This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3153759, IEEE Transactions on Dependable and Secure Computing

Towards Efficient and Privacy-Preserving Interval Skyline Queries over Time Series Data

Songnian Zhang, Suprio Ray, *Member, IEEE,* Rongxing Lu, *Fellow, IEEE,* Yandong Zheng, Yunguo Guan, and Jun Shao, *Member, IEEE*

Abstract—Outsourcing encrypted time series data and query services to a cloud has been widely adopted by data owners for economic considerations. However, it inevitably lowers data utility and query efficiency. Existing secure skyline query schemes either leak critical information or are inefficient. In this paper, we propose an efficient and privacy-preserving interval skyline query scheme by employing symmetric homomorphic encryption (SHE). Specifically, we first devise a secure sort protocol to sort the encrypted dataset and a secure high-dimensional dominance check protocol to securely determine dominance relations of time series data, in which a dominance check tree is presented. With these secure protocols, we propose our secure skyline computation protocol that can ensure both security and efficiency. Furthermore, to deal with the characteristics of time series data, we design a look-up table to index time series for quick query response. The security analysis shows that our proposed scheme can protect outsourced data, query results, and single-dimensional privacy and hide access patterns. In addition, we evaluate our proposed scheme and compare the core component of our scheme with the state-of-the-art solution, and the results indicate that our protocol outperforms the compared solution by two orders of magnitude in the computational cost and at least 23$\times$ in the communication cost.

Index Terms—Interval skyline query, Time series data, Privacy preservation, Access pattern, Symmetric homomorphic encryption.

1 Introduction

Time series data occurs ubiquitously across human endeavors and has a wide range of applications, such as biological experimental observations, social activity mining, and electricity consumption monitoring [1]–[3]. Hence, analyzing time series data has attracted extensive researches [1]–[8]. Among them, interval skyline over time series data, i.e., segments of time series that show dominating advantages over others, is of particular interest [6]–[8] since it can capture the time series that have high time series values in a query interval. For clarity, a real-world example is illustrated as follows.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
s_1	81	80	84	82	79	83	80	81
s_2	76	79	80	77	76	80	75	77
s_3	74	76	75	80	85	81	79	82
s_4	75	75	77	74	75	75	76	74

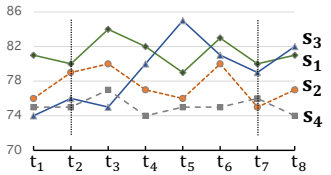


Fig. 1. An example of the interval skyline query over time series data.

Example 1. A hospital provides online medical monitoring for patients, and a patient's heart rate can be captured by a time series. Fig. 1 shows four time series (four patients) with the value of beats per minute (bpm) in a period from t_1 to t_8. Assume a doctor would like to analyze the heart rate of different patients by asking "which patients have high bpm in the time interval from t_2 to t_7?". s_1 and s_3 are interesting to the doctor. That is because s_1 has the highest average bpm, while s_3 has the highest bpm at t_5. Regarding s_2 and s_4, they are ignored since s_1 is higher than both of them at each timestamp of the query interval t_2-t_7.

In the above example, finding interesting time series $\{s_1, s_3\}$ can be achieved by performing an interval skyline query. Technically, a time series s is returned if, in the query interval, there does not exist another time series s' that dominates s. Here, "dominate" means a time series is not worse than the other one at each timestamp and is better than that time series at least one timestamp [6]. See the formal definition of dominance and interval skyline query in Section 3.1.

One of the characteristics of the time series data is that it involves continuous updates over time, which indicates the service providers should keep online at all times to receive the reported data from entities. However, with the data volume growing, the service providers may pay a great price to stay online and maintain the data. Accordingly, they tend to outsource their data and the corresponding services, e.g., interval skyline query in this paper, to a third-party cloud for reaping economic benefits. Nevertheless, it inevitably raises privacy issues since the real-world time series data may contain sensitive information, for instance, the medical data in the above example. To address the privacy challenge, a promising solution is to encrypt outsourced data and perform queries over encrypted time series data [9], [10].

However, existing privacy-preserving skyline schemes investigate over multi-dimensional data rather than time series data [11]–[14]. Although their core component, i.e., securely determining dominance relation (whether one data record dominates another one), can be used in the privacy-preserving interval skyline query scheme if we treat each

- *S. Zhang, S. Ray, R. Lu, Y. Zheng and Y. Guan are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: szhang17@unb.ca, sray@unb.ca, rlu1@unb.ca, yzheng8@unb.ca, yguan4@unb.ca).*
- *J. Shao is with School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, 310018, China (e-mail: chn.junshao@gmail.com).*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3153759, IEEE Transactions on Dependable and Secure Computing

timestamp as a dimension, they are either leaking critical information or inefficient. For example, the solution in [14] leaks the order relations of each dimension, while the solution in [12] is expensive though it is secure (see the analysis of other solutions in Section 8). Therefore, in this paper, our goal is to propose a privacy-preserving interval skyline query scheme while ensuring efficiency.

Nevertheless, the following challenges should be dealt with: i) Privacy preservation. In addition to the privacy of time series data, our proposed scheme should also preserve single-dimensional privacy [15] and hide access patterns [16], [17]. That is because revealing these information may incur inference attacks on the outsourced data [18], [19]. Here, single-dimensional privacy indicates the order or equality relation of values in each dimension (timestamp). However, in skyline computation, it is essential for an operator to determine the dominance relation by obtaining the order relation of each dimension. Therefore, it is challenging to compute skyline without leaking single-dimensional privacy. While for the access pattern, it requires the operator to select skyline points without knowing which ones are selected. As a result, ensuring these privacy is challenging in performing interval skyline queries. ii) Efficiency. To determine dominance relation without leaking privacy, the state-of-the-art solution [12] checks the order relation of dimensions one by one, which is expensive in the communication overhead. As a result, it raises the question "is there a more efficient solution to determine dominance relation without compromising privacy?". iii) Time series data. The characteristics of time series data elicit new challenges for computing interval skyline. First, time series data are usually high-dimensional, which deteriorates the performance of the existing solutions. Second, time series data continues updates over time. Consequently, it is not easy to dynamically index these time series data such that interval skyline queries can be quickly responded as little storage cost as possible.

Aiming at the above challenges, we propose a novel interval skyline query scheme over encrypted time series data, in which a lightweight symmetric homomorphic encryption (SHE) scheme is adopted [20]. Our proposed scheme can preserve time series data privacy, single-dimensional privacy, and access patterns while ensuring efficiency. Specifically, the main contributions of this paper are three-fold.

- First, we carefully design a Secure Sort (SS) protocol and a Secure High-dimensional Dominance Check (SHDC) protocol to securely achieve the core operations in the skyline computation. For the SS protocol, it can securely sort time series data according to their sum values by using the privacy-preserving XOR and XNOR gates, in which the former can be used in quick permutation, while the latter can be used to hide access patterns. Regarding the SHDC protocol, it can securely determine dominance relations without checking order relations for all dimensions one by one. In particular, we propose a dominance check tree, denoted as DC-tree, to deal with the high-dimensional time series data while improving performance.
- Second, based on the SS and SHDC protocols, we propose a secure skyline computation protocol to compute skyline over high-dimensional time series data, in which we employ a new algorithm by modifying the sort-filter-skyline (SFS) algorithm [21]. In addition, to quickly respond to the interval skyline query, we design a two-dimensional look-up table to index the time series data, which can balance the storage costs and computational costs to cope with continuous updates of time series data.
- Third, we evaluate the performance of our proposed scheme and compare the core component with the state-of-the-art solution [12]. The results show that our protocol outperforms the compared solution by two orders of magnitude in the computational cost and at least 23× in the communication cost.

The reminder of this paper is organized as follows. In Section 2, we introduce our system model and security model. Then, we review our preliminaries in Section 3. After that, we introduce the building blocks in Section 4 and present our proposed scheme in Section 5, followed by security analysis and performance evaluation in Section 6 and Section 7, respectively. Finally, we discuss some related works in Section 8 and draw our conclusion in Section 9.

2 System Model and Security Model

In this section, we formalize our system model and security model.

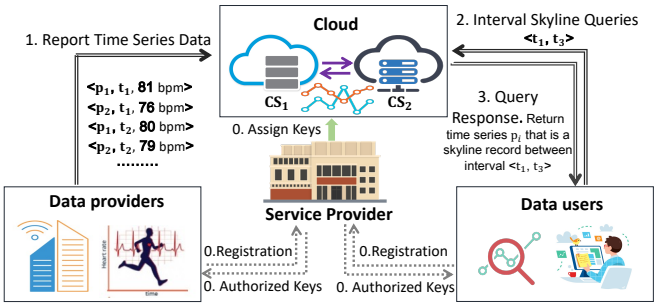


Fig. 2. System model under consideration.

2.1 System Model

In our system model, we consider a typical cloud-based interval skyline query model, which consists of four types of entities: a service provider $\mathcal{SP}$, a set of data providers $\mathcal{P} = \{p_1, p_2, \cdots\}$, a cloud $\mathcal{C}$ with two servers $\{\mathcal{CS}_1, \mathcal{CS}_2\}$, and multiple data users $\mathcal{U}$=$\{u_1, u_2, \cdots\}$, as shown in Fig. 2.

Service Provider $\mathcal{SP}$: In our system model, $\mathcal{SP}$ is an initiator for the whole system. It offers registration services to other entities and provides interval skyline query services to registered data users. However, since $\mathcal{SP}$ has limited storage and computing resources, it outsources the time series data to a cloud and employs the cloud to offer the interval skyline query services to users. Before offering these services, $\mathcal{SP}$ generates secret keys and securely distributes them to different entities in the system.

