The reasons are that [24] needs to initialize random positive integer vectors, binary vectors, etc., to calculate the initial dominance relationships. Then, all parties randomize their initial results further to calculate the final dominance relationships. But the proposed work calculates the initial dominance relationships directly by comparing the objects. Then, by using the set difference operation, simply it calculates the final dominance relations.

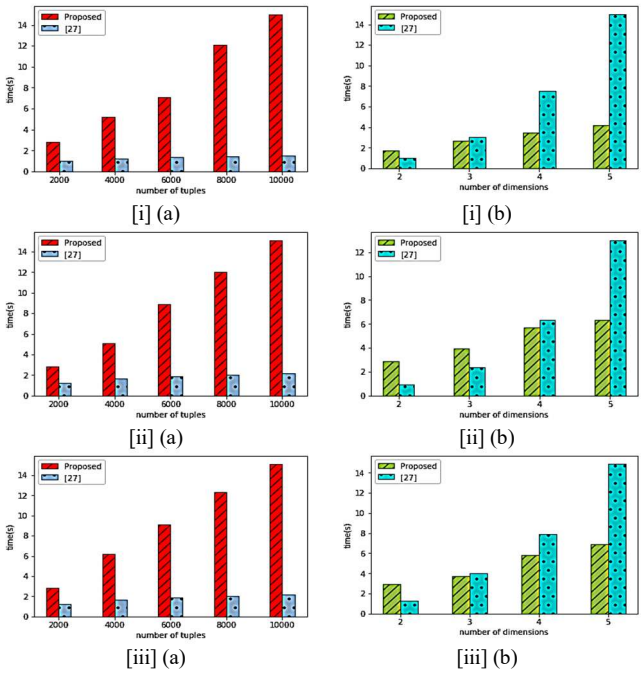

Fig. 9. Computation time comparison with [27] for [i] correlated, [ii] independent, and [iii] anti-correlated datasets based on (a) different numbers of dominance relationships for 2*D* data, and (b) varying the dimensions for 1000 comparisons.

Fig. 9 compares the proposed work with [27] by varying the numbers of tuples (*i.e.*, comparisons) based on 2*D* correlated, independent, and anti-correlated datasets, and outputs are shown in Figs. 9 [i] a), 9 [ii] a), and 9 [iii] a), respectively. Also, by varying dimensions and for uniformity considering the number of tuples as 1000, outputs are plotted accordingly in Figs. 9 [i] b), 9 [ii] b) and 9 [iii] b). Where, for 2*D* data (*i.e.*, Fig. 9 a)), [27] takes less time than the proposed one because its datasets become smaller after each round of comparisons by removing the dominated tuples, In contrast, when dimensions increase for the same tuples, [27] takes a relatively higher time than the proposed one because the comparison procedure of [27] takes more time about the increment of dimension.

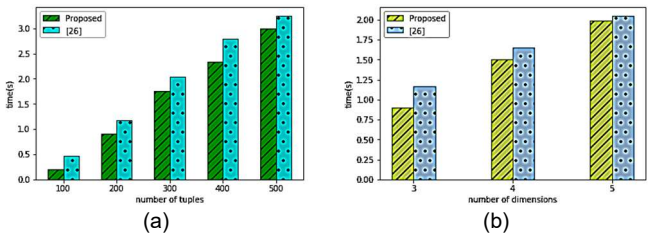

Fig. 10. Computation time comparison with [26] based on (a) varying the size of the dataset and selected dimensions 3 out of 5, and (b) selecting 3*D* ~ 5*D* out of 5*D* data for 200 tuples.

Further, by considering an analogous feature, *i.e.*, adapting flexible dimensions, Fig. 10 compares the pro-

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

posed work with [26]. Hence, same as [26], taking datasets 100 ~ 500 for 3*D* data, the output is shown in Fig. 10 a), and by selecting 3*D* ~ 5*D* out of 5*D* datasets considering 200 tuples, the result is depicted in Fig. 10 b). For both cases, [26] requires a higher time than the proposed one because, with its search-space, it first checks whether the query intersects or not, the point lies inside of it or not, and then finds the non-dominated points. Whereas, after picking the desired data dimensions, the proposed work mainly uses comparisons, set difference, etc., operations.

5.4 Security and Allied Requirements

Based on cryptographic, security, and allied requirements this section analyzes the proposed framework and compares it with [24], [5], [27], and [26] as follows. Table 6 summarizes the comparisons, and Table 7 presents their number of operations. Table 6 shows that the proposed MSQs exclusively incorporate data integrity, data and party's anonymity, cipher data approval, ranking of objects, etc. Where [24] assesses only the dominance relationships, and other ones produce dominance relationships based on query requests. Besides, [26] and [27] rely on lightweight cryptography and others are based on different asymmetric cryptosystems with distinct features. Also, to assess the MSQs, and store the data, the proposed framework engages the 3rd party and various *WBB*s, but [24] manages these issues by the parties and for the other ones, these are done by the respective cloud server (s).

Data privacy: Distinct private keys of M_1, …, M_P decrypt the encrypted data successively. As in Table 6 and Table 7, though underlying techniques of data privacy are distinct here, the proposed work amply retains it.

Data integrity: PA_n's encrypted dataset is approved by PA_n itself. Also, the deeds of entities are disclosed on *WBB*s. As in Section 4.4.5, due to *UT*s, any improper data is publicly identifiable which confirms data integrity. But, as in Table 7, the compared works do not focus on it.

Data anonymity: To re-encrypt and shuffle each encrypted data-pair, it requires $\{2 \times (2 \times P)\}$ operations. Also, as in Table 7, for shuffling [24] needs 1 random permutation, [5] needs 1 random noise addition and *D* operations for each picked data, where [26] and [27] do not use it. But, the compared ones do not re-encrypt together.

Party's anonymity: To deploy the AC_n it requires 1 exponentiation (exp.) and 1 modulus (mod) operation for each PA_n. Besides, to approve its deeds (*i.e.*, to create 2 seals), PA_n needs 2 exp. and 2 mod operations. But, as in Table 7, the compared works do not consider it.

Accuracy: Registered PA_n takes part in the system via AC_n. Now, PA_n's $\underline{DA}_n$ approved by $\underline{V}^{Wn}$ confirms that they are cast as PA_n intended. Also, the data on *WBB*s ensure that they are recorded as cast. Further, the uniqueness of *UT*s, and the comparisons ensure that MSQs are assessed from recorded data. Thus, the work retains accuracy.

Auditable deeds: Unique *UT*s, data on *WBB*s, comparisons in Section 4.4.5, etc., assure correct deeds of entities.

Robustness: In case of improper data on *Res_List*, M_1, …, M_P are asked to re-calculate results using the original approved data of *Data_List*. Since improper data is identified once, now M_1, …, M_P must calculate them correctly which assures the robustness of the framework.