Data Providers $\mathcal{P}$=$\{p_1, p_2, \cdots\}$: We consider each registered smart device as a data provider. A data provider p_i can collect data and report it to the cloud at a certain time interval. The format of the reported data is: $\langle \text{id}, \text{timestamp}, \text{value} \rangle$. Taking a patient with the chronic heart disease as an example, the implanted body sensor of the patient reports the heart rate, i.e., beats per minute (bpm), to the cloud as the format $\langle p_1, t_i, \text{bpm} \rangle$, where p_1 is id of the sensor, and t_i $(i = 1, 2, \cdots)$ denotes the timestamp.

For the reported values, we assume all of them are non-negative integers. It is reasonable since we can easily convert non-integer data into non-negative integers [22].

Cloud $\mathcal{C}$: In our system, $\mathcal{SP}$ employs two cloud servers $\mathcal{C} = \{\mathcal{CS}_1, \mathcal{CS}_2\}$ that are considered as powerful in storage and computing resources. They will manage the reported time series data and cooperatively offer reliable interval skyline query services to data users.

Data users $\mathcal{U}$=$\{u_1, u_2, ...\}$: In the system, only the authorized data users can enjoy the interval skyline query services from the cloud $\mathcal{C}$. That is, in order to obtain desired results from $\mathcal{C}$, data users must register to $\mathcal{SP}$ before launching the interval skyline query requests.

2.2 Security Model

In our security model, since $\mathcal{SP}$ is the service organizer and has no motivation to deviate from the proposed protocols, he/she is considered as *fully trusted*. For data providers $\mathcal{P}$ and data users $\mathcal{U}$, we consider the authorized ones are *honest*, i.e., they will honestly report time series data and launch the interval skyline queries, respectively. However, in our model, the cloud servers $\mathcal{CS}_1$ and $\mathcal{CS}_2$ are considered as *honest-but-curious* [23], [24]. They will faithfully follow the designed protocols but may be curious to learn the private information. For example, the cloud servers may be interested in the reported heart rate to determine whether the owner of the sensor has heart disease. As a result, to ensure privacy, the data providers report the encrypted time series data: $\langle \text{id}, \text{timestamp}, \text{encrypted value} \rangle$. Nonetheless, the cloud still attempts to obtain the private information, including the plaintexts of the time series values and query results, single-dimensional privacy, and access patterns, in the process of interval skyline queries. Meanwhile, we assume there is no collusion between $\mathcal{CS}_1$ and $\mathcal{CS}_2$, as well as no collusion between the cloud servers and other entities. It is reasonable since they should maintain their reputations and interests [24]. Note that the two-server model with the no-collusion assumption has already been considered in many research works in the security community [24]–[27]. In addition, regarding other active attacks, e.g., Dos attack, since our work focuses on the secure computation techniques, those attacks are beyond the scope of this work and will be discussed in future work.

3 Preliminaries

In this section, we first formally define the interval skyline query. Then, we recall the symmetric homomorphic encryption (SHE) scheme.

3.1 Interval Skyline Query

The interval skyline query is used to compute skyline on time series data, which was first introduced in [6]. Before delving into the details, we first present the time series data.

A time series $\mathtt{s}$ is a sequence of pairs $\langle t_i, \text{value} \rangle$ ordered by t_i, where t_i $(i = 1, 2, \cdots)$ is a timestamp. We denote the value of $\mathtt{s}$ at timestamp t_i as $\mathtt{s}[t_i]$. Following the assumption in [6], all time series are synchronized, i.e., each time series $\mathtt{s}$ holds a value $\mathtt{s}[t_i]$ on a timestamp $t_i > 0$.

Definition 1 (Time Interval). *A time interval $[t_i : t_j]$ indicates a range in time that contains the set of timestamps existing between t_i and t_j $(t_i < t_j)$. We say that $\mathtt{s}[t_i, t_j] = (\mathtt{s}[t_i], \mathtt{s}[t_{i+1}], \cdots, \mathtt{s}[t_j])$ is a subsequence of time series $\mathtt{s}$ in the time interval $[t_i : t_j]$.*

With the above definition, we formally define the interval dominance and interval skyline query as follows.

Definition 2 (Interval dominance). *Given two time series $\mathtt{s}_1$ and $\mathtt{s}_2$, $\mathtt{s}_1$ is said to dominate $\mathtt{s}_2$ in a time interval $[t_i : t_j]$, denoted as $\mathtt{s}_1 \succ_{[t_i:t_j]} \mathtt{s}_2$, if $\forall t_k \in [t_i, t_j], \mathtt{s}_1[t_k] \geq \mathtt{s}_2[t_k]$ and $\exists t_l \in [t_i, t_j], \mathtt{s}_1[t_l] > \mathtt{s}_2[t_l]$.*

Definition 3 (Interval Skyline Query). *Given a set $\mathcal{S}$ with n time series and a time interval $[t_i : t_j]$, the interval skyline query returns a set $\mathcal{S}_{\text{sky}} \subseteq \mathcal{S}$, in which the time series are not dominated by any other time series in $\mathcal{S}$. That is, $\mathcal{S}_{\text{sky}} = \{\mathtt{s}_k \in \mathcal{S} \mid \nexists \mathtt{s}_l \in \mathcal{S}$ such that $\mathtt{s}_l \succ_{[t_i:t_j]} \mathtt{s}_k\}$. Note that each time series in $\mathcal{S}_{\text{sky}}$ only contains values between t_i and t_j.*

3.2 Symmetric Homomorphic Encryption

SHE is an efficient symmetric homomorphic encryption scheme that can support homomorphic addition and multiplication. It was first proposed in [28] and then proved to be IND-CPA secure in [20]. Concretely, SHE includes three algorithms, namely i) key generation $\mathsf{KeyGen}()$; ii) encryption $\mathsf{Enc}()$; and iii) decryption $\mathsf{Dec}()$, as follows:

- $\mathsf{KeyGen}(k_0, k_1, k_2)$: Given three security parameters $\{k_0, k_1, k_2\}$ satisfying $k_1 \ll k_2 < k_0$, the algorithm first chooses two large prime numbers p, q with $|p| = |q| = k_0$ and sets $\mathcal{N} = pq$. Then, it generates the secret key $sk = (p, \mathcal{L})$, where $\mathcal{L}$ is a random number with $|\mathcal{L}| = k_2$, and the public parameter $pp = (k_0, k_1, k_2, \mathcal{N})$. Besides, the algorithm sets the basic message space $\mathcal{M} = [-2^{k_1-1}, 2^{k_1-1})$.
- $\mathsf{Enc}(sk, m)$: On input of a secret key sk and a message $m \in \mathcal{M}$, the encryption algorithm outputs the ciphertext $\mathsf{E}(m) = (r\mathcal{L} + m)(1 + r'p) \mod \mathcal{N}$, where $r \in \{0,1\}^{k_2}$ and $r' \in \{0,1\}^{k_0}$ are random numbers.
- $\mathsf{Dec}(sk, \mathsf{E}(m))$: Taking the secret key sk and a ciphertext $\mathsf{E}(m)$ as inputs, the algorithm recovers a message $m' = (\mathsf{E}(m) \mod p) \mod \mathcal{L} = (r\mathcal{L} + m) \mod \mathcal{L}$. If $m' < \frac{\mathcal{L}}{2}$, it indicates $m \geq 0$ and $m = m'$. Otherwise, $m < 0$ and $m = m' - \mathcal{L}$.

SHE satisfies the homomorphic addition and multiplication properties as follows: i) Homomorphic addition-I: $\mathsf{E}(m_1) + \mathsf{E}(m_2) \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 + m_2)$; ii) Homomorphic multiplication-I: $\mathsf{E}(m_1) \cdot \mathsf{E}(m_2) \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 \cdot m_2)$; iii) Homomorphic addition-II: $\mathsf{E}(m_1) + m_2 \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 + m_2)$; iv) Homomorphic multiplication-II: $\mathsf{E}(m_1) \cdot m_2 \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 \cdot m_2)$ when $m_2 > 0$.

Encryption with the public key setting. In order to realize the SHE encryption under public key setting, we take $sk = (p, \mathcal{L})$ as the private key and use it to generate two ciphertexts $\mathsf{E}(0)_1$, $\mathsf{E}(0)_2$ of 0 with different random numbers. Then, the public key is set as $pk = \{\mathsf{E}(0)_1, \mathsf{E}(0)_2, pp\}$. In such a way, one can use the above homomorphic properties to encrypt a message m by the following

$$\mathsf{E}(m) = m + r_1 \cdot \mathsf{E}(0)_1 + r_2 \cdot \mathsf{E}(0)_2 \mod \mathcal{N} \qquad (1)$$

where r_1 and $r_2 \in \{0,1\}^{k_2}$ are two random numbers. For example, when $m = -1$, $\mathsf{E}(-1)$ can be easily computed by

Eq. (1). This approach has been formally proven to be IND-CPA secure in [29]. Note that, since SHE has the limited number of Homomorphic multiplication-I operations, we adopt a bootstrapping protocol in [20] to support infinite multiplication operations.

4 Building Blocks

In this section, we introduce the secure subprotocols and privacy-preserving logic gates, which serve as the building blocks for constructing more complex protocols.

4.1 Secure Subprotocols

To achieve our privacy-preserving interval skyline query scheme, we need to compute some basic functions on encrypted data. Here, we show Secure Bigger Than (SBT) and Secure Equal (SEQ) subprotocols to determine whether two given encrypted inputs (messages) have the bigger than and equal to relations, respectively. Both of them are deployed in a two-server model, where $\mathcal{CS}_1$ holds encrypted inputs and pk, while $\mathcal{CS}_2$ has the secret key sk.

4.1.1 Secure Bigger Than (SBT) Subprotocol

Given two encrypted messages $\mathsf{E}(m_1)$ and $\mathsf{E}(m_2)$, the SBT subprotocol is to determine whether $m_1 > m_2$ without leaking any m_1, m_2 related information to $\mathcal{CS}_1$ or $\mathcal{CS}_2$. If $m_1 > m_2$, the subprotocol outputs $\mathsf{E}(1)$, otherwise $\mathsf{E}(0)$.

• Step-1: $\mathcal{CS}_1$ flips a coin $s \in \{-1, 1\}$ and chooses two random numbers $r_1, r_2 \in \{0,1\}^{k_1}$ satisfying $r_1 > r_2 > 0$. Then, $\mathcal{CS}_1$ computes:

$$\begin{aligned}\mathsf{E}(\theta) =& \mathsf{E}(s \cdot r_1) \cdot (\mathsf{E}(m_1) + \mathsf{E}(m_2) \cdot \mathsf{E}(-1)) + \mathsf{E}(s \cdot r_2) \cdot \mathsf{E}(-1) \\ =& \mathsf{E}(s \cdot r_1 \cdot (m_1 - m_2) - s \cdot r_2),\end{aligned}$$

where $\mathsf{E}(-1)$ can be easily computed from pk via Eq. (1). After that, $\mathcal{CS}_1$ sends $\mathsf{E}(\theta)$ to $\mathcal{CS}_2$.

• Step-2: On receiving $\mathsf{E}(\theta)$, $\mathcal{CS}_2$ uses sk to recover θ and checks whether $\theta > 0$. If yes, $\mathcal{CS}_2$ makes $\mu = 1$ and encrypts it into $\mathsf{E}(\mu)$. Otherwise, $\mathcal{CS}_2$ generates $\mathsf{E}(\mu) = \mathsf{E}(0)$. Next, $\mathcal{CS}_2$ returns $\mathsf{E}(\mu)$ to $\mathcal{CS}_1$.

• Step-3: If $s = 1$, $\mathcal{CS}_1$ lets $\mathsf{E}(\delta) = \mathsf{E}(\mu)$. If $s = -1$, $\mathcal{CS}_1$ computes $\mathsf{E}(\delta) = \mathsf{E}(1) + \mathsf{E}(\mu) \cdot \mathsf{E}(-1) = \mathsf{E}(1 - \mu)$. Finally, $\mathsf{E}(\delta)$ is the output of the SBT subprotocol.

Correctness. When $s = 1$, we have $\mathsf{E}(\theta) = \mathsf{E}(r_1 \cdot (m_1 - m_2) - r_2)$. If $m_1 > m_2$, $\theta > 0$. As a result, $\mathsf{E}(\delta) = \mathsf{E}(\mu) = \mathsf{E}(1)$. If $m_1 \leq m_2$, $\theta < 0$. In this case, $\mathsf{E}(\delta) = \mathsf{E}(\mu) = \mathsf{E}(0)$. Therefore, when $s = 1$, *iff* $m_1 > m_2$, the SBT subprotocol outputs $\mathsf{E}(\delta) = \mathsf{E}(1)$. Similarly, when $s = -1$, we can prove that *iff* $m_1 > m_2$, the SBT subprotocol outputs $\mathsf{E}(\delta) = \mathsf{E}(1)$. Thus, the SBT subprotocol is correct.

4.1.2 Secure Equal (SEQ) Subprotocol

Given two encrypted messages $\mathsf{E}(m_1)$ and $\mathsf{E}(m_2)$, the SEQ subprotocol [30] is to determine whether $m_1 = m_2$ without leaking any m_1, m_2 related information to $\mathcal{CS}_1$ or $\mathcal{CS}_2$. If $m_1 = m_2$, the subprotocol outputs $\mathsf{E}(1)$, otherwise $\mathsf{E}(0)$.

• Step-1: $\mathcal{CS}_1$ flips a coin $s \in \{-1, 1\}$ and chooses two random numbers $r_1, r_2 \in \{0,1\}^{k_1}$ satisfying $r_1 > r_2 > 0$. Then, $\mathcal{CS}_1$ computes:

$$\begin{aligned}\mathsf{E}(\theta) =& \mathsf{E}(s \cdot r_1) \cdot \mathsf{E}((m_1 - m_2)^2) + \mathsf{E}(s \cdot r_2) \cdot \mathsf{E}(-1) \\ =& \mathsf{E}(s \cdot r_1 \cdot (m_1 - m_2)^2 - s \cdot r_2).\end{aligned}$$

After that, $\mathcal{CS}_1$ sends $\mathsf{E}(\theta)$ to $\mathcal{CS}_2$.

• Step-2: On receiving $\mathsf{E}(\theta)$, $\mathcal{CS}_2$ uses sk to recover θ and checks whether $\theta < 0$. If yes, $\mathcal{CS}_2$ makes $\mu = 1$ and encrypts it into $\mathsf{E}(\mu)$. Otherwise, $\mathcal{CS}_2$ generates $\mathsf{E}(\mu) = \mathsf{E}(0)$. Next, $\mathcal{CS}_2$ returns $\mathsf{E}(\mu)$ to $\mathcal{CS}_1$.

• Step-3: If $s = 1$, $\mathcal{CS}_1$ lets $\mathsf{E}(\delta) = \mathsf{E}(\mu)$. If $s = -1$, $\mathcal{CS}_1$ computes $\mathsf{E}(\delta) = \mathsf{E}(1) + \mathsf{E}(\mu) \cdot \mathsf{E}(-1) = \mathsf{E}(1 - \mu)$. Finally, $\mathsf{E}(\delta)$ is the output of the SEQ subprotocol.

Correctness. Similar to the SBT subprotocol, the correctness of the SEQ subprotocol is also held.

4.2 Privacy-Preserving Logic Gates

In addition to the secure subprotocols, we also need some digital logic gates as building blocks to construct our proposed scheme. Here, we introduce two simple privacy-preserving logic gates: XOR and XNOR [30].

4.2.1 Privacy-Preserving XOR Gate

The XOR gate works by receiving two inputs, each designated with either 1 or 0. It outputs 0 if the two inputs are the same. Otherwise, the gate produces 1. If one input is in plaintext, denoted as a, and the other input is in ciphertext, denoted as $\mathsf{E}(b)$, one can obtain the privacy-preserving XOR gate by using the following equation.

$$\mathsf{E}(out) = a \oplus \mathsf{E}(b) = \begin{cases} \mathsf{E}(b) & if\ a = 0 \\ \mathsf{E}(1-b) & if\ a = 1, \end{cases} \tag{2}$$

where $\mathsf{E}(1 - b) = \mathsf{E}(1) + \mathsf{E}(b) \cdot \mathsf{E}(-1)$. The privacy-preserving XOR gate can securely and efficiently compute the encrypted output $\mathsf{E}(out)$. Although the input a is in plaintext, the output $\mathsf{E}(out)$ is kept secret, since the input b is encrypted.

4.2.2 Privacy-Preserving XNOR Gate

The XNOR gate is the opposite of the XOR gate, i.e., it outputs 1 if the two inputs are the same. With the same input setting as the XOR gate, the privacy-preserving XNOR gate [30] can obtain the encrypted output as follows.

$$\mathsf{E}(out) = a \odot \mathsf{E}(b) = \begin{cases} \mathsf{E}(1-b) & if\ a = 0 \\ \mathsf{E}(b) & if\ a = 1, \end{cases} \tag{3}$$

Similarly, $\mathsf{E}(out)$ can be quickly computed from $\mathsf{E}(b)$ and is kept secret due to the encrypted input $\mathsf{E}(b)$.

5 Our Proposed Scheme

In this section, we will introduce our privacy-preserving interval skyline query scheme.

The main idea is to securely realize the SFS algorithm [21] to compute skylines. In the SFS algorithm, the input dataset is sorted according to a monotone preference function, e.g., the sum of a data record. Afterward, the algorithm obtains candidate data records by the descending order of their function values and then examines dominance relations between the obtained data record and the existing skyline points. If the data record is not dominated by all existing skyline points, it will be added to a skyline set. We can see that the key operations in the SFS algorithm are sorting and checking dominance relations. Based on this observation, we devise a secure sort (SS) protocol and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3153759, IEEE Transactions on Dependable and Secure Computing

introduce a secure dominance check (SDC) protocol to make the SFS algorithm available while ensuring both efficiency and security. Furthermore, since the time series data is usually high-dimensional, we design a secure high-dimensional dominance check (SHDC) protocol to specially determine the dominance relation for high-dimensional data.

In the following, we first present SS, SDC, and SHDC protocols. Then, with these secure protocols, we design two skyline computation protocols. Finally, we present our privacy-preserving interval skyline query scheme.

5.1 Secure Protocols

5.1.1 Secure Sort (SS) Protocol

Assume $\mathcal{CS}_1$ has a set of d-dimensional data $\{\mathsf{E}(\vec{\mathrm{x}}_i) = (\mathsf{E}(\mathrm{x}_i^1), \mathsf{E}(\mathrm{x}_i^2), \cdots, \mathsf{E}(\mathrm{x}_i^d)) \mid \mathrm{x}_i^j \in \mathcal{M}, i \in [1,n], j \in [1,d]\}$ and their sum values $\mathsf{E}(\sigma_i) = \mathsf{E}(\sum_{j=1}^{d} \mathrm{x}_i^j)$, and $\mathcal{CS}_2$ holds sk. We also suppose that $\mathcal{CS}_1$ has randomly assigned a unique binary sequence $b_i = b_i^\rho|_{\rho=1}^{\lceil \log_2 n \rceil}$, where $b_i^\rho \in \{0,1\}$, to each data record $\mathsf{E}(\vec{\mathrm{x}}_i)$, as shown in Fig. 3. The goal of the SS protocol is to sort the encrypted dataset in descending order according to the sum values $\{\sigma_i \mid i \in [1,n]\}$ without leaking underlying plaintexts and access patterns to the cloud. That is, neither $\mathcal{CS}_1$ nor $\mathcal{CS}_2$ knows the plaintexts $\{(\mathrm{x}_i^1, \mathrm{x}_i^2, \cdots, \mathrm{x}_i^d, \sum_{j=1}^{d} \mathrm{x}_i^j) \mid i \in [1,n]\}$ or the link between the sorted dataset and the original dataset. We describe the details of our SS protocol as follows:

- Step-1: First, $\mathcal{CS}_1$ generates $\lceil \log_2 n \rceil$ random bits $\{r^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], r^\rho \in \{0,1\}\}$. After that, $\mathcal{CS}_1$ computes a new bit sequence for each data record, denoted as $\hat{b}_i = \hat{b}_i^\rho|_{\rho=1}^{\lceil \log_2 n \rceil}$, by performing the XOR gate $\hat{b}_i^\rho = b_i^\rho \oplus r^\rho$ (not privacy-preserving XOR gate). Next, $\mathcal{CS}_1$ chooses $n+1$ random numbers $\{r_0, r_1, \cdots, r_n\}$ satisfying $r_0 > r_i, i \in [1,n]$. Finally, $\mathcal{CS}_1$ constructs a pair $\langle \mathsf{E}(\hat{\sigma}_i), \hat{b}_i \rangle$ for each data record and sends n pairs to $\mathcal{CS}_2$, where $\mathsf{E}(\hat{\sigma}_i) = \mathsf{E}(r_0 \cdot \sigma_i + r_i)$.
- Step-2: On receiving these pairs $\{\langle \mathsf{E}(\hat{\sigma}_i), \hat{b}_i \rangle \mid i \in [1,n]\}$, $\mathcal{CS}_2$ first uses sk to recover $\hat{\sigma}_i$ and sorts the bit sequences $\{\hat{b}_i \mid i \in [1,n]\}$ in a descending order according to the corresponding $\hat{\sigma}_i$. Then, $\mathcal{CS}_2$ encrypts each bit with SHE, i.e.,$\mathsf{E}(\hat{b}_i^\rho)$. Next, $\mathcal{CS}_2$ returns the sorted and encrypted bit sequences $\{\mathsf{E}(\hat{b}_i^\rho) \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ to $\mathcal{CS}_1$.
- Step-3: With the random bits $\{r^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], r^\rho \in \{0,1\}\}$ and the received encrypted bit sequences, $\mathcal{CS}_1$ adopts the privacy-preserving XOR gate, i.e., Eq. (2), to compute a set of new bit sequences $\{\mathsf{E}(\delta_i^\rho) = r^\rho \oplus \mathsf{E}(\hat{b}_i^\rho) \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$, which indicates the sorted index of original dataset and is the output of our SS protocol.

Note that we can further use the privacy-preserving XNOR gate to compute the sorted dataset. For the k-th data record, i.e., whose sum value is the k-th largest, one can obtain $\{\mathsf{E}(\mathrm{x}_k^j) \mid j \in [1,d]\}$ as follows.

$$\mathsf{E}(\mathrm{x}_k^j) = \sum_{i=1}^{n} \mathsf{E}(\mathrm{x}_i^j) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_k^\rho)) \qquad (4)$$

Fig. 3 demonstrates an example of our secure sort protocol, in which there are four time series from Fig. 1.

Correctness. We say our SS protocol is correct if the encrypted bit sequence set $\{\mathsf{E}(\delta_i^\rho) \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ represents the sorted indexes $\{b_i^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ according to the corresponding sum value σ_i. In our SS protocol, $\mathcal{CS}_2$ sorts the bit sequences $\{\hat{b}_i^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ according to their sum values $\{\hat{\sigma}_i \mid i \in [1,n]\}$. Since $r_0 > r_i$ and $\hat{\sigma}_i = r_0 \cdot \sigma_i + r_i$, $\{\hat{\sigma}_i \mid i \in [1,n]\}$ keeps the order of $\{\sigma_i \mid i \in [1,n]\}$. Meanwhile, since the privacy-preserving XOR gate has the same truth table as the original XOR gate and $\hat{b}_i^\rho = b_i^\rho \oplus r^\rho$, we have $\delta_i^\rho = r^\rho \oplus \hat{b}_i^\rho = r^\rho \oplus b_i^\rho \oplus r^\rho = b_i^\rho$. Besides, the sorted $\{\hat{b}_i^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ means the set $\{\mathsf{E}(\delta_i^\rho) \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ is sorted. Thus, $\{b_i^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ is sorted according to the corresponding sum value σ_i.

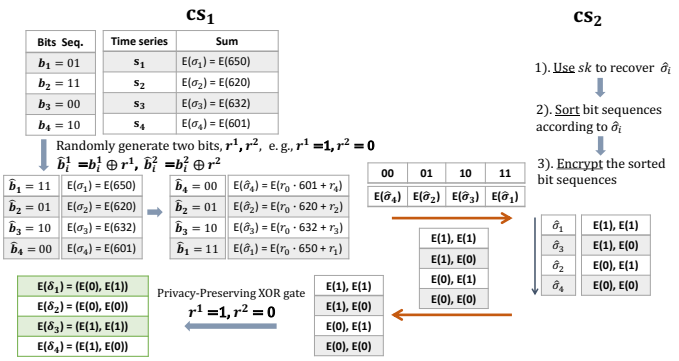


Fig. 3. Our secure sort protocol.

5.1.2 Secure Dominance Check (SDC) Protocol

Assume $\mathcal{CS}_1$ has two encrypted d-dimensional data records $\mathsf{E}(\vec{\mathrm{x}}_1), \mathsf{E}(\vec{\mathrm{x}}_2)$ and pk, while $\mathcal{CS}_2$ holds sk. The SDC protocol is to determine whether $\vec{\mathrm{x}}_1$ dominates $\vec{\mathrm{x}}_2$ without leaking the underlying plaintexts and the single-dimensional privacy to the cloud. If $\vec{\mathrm{x}}_1$ dominates $\vec{\mathrm{x}}_2$, the SDC protocol outputs $\mathsf{E}(1)$, otherwise $\mathsf{E}(0)$. Note that, since the d-dimensional data record can be treated as extracting a d-length time interval from the time series data, here the dominance relation between $\vec{\mathrm{x}}_1$ and $\vec{\mathrm{x}}_2$ is equivalent to the interval dominance (Definition 2) with any d length time interval. We show the details of the SDC protocol as follows.

- Step-1: First, for the ith-dimension, $\mathcal{CS}_1$ flips a coin $s_i \in \{-1, 1\}$ and chooses two random numbers $r_1^i, r_2^i \in \{0,1\}^{k_1}$ satisfying $r_1^i > r_2^i > 0$. Then, $\mathcal{CS}_1$ computes $\mathsf{E}(\theta^i) = \mathsf{E}(s_i \cdot r_1^i \cdot (\mathrm{x}_1^i - \mathrm{x}_2^i) + s_i \cdot r_2^i)$. If $s_i = 1$, let $\alpha^i = 2^i$ (hereafter, 2^i means 2 to the power of i), otherwise $\alpha^i = 0$. Next, $\mathcal{CS}_1$ computes $\alpha = \sum_{i=1}^{d} \alpha^i$ and sends $\{\mathsf{E}(\theta^i) \mid i \in [1,d]\}$ to $\mathcal{CS}_2$.
- Step-2: On receiving $\{\mathsf{E}(\theta^i) \mid i \in [1,d]\}$, $\mathcal{CS}_2$ first uses sk to recover θ^i. If $\theta^i > 0$, $\mathcal{CS}_2$ computes $\beta^i = 2^i$. Otherwise $\beta^i = 0$. Afterward, $\mathcal{CS}_2$ computes $\beta = \sum_{i=1}^{d} \beta^i$ and encrypts it into $\mathsf{E}(\beta)$. Next, $\mathcal{CS}_2$ sends $\mathsf{E}(\beta)$ to $\mathcal{CS}_1$.
- Step-3: With pk, $\mathcal{CS}_1$ encrypts α into $\mathsf{E}(\alpha)$. Then, $\mathcal{CS}_1$ runs the SEQ subprotocol, where the inputs are $\mathsf{E}(\alpha)$ and $\mathsf{E}(\beta)$, to test whether $\alpha = \beta$. If yes, the SEQ subprotocol returns $\mathsf{E}(\delta_1) = \mathsf{E}(1)$, otherwise $\mathsf{E}(\delta_1) = \mathsf{E}(0)$.
- Step-4: $\mathcal{CS}_2$ first adds up all dimensions of $\mathsf{E}(\vec{\mathrm{x}}_1)$ and $\mathsf{E}(\vec{\mathrm{x}}_2)$, i.e., $\mathsf{E}(\sigma_1) = \mathsf{E}(\sum_{i=1}^{d} \mathrm{x}_1^i)$ and $\mathsf{E}(\sigma_2) = \mathsf{E}(\sum_{i=1}^{d} \mathrm{x}_2^i)$. Next, $\mathcal{CS}_1$ runs the SBT subprotocol to determine whether $\sigma_1 > \sigma_2$. If yes, the SBT subprotocol returns $\mathsf{E}(\delta_2) = \mathsf{E}(1)$, otherwise $\mathsf{E}(\delta_2) = \mathsf{E}(0)$.
- Step-5: With $\mathsf{E}(\delta_1)$ and $\mathsf{E}(\delta_2)$, $\mathcal{CS}_1$ computes $\mathsf{E}(\delta) = \mathsf{E}(\delta_1) \cdot \mathsf{E}(\delta_2) = \mathsf{E}(\delta_1 \cdot \delta_2)$ as the output of the SDC protocol.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3153759, IEEE Transactions on Dependable and Secure Computing

Correctness. As the key idea of the SDC protocol is the same as the SDD protocol in [30], see the detailed correctness proof in [30].

5.1.3 Secure High-dimensional Dominance Check (SHDC) Protocol

With the same assumption as that in the SDC protocol, the SHDC protocol is to deal with the high-dimensional data. The stricter assumption of the SHDC protocol is that the sum value of $\mathsf{E}(\vec{\mathrm{x}}_1)$ is no less than that of $\mathsf{E}(\vec{\mathrm{x}}_2)$, namely, $\sigma_1 \geq \sigma_2$. In fact, it is easy to answer the assumption by applying our SS protocol. The main idea of our SHDC protocol is to convert the high-dimensional data to low dimensions and then apply the SDC protocol. To achieve it, we design a dominance check tree, denoted as DC-tree, in our SHDC protocol, which improves the performance when checking the dominance relation for high-dimensional data.