TABLE 6
A COMPARISON BASED ON CRYPTOGRAPHIC TECHNIQUES AND OTHER MAJOR ASPECTS

Aspects	Framework				
	Proposed	[24]	[5]	[27]	[26]
How to authenticate registered parties	by anonymous credential	not mentioned	not mentioned	not mentioned	not mentioned
Trust about entities/models	not trusted (except a single mix-server)	semi-honest	curious-but-honest	semi-honest and non-colluding	honest-but-curious
How many entities hold decryption key(s)	multiple (≥ 2) separate mix-servers	a single entity (each party individually)	2 fixed separate cloud-servers	equal to number of servers (*i.e.*, 2)	none (but single key for query)
How encryption key is generated	combining ≥ 2 public keys	individual key of every party	combining public keys of 2 servers	2 secret shares	a random invertible matrix
Skyline datasets encryption and decryption	encryption by a single key, decryption by ≥ 2 distinct keys	encryption and decryption both by single distinct key	encryption by single key, decryption by 2 distinct keys	associated with the number of servers (*i.e.*, 2)	encryption by step wise separate secret keys
Cipher data approval	by its owner (via seal)	not considered	not considered	not considered	sends securely
Fashion of basic cryptosystem	ElGamal (by ≥ 2 entities)	Paillier (by a single entity)	Paillier (by 2 entities)	lightweight additive secret sharing	lightweight predicate encryption
Re-encryption of data	by (≥ 2) mix-servers	not considered	not considered	not considered	not considered
How retains data integrity	uniting *UT*s with data	not considered	not specified clearly	not specified clearly	not specified clearly
Anonymization of encrypted data	re-encryption and shuffling	random shuffling	random noise and shuffling	not considered	not considered
Who calculates skyline	3rd party (*i.e.*, *SB*)	involved parties	cloud-servers	cloud-servers	the cloud-server
Storage of data	public *WBB*s	to every party	cloud-servers	cloud-servers	the cloud-server
Usages of datasets for skyline queries	all parties' all data (dimension may vary)	all parties' all data	based on request of query	based on request of query	based on request of query
Ranking of objects	by rank-wise listing	not considered	not considered	not considered	not considered
Verification of skyline queries	comparison of $\underline{DS}_{dec}$ and $\{\underline{SW}, SW, SL\}$	not considered	additive homomorphism	data labeling, filtering, etc.	inner product over data

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

TABLE 7
A COMPARISON BASED ON THE NUMBER OF OPERATIONS FOR CONSIDERED SECURITY AND SKYLINE ASPECTS

Aspects		Framework				
		Proposed	[24]	[5]	[27]	[26]
Data privacy	Encryption key	1 (from P public keys)	1*	1 (from 2 public keys)	2 shares	stepwise {1, 1, 2}
	Decryption key	P distinct keys	1*	2 distinct keys	–	–
Data integrity (per data)		1 homomorphic encryption	NC	NC	NC	NC
Data anonymity (per DP)	Re-encryption	$2 \times P$	NC	NC	NC	NC
	Shuffling	$2 \times P$	1	$1 + D^{\$}$	NC	NC
Party's anonymity	For credential	1 exp. and 1 mod	NC	NC	NC	NC
	Approval of deeds	2 exp. and 2 mod (for 2 seals)	NC	NC	NC	NC
Operations for ranking of objects		[{(D×$\underline{DS}_{dec}$)×((č+ã)/č)}+{$\underline{DS}_{dec}$×(č +(2×ã))}]+[{(č+SD+ã)/(č+ã)}]	NC	NC	NC	NC

* = separately for each party; NC = not considered; $ = while data is picked; – = works on cipher data; SD = set difference;

Verifiability of skyline: To calculate MSQs, verified data $\underline{DS}_{dec}$ is used. Now, *SB* or any entity calculates MSQs outcomes, and as in Table 6, they are publicly verifiable.

Ranking of objects: As in Table 7, to form the rank list, the operations are divided into the calculation of {*SW*, *SL*} and $\underline{SW}$. For {*SW*, *SL*}, it requires {(1 comparison (č) plus 1 assignment (ã)), or a single č} for each dimension of every data-pair of $\underline{DS}_{dec}$, and {1 č plus 2 ã} for every data-pair of $\underline{DS}_{dec}$. Again, for $\underline{SW}$, it requires {(1 č, 1 SD and 1 ã), or (1 č plus 1 ã)}. But, the compared ones do not consider it.

TABLE 8
PROBABLE DISRUPTIONS BY DISHONEST MIX-SERVERS

Desired outcomes	Obtained outcomes	Remarks
{U_{a^*}, …, U_{z^*}}	{U_{a^*}, …, U_{z^*}}	Consistent *UT*s
	{U_{a^*}, …, U_{y^*}, $U_{\$}$}	Disrupted *UT*s
{$DA_{nj}U_{a^*}$, U_{a^*}}, {$DA_{oq}U_{b^*}$, U_{b^*}}	{$DA_{nj}U_{a^*}$, U_{a^*}}, {$DA_{oq}U_{b^*}$, U_{b^*}}	Consistent data
	{$DA_{nj}U_{\&}$, $U_{\&}$}, {$DA_{oq}U_{b^*}$, U_{b^*}} / vice-versa / both *UT*s	Disrupted *UT*s
	{$DA_{nj^*}U_{a^*}$, U_{a^*}}, {$DA_{oq}U_{b^*}$, U_{b^*}} / vice-versa / both data	Disrupted data