Here, we first introduce the DC-tree. For the non-leaf node, it has two children: left child and right child, and contains three fields: $\langle [j_1, j_2], \mathsf{E}(\sigma_1^{[j_1,j_2]}), \mathsf{E}(\sigma_2^{[j_1,j_2]})\rangle$. For the leaf node, it also has three fields $\langle [j_1, j_2], \mathsf{E}(\vec{\mathrm{x}}_1^{[j_1,j_2]}), \mathsf{E}(\vec{\mathrm{x}}_2^{[j_1,j_2]})\rangle$. We list the detailed description of these fields in Table 1.

TABLE 1
Fields of leaf and non-leaf nodes

Field	Description
$[j_1, j_2]$	a dimension range, $[j_1, j_2] \subseteq [1, d]$
$\mathsf{E}(\sigma_i^{[j_1,j_2]})$ $(i = 1, 2)$	the encrypted sum value of $\mathsf{E}(\vec{\mathrm{x}}_i)$ in the dimension range $[j_1, j_2]$. $\mathsf{E}(\sigma_i^{[j_1,j_2]}) = \sum_{j=j_1}^{j_2} \mathsf{E}(\mathrm{x}_i^j)$
$\mathsf{E}(\vec{\mathrm{x}}_i^{[j_1,j_2]})$ $(i = 1, 2)$	the sub vector of $\mathsf{E}(\vec{\mathrm{x}}_i)$ extracting from j_1 to j_2 dimension.

Unlike the traditional tree-based data structure used to retrieve data, our DC-tree is used to navigate and facilitate the dominance check. Specifically, the process of tree building is the running of our SHDC protocol. When the building process stops, our SHDC protocol outputs the dominance relation between $\mathsf{E}(\vec{\mathrm{x}}_1)$ and $\mathsf{E}(\vec{\mathrm{x}}_2)$. Next, we depict the process of our SHDC protocol as follows.

• Step-1: $\mathcal{CS}_1$ first uses the permutation π to permute these two data records $(\mathsf{E}(\mathrm{x}_i^1), \mathsf{E}(\mathrm{x}_i^2), \cdots, \mathsf{E}(\mathrm{x}_i^d))$ as $(\mathsf{E}(\mathrm{x}_i^{\pi(1)}), \mathsf{E}(\mathrm{x}_i^{\pi(2)}), \cdots, \mathsf{E}(\mathrm{x}_i^{\pi(d)}))$, where $i = 1, 2$. To simplify the description, we ignore the permutation symbol π in the protocol. Then, $\mathcal{CS}_1$ constructs the root node. Since the underlying plaintext of $\mathsf{E}(\sigma_1^{[1,d]})$ must be no less than that of $\mathsf{E}(\sigma_2^{[1,d]})$ (due to the assumption of $\sigma_1 \geq \sigma_2$), we mark the root node as $\langle [1, d], \perp, \perp\rangle$. After that, $\mathcal{CS}_1$ divides the d dimensional data records $\mathsf{E}(\vec{\mathrm{x}}_1)$ and $\mathsf{E}(\vec{\mathrm{x}}_2)$ into two parts, respectively, in which the left part is $\mathsf{E}(\vec{\mathrm{x}}_i^l) = (\mathsf{E}(\mathrm{x}_i^1), \mathsf{E}(\mathrm{x}_i^2), \cdots, \mathsf{E}(\mathrm{x}_i^{\lceil d/2 \rceil}))$ and the right part is $\mathsf{E}(\vec{\mathrm{x}}_i^r) = (\mathsf{E}(\mathrm{x}_i^{\lceil d/2 \rceil+1}), \mathsf{E}(\mathrm{x}_i^{\lceil d/2 \rceil+2}), \cdots, \mathsf{E}(\mathrm{x}_i^d))$. Next, $\mathcal{CS}_1$ computes the sum value of these two parts

$$\mathsf{E}(\sigma_i^l) = \mathsf{E}(\sigma_i^{[1,\lceil d/2 \rceil]}) = \sum_{j=1}^{\lceil d/2 \rceil} \mathsf{E}(\mathrm{x}_i^j)$$

$$\mathsf{E}(\sigma_i^r) = \mathsf{E}(\sigma_i^{[\lceil d/2 \rceil+1,d]}) = \sum_{j=\lceil d/2 \rceil+1}^{d} \mathsf{E}(\mathrm{x}_i^j),$$

and generates $\langle [1, \lceil d/2 \rceil], \mathsf{E}(\sigma_1^l), \mathsf{E}(\sigma_2^l)\rangle$ and $\langle [\lceil d/2 \rceil + 1, d], \mathsf{E}(\sigma_1^r), \mathsf{E}(\sigma_2^r)\rangle$ as the root node's left and right child nodes, respectively. After choosing random numbers $r_1, r_2 \in \{0, 1\}^{k_1}$ satisfying $r_1 > r_2 > 0$, $\mathcal{CS}_1$ computes $\mathsf{E}(\theta) = \mathsf{E}(r_1 \cdot (\sigma_2^l - \sigma_1^l) \cdot (\sigma_2^r - \sigma_1^r) + r_2)$ and sends it to $\mathcal{CS}_2$.

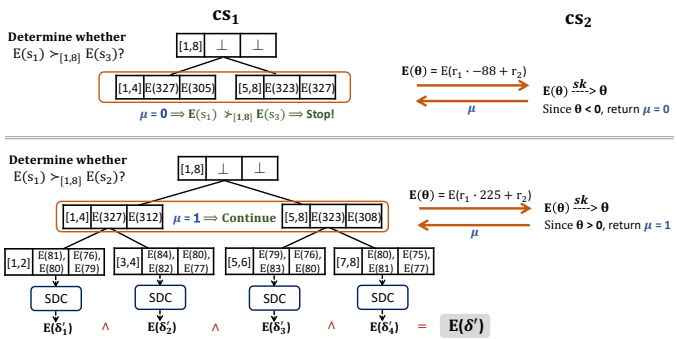


Fig. 4. Examples of DC-tree under the assumption of $\tau = 4$.

• Step-2: On receiving $\mathsf{E}(\theta)$, $\mathcal{CS}_2$ first uses sk to recover θ. If $\theta > 0$, $\mathcal{CS}_2$ makes $\mu = 1$. Otherwise $\mu = 0$. Then, $\mathcal{CS}_2$ sends μ to $\mathcal{CS}_1$.

• Step-3: If the received $\mu = 0$, $\mathcal{CS}_1$ will stop the SHDC protocol (stop building DC-tree) and make $\delta = 0$ as the output, which means $\mathsf{E}(\vec{\mathrm{x}}_1)$ does not dominate $\mathsf{E}(\vec{\mathrm{x}}_2)$. If $\mu = 1$, it indicates we cannot determine whether $\mathsf{E}(\vec{\mathrm{x}}_1)$ dominates $\mathsf{E}(\vec{\mathrm{x}}_2)$. As a result, $\mathcal{CS}_1$ makes $\mathsf{E}(\vec{\mathrm{x}}_1^l)$ and $\mathsf{E}(\vec{\mathrm{x}}_2^r)$ as new inputs and recursively performs Step-1 and Step-2. If the number of dimension of $\mathsf{E}(\vec{\mathrm{x}}_i^l)$ and $\mathsf{E}(\vec{\mathrm{x}}_i^r)$ is less than τ (usually $\tau \geq 4$), $\mathcal{CS}_1$ will not divide them and will treat them as leaf nodes: $\langle$*dimension range of* $\mathsf{E}(\vec{\mathrm{x}}_i^l), \mathsf{E}(\vec{\mathrm{x}}_1^l), \mathsf{E}(\vec{\mathrm{x}}_2^l)\rangle$ and $\langle$*dimension range of* $\mathsf{E}(\vec{\mathrm{x}}_i^r), \mathsf{E}(\vec{\mathrm{x}}_1^r), \mathsf{E}(\vec{\mathrm{x}}_2^r)\rangle$. When all leaf nodes are generated, $\mathcal{CS}_1$ performs **the modified SDC protocol** for each leaf node, where the inputs are the second and third fields. The modified SDC protocol does not run Step-4 and Step-5 in the original SDC protocol, and outputs $\mathsf{E}(\delta_1)$ (in Step-3 of SDC protocol) as the result. We suppose there are totally p leaf nodes and denote the output of the modified SDC protocol at each leaf node as $\mathsf{E}(\delta_i')$, where $i \in [1, p]$. Afterward, $\mathcal{CS}_1$ computes $\mathsf{E}(\delta') = \mathsf{E}(\delta_1') \wedge \mathsf{E}(\delta_2') \wedge \cdots \wedge \mathsf{E}(\delta_p') = \prod_{i=1}^{p} \mathsf{E}(\delta_i')$. Next, $\mathcal{CS}_1$ obtains $\mathsf{E}(\delta'')$ by performing SBT subprotocol, in which the inputs are $\mathsf{E}(\sigma_1)$ and $\mathsf{E}(\sigma_2)$. Finally, $\mathsf{E}(\delta) = \mathsf{E}(\delta') \cdot \mathsf{E}(\delta'') = \mathsf{E}(\delta' \cdot \delta'')$ will be the output of our SHDC protocol.

Fig. 4 illustrates two examples of DC-tree. The first one is to determine whether $\mathtt{s}_1$ dominates $\mathtt{s}_3$, and the second one is to determine whether $\mathtt{s}_1$ dominates $\mathtt{s}_2$, in which $\mathtt{s}_i$ ($i =$ 1 to 4) are from Fig. 1. For the first example, since one of the root node's children shows that the partial sum of $\mathtt{s}_2$ is larger than $\mathtt{s}_1$, it is definite that $\mathtt{s}_1$ does not dominates $\mathtt{s}_2$. Thus, we stop the tree building and output the result. For the second example, since $\mu = 1$, $\mathcal{CS}_1$ cannot determine whether $\mathtt{s}_1$ dominates $\mathtt{s}_2$. As the number of dimensions of the third level's nodes is less than 4, they are treated as leaf nodes and run the modified SDC protocol to determine the dominance relation.