Moreover, by introducing dishonest mix-servers an experiment was conducted, and the symbolic outputs are shown in Table 8. Here, the obtained outcomes of the 2nd, 4th, and 5th rows are disrupted due to the unregistered *UT* (*i.e.*, $U_{\$}$), modified or not-assigned *UT* (*i.e.*, $U_{\&}$), and inconsistent data-object (*i.e.*, $DA_{nj^*}U_{a^*}$) which are related to steps 1 ~ 3 of Section 4.4.5. But, they are recoverable.

Based on the retained aspects, *e.g.*, level of privacy, data integrity, data and parties' anonymity, auditable deeds, robustness, ranking of objects, the empirical results of Sections 5.2 and 5.3, comparisons of Tables 6 and 7, etc., prove the efficacy and practicality of the proposed work.

5.5 Security Analysis

Considering the ideal functionality of the proposed MSQs as ξ, the formal security analysis is as follows.
Input. Party PA_n provides its dataset DA_n to ξ.
Computation. From datasets, ξ finds MSQs and rank list.
Output. ξ discloses allied data, rank list, etc., on *WBB*s.
Let, Ω represents a protocol for secure MSQs which realizes ξ. The security of Ω is formally defined as follows:
Definition 1. *Let,* $£_1$, $£_2$, …, $£_\#$ *are security requirements for a framework* ¥ *and an adversary* $\mathcal{A}$ *has a view of a corrupted system during the execution of* Ω. *A* ¥ *is secure if a PPT simulator S can simulate a view for* $\mathcal{A}$ *which is indistinguishable from the real view, i.e.,* $view_{Real}$ ($\mathcal{A}$, Ω) ≈ $view_{Ideal}$ (*S*) *[28].*
Theorem 1. *According to Definition 1, in the not-trusted framework, proposed MSQs* ¥ *can securely realize the ideal* ξ.

Proof. Here, $S_{¥}$(*C*, *X*) represents the simulator of ¥ that generates *C*'s view in the execution of item *X* on the corresponding input and output. By the proposed ¥, the major succeeded security concerns are: *i*) anonymity of PA_n, *ii*) privacy, integrity, approval, and anonymity of $\underline{DA}_n$, *iii*) honest deeds of M_1, …, M_P, *iv*) verifiable deeds including MSQs results, etc., along with *v*) public data on *WBB*s (as stated in Sections 4.4.2 ~ 4.4.6, 5.4). Also, security problems argued in individual stages of Section 4.4 imply that ¥ resists them, results are verifiable (if required, recoverable) through various *WBB*s, so, *C* grasps nothing during the execution of ¥. Hence, $S_{¥}$(*C*, *X*) exists. Q.E.D.

6 CONCLUSIONS

The proposed framework of MSQs allocates separate decryption keys for mutually separate entities, appoints them to re-encrypt and shuffle the encrypted data and retrieve the plain form of data through repeated decryptions. Thus, the framework retains ample data privacy and data anonymity. Also, singly the attachment of *UT*s with parties' encrypted datasets attains data integrity. Besides, the use of anonymous credentials and publicly exposed deeds of entities attains parties' anonymity and makes the framework robust and auditable. Additionally, from the dominance relationships of skyline queries, finally, it ranks the objects. The evaluation, comparisons, and security analyses imply that the framework is adequately practical and efficient. As for upcoming works, adopting dynamic queries as well as incomplete datasets, implementing through blockchain, considering the variants of the skyline queries, etc., are the plans.

REFERENCES

[1] K. Hose and A. Vlachou, "A survey of skyline processing in highly distributed environments," Int. J. Very Large Data Bases, Vol. 21, No. 3, pp. 359–384, 2012.
[2] M. Bai, S. Jiang, X. Zhang, X. Wang, "An efficient skyline query algorithm in the distributed environment,".Journal of Computational Science, Elsevier, Vol. 58, p. 101524, Feb. 2022.
[3] D. Das, K. M. R. Alam, and Y. Morimoto, "A framework for multi-party skyline query maintaining privacy and data integri-

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3323961

ty," in 2021 24th Int. Conf. on Computer and Information Technology (ICCIT). IEEE, pp. 1–6, Dec. 2021.

[4] X. Zhang, R. Lu, J. Shao, H. Zhu, and A. A. Ghorbani,. "Continuous Probabilistic Skyline Query for Secure Worker Selection in Mobile Crowdsensing," IEEE Internet of Things Journal, Vol. 8, No. 14, pp. 11758-11772, 2021.

[5] X. Ding, Z. Wang, P. Zhou, K.-K. R. Choo, and H. Jin, "Efficient and privacy-preserving multi-party skyline queries over encrypted data," IEEE Trans. on Information Forensics and Security, Vol. 16, pp. 4589–4604, 2021.

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

[7] S. Tamura and S. Taniguchi, "Enhancement of anonymous tag based credentials," in Information Security Comput. Fraud, Vol. 2, No. 1, pp. 10–20, 2014.

[8] N. Islam, K. M. R. Alam, and S. S. Rahman, "Commutative re-encryption techniques: Significance and analysis," Information Security Journal: A Global Perspective, Taylor & Francis, Vol. 24, No. 4-6, pp. 185-193, 2015.

[9] N. Islam, K. M. R. Alam, and A. Rahman, "The effectiveness of mixnets–an empirical study," Computer Fraud & Security, Elsevier, Vol. 2013, No. 12, pp. 9–14, 2013.

[10] J. Chen, J. Huang, B. Jiang, J. Pei, and J. Yin, "Recommendations for two-way selections using skyline view queries," Knowledge and information systems, Vol. 34, No. 2, pp. 397-424, 2013.

[11] M. Qaosar, K. M. R. Alam, C. Li, and Y. Morimoto, "Privacy-Preserving Top-K Dominating Queries in Distributed Multi-Party Databases," 2019 IEEE Int. Conf. on Big Data, pp. 5794-5803, 2019.

[12] H. A. Fattah, K. M. A. Hasan, and T. Tsuji, "Weighted top-k dominating queries on highly incomplete data," Information Systems, Elsevier, Vol. 107, p. 102008, 2022.

[13] Y. Gulzar, A. A. Alwan, H. Ibrahim, S. Turaev, S. Wani, A. B. Soomo, and Y. Hamid, "IDSA: An efficient algorithm for skyline queries computation on dynamic and incomplete data with changing states," IEEE Access, Vol. 9, pp. 57291-57310, 2021.

[14] X. Liu, K. K. R. Choo, R. H. Deng, Y. Yang, and Y. Zhang, "PUSC: Privacy-preserving user-centric skyline computation over multiple encrypted domains," in Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE), pp. 958–963, 2018.

[15] K. M. R. Alam, S. Tamura, S. Taniguchi, and T. Yanase, "An anonymous voting scheme based on confirmation numbers," IEEJ Trans. on EIS, Vol. 130, No. 11, pp. 2065–2073, 2010.

[16] X. Liu, R. Lu, J. Ma, L. Chen, and H. Bao, "Efficient and privacy preserving skyline computation framework across domains," Future Generation Computer Systems, Elsevier, Vol. 62, pp. 161–174, 2016.

[17] J. Hua et al., "CINEMA: Efficient and privacy-preserving online medical primary diagnosis with skyline query," IEEE Internet Things J., Vol. 6, No. 2, pp. 1450–1461, Apr. 2019.

[18] D. Dolev, C. Dwork, and M. Naor, "Non-malleable cryptography," in Proc. of the twenty-third annual ACM symposium on Theory of computing, pp. 542-552, 1991.