Correctness. Since we assume that $\sigma_1 \geq \sigma_2$, $\vec{\mathrm{x}}_2$ certainly does not dominate $\vec{\mathrm{x}}_1$. Therefore, the SHDC protocol only needs to determine whether $\vec{\mathrm{x}}_1$ dominates $\vec{\mathrm{x}}_2$. We say our SHDC protocol is correct if it outputs $\mathsf{E}(1)$ when $\vec{\mathrm{x}}_1$ dominates $\vec{\mathrm{x}}_2$ and outputs $\mathsf{E}(0)$ or 0 when $\vec{\mathrm{x}}_1$ does not dominate $\vec{\mathrm{x}}_2$. First, we prove that when $\delta = 0$, it indicates $\vec{\mathrm{x}}_1$ does not dominate $\vec{\mathrm{x}}_2$. From Step-3, we know that when $\mu = 0$, we have $\delta = 0$. In this case, $\theta = r_1 \cdot (\sigma_2^l - \sigma_1^l) \cdot (\sigma_2^r - \sigma_1^r) + r_2 < 0$ (θ

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3153759, IEEE Transactions on Dependable and Secure Computing

Algorithm 1 Basic Skyline Computation Protocol

Input: An encrypted dataset $\{\mathsf{E}(\vec{\mathrm{x}}_i) \mid i \in [1,n]\}$. Assigned bit sequences for each data record, $\{b_i \mid i \in [1,n]\}$.
Output: A set containing encrypted skyline data records, $\mathcal{S}_{\mathrm{sky}}$.
1: **for** $i = 1$ to n **do**
2: $\quad \mathsf{E}(\sigma_i) = \mathsf{E}(i + (n+1) \cdot \sum_{j=1}^{d} \mathrm{x}_i^j)$;
3: $\{\mathsf{E}(\delta_i^\rho)\}$=SS($\{\mathsf{E}(\vec{\mathrm{x}}_i, \mathsf{E}(\sigma_i))\}$), $i \in [1,n], \rho \in [1, \lceil \log_2 n \rceil]$;
4: $\mathsf{E}(\mathrm{t}_1^j) = \sum_{i=1}^{n} \mathsf{E}(\mathrm{x}_i^j) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_1^\rho))$, $j \in [1,d]$;
5: $\mathcal{S}_{\mathrm{sky}}$.add($\mathsf{E}(\vec{\mathrm{t}}_1)$);
6: **for** $i = 2$ to n **do**
7: $\quad \mathsf{E}(\mathrm{p}_i^j) = \sum_{l=1}^{n} \mathsf{E}(\mathrm{x}_l^j) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_l^\rho \odot \mathsf{E}(\delta_i^\rho))$, $j \in [1,d]$;
8: $\quad$ flag = true;
9: $\quad$ **for** each data record $\mathsf{E}(\vec{\mathrm{t}})$ in $\mathcal{S}_{\mathrm{sky}}$ **do**
10: $\quad\quad \delta$= WSHDC($\mathsf{E}(\vec{\mathrm{t}})$, $\mathsf{E}(\vec{\mathrm{p}}_i)$);
11: $\quad\quad$ **if** $\delta == 1$ **then**
12: $\quad\quad\quad$ flag = false;
13: $\quad\quad\quad$ break;
14: $\quad$ **if** flag == true **then** $\mathcal{S}_{\mathrm{sky}}$.add($\mathsf{E}(\vec{\mathrm{p}}_i)$).
15: **return** $\mathcal{S}_{\mathrm{sky}}$.

is never equal to 0). As a result, there must exist $\sigma_2^l - \sigma_1^l > 0$ or $\sigma_2^r - \sigma_1^r > 0$. Consequently, $\exists j \in [1,d]$, $\vec{\mathrm{x}}_2^j > \vec{\mathrm{x}}_1^j$. Thus, $\vec{\mathrm{x}}_1$ does not dominate $\vec{\mathrm{x}}_2$. Then, we prove only $\mathsf{E}(\delta) = \mathsf{E}(1)$ indicates $\vec{\mathrm{x}}_1$ dominates $\vec{\mathrm{x}}_2$. Since $\mathsf{E}(\delta') = \prod_{i=1}^{p} \mathsf{E}(\delta_i')$, when $\mathsf{E}(\delta') = \mathsf{E}(1)$, it means $\forall j \in [1,d], \vec{\mathrm{x}}_1^j \geq \vec{\mathrm{x}}_2^j$. Further, $\mathsf{E}(\delta'') = \mathsf{E}(1)$ indicates $\exists j \in [1,d], \vec{\mathrm{x}}_1^j > \vec{\mathrm{x}}_2^j$. Therefore, only when $\mathsf{E}(\delta) = \mathsf{E}(1)$, $\vec{\mathrm{x}}_1$ dominates $\vec{\mathrm{x}}_2$.

In our SHDC protocol, although $\mathcal{CS}_1$ knows $\mathsf{E}(\vec{\mathrm{x}}_1)$ does not dominate $\mathsf{E}(\vec{\mathrm{x}}_2)$ in some cases, it only knows there exists one dimension j that $\vec{\mathrm{x}}_2^j > \vec{\mathrm{x}}_1^j$. Since we limit the leaf node to have at least 2 ($\tau/2 > 2$) dimensions, and they are checked as a whole in the modified SDC protocol, it guarantees the single-dimensional privacy, i.e., $\mathcal{CS}_1$ cannot infer the order relations of one dimension. Thus, our SHDC protocol can preserve single-dimensional privacy. For efficiency, our SHDC protocol will stop if the non-leaf node offers the no dominance relation, which significantly improves the performance in dealing with high-dimensional data. See Section 7.2 for performance evaluations.

5.2 Privacy-Preserving Skyline Computation

Based on the above secure protocols, we present our privacy-preserving skyline computation protocols. First, we introduce a basic protocol that is efficient but leaks single-dimensional privacy in some cases. Then, we introduce a secure protocol without leaking plaintexts, single-dimensional privacy, and access patterns.

5.2.1 Basic Skyline Computation Protocol

The main idea of the straw-man protocol is to leak the dominance relations to $\mathcal{CS}_1$ so that it can directly adopt the SFS algorithm to compute skyline. See Algorithm 1 for the protocol. $\mathcal{CS}_1$ first adds up the value of all dimensions $\mathsf{E}(\sigma_i) = \mathsf{E}(\sum_{j=1}^{d} \mathrm{x}_i^j)$ for each data record. To ensure the same data record holding different sum values, $\mathcal{CS}_1$ updates $\mathsf{E}(\sigma_i)$ by $\mathsf{E}(\sigma_i) = \mathsf{E}(\sigma_i \cdot (n+1) + i)$ showing in lines 1-2. Then, $\mathcal{CS}_1$ uses the SS protocol to sort the original dataset, which not only sorts the dataset but also breaks the link between the sorted dataset and the original dataset. Next, the first data record in the sorted dataset must be a skyline point, as shown in lines 3-5. In this protocol, we weaken our SHDC protocol, denoted as WSHDC (see line 10), by returning a plaintext δ to $\mathcal{CS}_1$. In this way, $\mathcal{CS}_1$ can know $\mathsf{E}(\vec{\mathrm{x}}_1)$ dominates $\mathsf{E}(\vec{\mathrm{x}}_2)$ if $\delta = 1$, and $\mathsf{E}(\vec{\mathrm{x}}_1)$ does not dominate $\mathsf{E}(\vec{\mathrm{x}}_2)$ if $\delta = 0$. After checking dominance relation of each data record, $\mathcal{CS}_1$ obtains a skyline set, in which data records are not dominated with each other.

Algorithm 2 Secure Skyline Computation Protocol

Input: An encrypted dataset $\{\mathsf{E}(\vec{\mathrm{x}}_i) \mid i \in [1,n]\}$. Assigned bit sequences for each data record, $\{b_i \mid i \in [1,n]\}$.
Output: A set containing encrypted skyline data records, $\mathcal{S}_{\mathrm{sky}}$.
1: $\mathcal{X} \leftarrow \varnothing, \mathcal{S} \leftarrow \varnothing, \mathcal{F} \leftarrow \varnothing$;
2: **for** $i = 1$ to n **do**
3: $\quad \mathsf{E}(\sigma_i) = \mathsf{E}(i + n + (n+1) \cdot \sum_{j=1}^{d} \mathrm{x}_i^j)$;
4: $\quad \mathcal{X}$.add($\mathsf{E}(\vec{\mathrm{x}}_i)$); $\mathcal{S}$.add($\mathsf{E}(\sigma_i)$); $\mathcal{F}$.add($\mathsf{E}(0)$);
5: $\{\mathsf{E}(\delta_i^\rho)\}$=SS($\{\mathsf{E}(\vec{\mathrm{x}}_i)\}, \{\mathsf{E}(\sigma_i)\}$), $i \in [1,n], \rho \in [1, \lceil \log_2 n \rceil]$;
6: $\mathsf{E}(\sigma_{\max}) = \sum_{i=1}^{n} \mathsf{E}(\sigma_i) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_1^\rho))$;
7: $\mathsf{E}(\sigma_{\min}) = \sum_{i=1}^{n} \mathsf{E}(\sigma_i) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_n^\rho))$;
8: $\mathsf{E}(\mathtt{MIN}) = \mathsf{E}(\sigma_{\min}) + \mathsf{E}(-1) = \mathsf{E}(\sigma_{\min} - 1)$;
9: cnt = 1, N = n;
10: **while** BIG($\mathsf{E}(\sigma_{\max})$, $\mathsf{E}(\mathtt{MIN})$) && $\mathcal{S}_{\mathrm{sky}}$.size $<$ N **do**
11: $\quad \mathsf{E}(\vec{\mathrm{t}}_1) = \sum_{i=1}^{n} \mathcal{X}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_1^\rho))$;
12: $\quad \mathcal{S}_{\mathrm{sky}}$.add($\mathsf{E}(\vec{\mathrm{t}}_1)$);
13: $\quad$ **for** $l = 2$ to n **do**
14: $\quad\quad \mathsf{E}(\vec{\mathrm{p}}_l) = \sum_{i=1}^{n} \mathcal{X}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_l^\rho))$;
15: $\quad\quad$ flag = true;
16: $\quad\quad$ **for** each data record $\mathsf{E}(\vec{\mathrm{t}})$ in $\mathcal{S}_{\mathrm{sky}}$ **do**
17: $\quad\quad\quad \mathsf{E}(\delta)$= SHDC($\mathsf{E}(\vec{\mathrm{t}})$, $\mathsf{E}(\vec{\mathrm{p}}_l)$);
18: $\quad\quad\quad$ **if** $\mathsf{E}(\delta) \neq 0$ **then**
19: $\quad\quad\quad\quad$ flag = false;
20: $\quad\quad\quad\quad$ **break**;
21: $\quad\quad$ **if** flag == true **then**
22: $\quad\quad\quad \mathcal{S}_{\mathrm{sky}}$.add($\mathsf{E}(\vec{\mathrm{p}}_l)$).
23: $\quad\quad$ **else**
24: $\quad\quad\quad$ Define $\{\mathsf{E}(\delta_i^\rho), b_i\}$, $i \in [1,n], \rho \in [1, \lceil \log_2 n \rceil]$ as a set $\mathcal{I}$;
25: $\quad\quad\quad$ ($\mathcal{X}$, $\mathcal{S}$, $\mathcal{F}$) = updateSets($\mathcal{X}$, $\mathcal{S}$, $\mathcal{F}$, l, cnt, $\mathcal{I}$, $\mathsf{E}(\mathtt{MIN})$, $\mathcal{S}_{\mathrm{sky}}$);
26: $\quad\quad\quad n = \mathcal{X}$.size, cnt = $\mathcal{S}_{\mathrm{sky}}$.size;
27: $\quad\quad\quad$ Assign new bit sequences $\{b_i \mid i \in [1,n]\}$.
28: $\quad\quad\quad \{\mathsf{E}(\delta_i^\rho)\}$=SS($\mathcal{X}, \mathcal{S}$), $i \in [1,n], \rho \in [1, \lceil \log_2 n \rceil]$;
29: $\quad\quad\quad \mathsf{E}(\sigma_{\max}) = \sum_{i=1}^{n} \mathcal{S}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_1^\rho))$;
30: $\quad\quad\quad$ **break**;
31: **return** $\mathcal{S}_{\mathrm{sky}}$.

In the basic protocol, if there is no dominance relation between two data records, $\mathcal{CS}_1$ cannot know the actual order relation of each dimension, i.e., the basic protocol can ensure the single-dimensional privacy. Only when there is dominance relation between them, $\mathcal{CS}_1$ knows that $\forall j, \vec{\mathrm{x}}_1^j > (\text{or} \geq)\ \vec{\mathrm{x}}_2^j$, which is determined by the definition of dominance. However, the SS protocol is adopted to sort the original dataset, which makes $\mathcal{CS}_1$ learn nothing about which two data records (in original dataset) have the dominance relation.

5.2.2 Secure Skyline Computation Protocol

To obtain high efficiency, the basic protocol leaks single-dimensional privacy in some cases. In order to preserve this privacy, the intuitive approach is to use our SHDC protocol and output encrypted value $\mathsf{E}(\delta)$. However, it makes Algorithm 1 unavailable, since $\mathcal{CS}_1$ cannot determine whether one data record dominates the other one with the encrypted value. To tackle this issue, we design a secure skyline computation protocol that can find skylines in a secure manner, i.e., protecting underlying plaintexts, single-dimensional privacy, and access patterns.

Algorithm 2 shows our secure skyline computation protocol. Compared with the basic protocol, this protocol uses the SHDC instead of WSHDC to check dominance relations. Consequently, there are two cases:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3153759, IEEE Transactions on Dependable and Secure Computing

Algorithm 3 Update Sets

Input: Three encrypted sets, $\mathcal{X}$, $\mathcal{S}$, $\mathcal{F}$. A start point, l. A counter, cnt. A sorted index set, $\mathcal{I}$. An encrypted minimum sum, $\mathsf{E}(\mathtt{MIN})$. A skyline set, $\mathcal{S}_{\text{sky}}$.

Output: Three new encrypted sets, $\mathcal{X}_{\text{new}}, \mathcal{S}_{\text{new}}, \mathcal{F}_{\text{new}}$.

1: $\mathcal{X}_{\text{new}} \leftarrow \varnothing, \mathcal{S}_{\text{new}} \leftarrow \varnothing, \mathcal{F}_{\text{new}} \leftarrow \varnothing, n = \mathcal{X}.\text{size}, \{\mathsf{E}(\hat{\delta}_i^{\rho}), b_i\} = \mathcal{I}$;
2: **for** $k = l$ to n **do**
3: $\quad \mathsf{E}(\mathtt{p}_k^j) = \sum_{i=1}^{n} \mathcal{X}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^{\rho} \odot \mathsf{E}(\delta_k^{\rho}))$, $j \in [1, d]$;
4: $\quad \mathsf{E}(\sigma_k) = \sum_{i=1}^{n} \mathcal{S}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^{\rho} \odot \mathsf{E}(\delta_k^{\rho}))$;
5: $\quad \mathsf{E}(f_k) = \sum_{i=1}^{n} \mathcal{F}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^{\rho} \odot \mathsf{E}(\delta_k^{\rho}))$;
6: $\quad \mathcal{X}_{\text{new}}.\text{add}(\mathsf{E}(\vec{\mathtt{p}}_k))$, $\mathcal{S}_{\text{new}}.\text{add}(\mathsf{E}(\sigma_k))$, $\mathcal{F}_{\text{new}}.\text{add}(\mathsf{E}(f_k))$;
7: **for** $m =$ cnt to $\mathcal{S}_{\text{sky}}.\text{size}$ **do**
8: $\quad \mathsf{E}(\vec{\mathtt{t}}_m) = \mathcal{S}_{\text{sky}}[m]$;
9: $\quad$ **for** $k = l$ to n **do**
10: $\quad\quad \mathsf{E}(\delta) = \text{SHDC}(\mathsf{E}(\vec{\mathtt{t}}_m), \mathcal{X}_{\text{new}}[k - l + 1])$;
11: $\quad\quad$ **if** $\mathsf{E}(\delta) == 0$ **then** $\mathsf{E}(\delta) = \mathsf{E}(0)$;
12: $\quad\quad \mathcal{F}_{\text{new}}.\text{update}(k - l + 1, \mathcal{F}_{\text{new}}[k - l + 1] \vee \mathsf{E}(\delta))$;
13: **for** $k = l$ to n **do**
14: $\quad$ randomly choose $r \in \mathbb{Z}_n$ and generate $\mathsf{E}(r)$;
15: $\quad \mathsf{E}(\sigma_{\min}) = \mathsf{E}(\mathtt{MIN}) + \mathsf{E}(r) \cdot \mathsf{E}(-1) = \mathsf{E}(\mathtt{MIN} - r)$;
16: $\quad \mathsf{E}(\sigma_{\text{old}}) = \mathcal{S}_{\text{new}}[k - l + 1]$;
17: $\quad \mathsf{E}(\sigma) = \mathcal{F}_{\text{new}}[k - l + 1] \cdot (\mathsf{E}(\sigma_{\min}) - \mathsf{E}(\sigma_{\text{old}})) + \mathsf{E}(\sigma_{\text{old}})$;
18: $\quad \mathcal{S}_{\text{new}}.\text{update}(k - l + 1, \mathsf{E}(\sigma))$;
19: **return** $(\mathcal{X}_{\text{new}}, \mathcal{S}_{\text{new}}, \mathcal{F}_{\text{new}})$.

Case 1: When the non-leaf node shows there is no dominance relation between two tested data records, the SHDC protocol returns 0. If the candidate data record is not dominated by all skyline points, it will be directly added to the skyline set. See lines 13-22 for details.

Case 2: When all leaf nodes are checked by the modified SDC protocol, the SHDC protocol returns $\mathsf{E}(\delta)$. In this case, $\mathcal{CS}_1$ computes skyline by the fact that the data record that has the maximum sum value must be a skyline point, as shown in lines 24-30 and lines 11-12. The main idea is to maintain a dominance flag set $\mathcal{F} = \{\mathsf{E}(f_i) \mid i \in [1, n]\}$ and an encrypted sum set $\mathcal{S} = \{\mathsf{E}(\sigma_i) \mid i \in [1, n]\}$. Once the SHDC protocol outputs an encrypted dominance flag for the input data record, $\mathcal{CS}_1$ updates the following data records' (note that we have sorted the dataset by our SS protocol) dominance flags and sum values, as shown in Algorithm 3.

Update dominance flag: Suppose a data record already has a dominance flag $\mathsf{E}(f_{\text{old}})$, $f_{\text{old}} \in \{0, 1\}$. For the data record, our SHDC protocol outputs an encrypted dominance flag $\mathsf{E}(\delta)$, $\delta \in \{0, 1\}$. With these two encrypted data, $\mathcal{CS}_1$ updates the dominance flag as follows:

$$\begin{aligned} \mathsf{E}(f_{\text{new}}) &= \mathsf{E}(f_{\text{old}}) \vee \mathsf{E}(\delta) \\ &= \mathsf{E}(f_{\text{old}}) + \mathsf{E}(\delta) + \mathsf{E}(f_{\text{old}}) \cdot \mathsf{E}(\delta) \cdot \mathsf{E}(-1) \\ &= \mathsf{E}(f_{\text{old}} + \delta - f_{\text{old}} \cdot \delta). \end{aligned} \tag{5}$$

It indicates that once the data record is dominated by one skyline point, its final dominance flag must be $\mathsf{E}(1)$. This updating process is illustrated in lines 7-12 of Algorithm 3.