[19] S. Bothe, A. Cuzzocrea, P. Karras, and A. Vlachou, "Skyline query processing over encrypted data: An attribute-order-preserving-free approach," in Proc. 1st Int. Workshop Privacy Secuirty Big Data (PSBD), pp. 37–43, 2014.

[20] S. Bothe, P. Karras, and A. Vlachou, "Eskyline: Processing skyline queries over encrypted data," in Proc. VLDB, Vol. 6, No. 12, pp. 1338–1341, 2013.

[21] W. Chen, M. Liu, R. Zhang, Y. Zhang, and S. Liu, "Secure outsourced skyline query processing via untrusted cloud service providers," in Proc. IEEE INFOCOM 35th Annu. Int. Conf. Comput. Commun., pp. 1–9, Apr. 2016.

[22] K. M. R. Alam, S. Tamura, S. M. S. Rahman, and Y. Morimoto, "An electronic voting scheme based on revised-SVRM and confirmation numbers," IEEE Trans. on Dependable and Secure Computing, Vol. 18, No. 1, pp. 400-410, 2021.

[23] M. S. Rahman, I. Khalil, A. Alabdulatif, and X. Yi, "Privacy preserving service selection using fully homomorphic encryption scheme on untrusted cloud service platform," Knowledge-Based Systems, Vol. 180, pp. 104–115, 2019.

[24] M. Qaosar, K. M R. Alam, A. Zaman, C. Li, S. Ahmed, M. A. Siddique, and Y. Morimoto, "A framework for privacy-preserving multi-party skyline query based on homomorphic encryption," IEEE Access, Vol. 7, pp. 167481–167496, 2019.

[25] K. Sampigethaya and R. Poovendran, "A Framework and Taxonomy for Comparison of Electronic Voting Schemes," Computers and Security, Vol. 25, No. 2, pp. 137-153, 2006.

[26] S. Zhang, S. Ray, R. Lu, Y. Zheng, Y. Guan, J. Shao, "Towards efficient and privacy-preserving user-defined skyline query over single cloud," IEEE Trans. on Dependable and Secure Computing, Feb. 2022.

[27] Y. Zheng, W. Wang, S. Wang, X. Jia, H. Huang, and C. Wang, "SecSkyline: Fast Privacy-Preserving Skyline Queries over Encrypted Cloud Databases," IEEE Trans. on Knowledge and Data Engineering, Nov. 2022.

[28] Y. Lindell, "How to simulate it - a tutorial on the simulation proof technique," in Tutorials on the Foundations of Cryptography, pp. 277–346, 2017.

Dola Das received both the BSc and MSc degrees in Computer Science and Engineering (CSE) from Khulna University of Engineering & Technology (KUET), Bangladesh in 2019 and 2022, respectively. She is currently working as an Assistant Professor in the same department. Her research interests include applied cryptography, information security, and embedded system design.

Kazi Md. Rokibul Alam received the BSc and MSc degrees both in Computer Science and Engineering (CSE) from Khulna University, Bangladesh, and Bangladesh University of Engineering and Technology (BUET) in 1999 and 2004, respectively, and the Dr (Eng.) degree in System Design Engineering from University of Fukui, Japan, in 2010. He is a Professor in the Dept. of CSE of Khulna University of Engineering & Technology (KUET), Bangladesh. He also worked as a Researcher at Hiroshima University, Japan. His research interests include applied cryptography and information security.

Yasuhiko Morimoto is a Professor at Hiroshima University, Japan. He received his B.E., M.E. and Ph.D. degrees from Hiroshima University in 1989, 1991 and 2002, respectively. From 1991 to 2002, he had been with IBM Tokyo Research Laboratory where he worked for data mining project and multimedia database project. Since 2002, he has been with Hiroshima University. His current research interest includes data mining, machine learning, geographic information system and privacy preserving information retrieval.

Authorized licensed use limited to: Khulna Univ of Engineering & Technology - KUET. Downloaded on May 06,2024 at 03:12:01 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.