Update sum value: Suppose a data record already has an encrypted sum value $\mathsf{E}(\sigma_{\text{old}})$, and the data record's up-to-date dominance flag is $\mathsf{E}(f)$. Meanwhile, we assume $\mathcal{CS}_1$ has already computed a minimum sum value $\mathsf{E}(\mathtt{MIN})$ (lines 7-8 in Algorithm 2), which guarantees that all sum values of the original dataset must be larger than $\mathtt{MIN}$. First, $\mathcal{CS}_1$ chooses a random number $r \in \mathbb{Z}_n$ and generates $\mathsf{E}(r)$ by Eq. (1). Then, $\mathcal{CS}_1$ obtains a new minimum sum value: $\mathsf{E}(\sigma_{\min}) = \mathsf{E}(\mathtt{MIN}) + \mathsf{E}(r) \cdot \mathsf{E}(-1) = \mathsf{E}(\mathtt{MIN} - r)$. Next, $\mathcal{CS}_1$ updates the data record's sum value as follows:

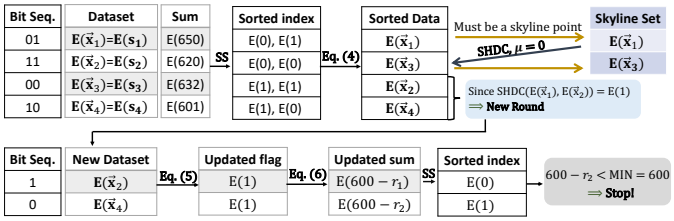


Fig. 5. An exemplary secure skyline computation protocol under the assumption of $r_2 < r_1 \in \mathbb{Z}_n$.

$$\begin{aligned} \mathsf{E}(\sigma_{\text{new}}) &= \mathsf{E}(f) \cdot (\mathsf{E}(\sigma_{\min} - \sigma_{\text{old}})) + \mathsf{E}(\sigma_{\text{old}}) \\ &= \mathsf{E}(f \cdot (\sigma_{\min} - \sigma_{\text{old}}) + \sigma_{\text{old}}). \end{aligned} \tag{6}$$

If $f = 0$, $\mathsf{E}(\sigma_{\text{new}}) = \mathsf{E}(\sigma_{\text{old}})$. On the other hand, if $f = 1$, $\mathsf{E}(\sigma_{\text{new}}) = \mathsf{E}(\sigma_{\min})$. In this case, $\mathsf{E}(\sigma_{\text{new}})$ must be less than $\mathsf{E}(\mathtt{MIN})$ due to $\mathsf{E}(\sigma_{\min}) = \mathsf{E}(\mathtt{MIN} - r)$. Besides, since $\mathtt{MIN} > n + 1$ and $r \in \mathbb{Z}_n$, $\mathtt{MIN} - r > 0$. See details of updating sum values from line 13 to line 18 in Algorithm 3.

If all sum values are updated, $\mathcal{CS}_1$ checks whether the largest sum value is bigger than $\mathsf{E}(\mathtt{MIN})$ (line 29 and line 31 in Algorithm 2). If yes, $\mathcal{CS}_1$ launches a new round to find new skyline points. Otherwise, $\mathcal{CS}_1$ stops the protocol. When all sum values are less than $\mathsf{E}(\mathtt{MIN})$, it means all data records in the current set are dominated. Here, the BIG protocol can be achieved by making $\mathcal{CS}_2$ return plaintext μ in the SBT subprotocol. Note that, since the updated values (dominates flags and sum values) usually involve several homomorphic multiplications in each round, we adopt the bootstrapping protocol in [20] to refresh these ciphertexts.

Fig. 5 shows an example of our secure skyline computation protocol, in which there are four time series from Fig. 1. The sorted indexes are computed by our SS protocol, seeing Fig. 3 for details. In the example, $\mathsf{E}(\vec{\mathtt{x}}_1)$ must be a skyline point, as it has the largest sum value. With SHDC protocol, we can obtain that $\mathsf{E}(\vec{\mathtt{x}}_1)$ does not dominate $\mathsf{E}(\vec{\mathtt{x}}_3)$ (see the first example in Fig. 4). Therefore, $\mathsf{E}(\vec{\mathtt{x}}_3)$ is a skyline point.

5.3 Description of Our Proposed Scheme

In this subsection, we present our privacy-preserving interval skyline query scheme, which is comprised of four phases: 1) system initialization; 2) data reporting and organization; 3) interval skyline search; and 4) data recovery.

5.3.1 System Initialization

In our scheme, the service provider $\mathcal{SP}$ initializes the entire system. First, given security parameters (k_0, k_1, k_2), $\mathcal{SP}$ calls $\mathsf{KeyGen}(k_0, k_1, k_2)$ of SHE to generate the secret key sk and pp. Meanwhile, $\mathcal{SP}$ generates $pk = \{\mathsf{E}(0)_1, \mathsf{E}(0)_2, pp\}$. Then, $\mathcal{SP}$ chooses a secure hash function $\mathsf{H}()$, e.g., SHA-256, and generates two master keys k_p, k_u for data providers and data users, respectively.

- When a data provider p_i with its identity $\mathtt{ID}_{pi}$ registers to the system, $\mathcal{SP}$ authorizes $k_{pi} = \mathsf{H}(\mathtt{ID}_{pi}, k_p)$ to p_i.
- When a data user u_i with the identity $\mathtt{ID}_{ui}$ registers to the system, $\mathcal{SP}$ authorizes $k_{ui} = \mathsf{H}(\mathtt{ID}_{ui}, k_u)$ to u_i.

Besides, since the recent data is often considered more important in time series data [6], it is practical to maintain the most recent timestamps. As a result, $\mathcal{SP}$ needs to choose the size w (sliding window) for the most recent timestamps.

Finally, $\mathcal{SP}$ publishes $\{\mathsf{H}(), w, pk\}$, sends $\{k_p, k_u\}$ to $\mathcal{CS}_1$, and authorizes sk to $\mathcal{CS}_2$.

5.3.2 Data Reporting and Organization

If a data provider p_i has a sensed data x_i^j at a time stamp t_j, it will encrypt x_i^j into $\mathsf{E}(\mathrm{x}_i^j)$ by using Eq. (1). Then, p_i computes $\mathsf{H}(\mathsf{E}(\mathrm{x}_i^j), k_{pi})$ with the authorized key k_{pi} and sends $\langle p_i, t_j, \mathsf{E}(\mathrm{x}_i^j)\rangle || \mathtt{ID}_{pi} || \mathsf{H}(\mathsf{E}(\mathrm{x}_i^j), k_{pi})$ to $\mathcal{CS}_1$.

Upon receiving it, $\mathcal{CS}_1$ first computes $k_{pi} = \mathsf{H}(\mathtt{ID}_{pi}, k_p)$ with the authorized k_p and then calculates $\mathsf{H}(\mathsf{E}(\mathrm{x}_i^j), k_{pi})$. Next, $\mathcal{CS}_1$ extracts $\mathsf{H}(\mathsf{E}(\mathrm{x}_i^j), k_{pi})$ from the received message and checks whether the calculated $\mathsf{H}(\mathsf{E}(\mathrm{x}_i^j), k_{pi})$ is the same as the extracted one. If yes, $\mathcal{CS}_1$ accepts the reported data, otherwise rejects. Afterward, $\mathcal{CS}_1$ randomly generates a unique bit sequence $b_i = b_i^{\rho}|_{\rho=1}^{\lceil \log_2 n \rceil}$ for p_i, where n is the number of data providers, and $b_i^{\rho} \in \{0, 1\}$. If a new data is reported, the data in t_1 column will be expired, and a new column t_{w+1} will be added. Note that, similar to [6], we assume all time series are synchronized.

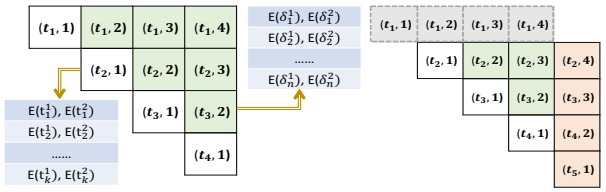
5.3.3 Interval Skyline Search

When a data user u_i wants to use the interval skyline query services, he/she can first define a time interval $[t_{j_1} : t_{j_2}] \subset [t_1 : t_w]$, $t_{j_1} < t_{j_2}$, and computes $\mathsf{H}(t_{j_1}||t_{j_2}, k_{ui})$ with the authorized key k_{ui}. Then, u_i sends $[t_{j_1} : t_{j_2}]||\mathtt{ID}_{ui}||\mathsf{H}(t_{j_1}||t_{j_2}, k_{ui})$ to $\mathcal{CS}_1$. Using the same approach introduced in the data reporting and organization phase, $\mathcal{CS}_1$ checks whether $\mathsf{H}(t_{j_1}||t_{j_2}, k_{ui})$ is correct. If yes, $\mathcal{CS}_1$ first extracts the encrypted data from t_{j_1} to t_{j_2} for all data provides and generates an encrypted dataset $\{\mathsf{E}(\vec{\mathrm{x}}_i) = (\mathsf{E}(\mathrm{x}_i^1), \mathsf{E}(\mathrm{x}_i^2), \cdots, \mathsf{E}(\mathrm{x}_i^{j_2-j_1+1})) \mid i \in [1, n]\}$. Then, $\mathcal{CS}_1$ runs our secure skyline computation protocol (Section 5.2.2) to obtain the desired interval skyline points $\{\mathsf{E}(\vec{\mathrm{t}}_i) \mid i \in [1, k]\}$, where k is the number of skyline points.

To quickly respond to the interval skyline query, we design a 2-dimensional look-up table, as shown in Fig. 6(a), to index the interval skyline points. First, $\mathcal{CS}_1$ generates the 2-dimensional table, in which each cell is a pair: $\langle$*start time*, *time step*$\rangle$. For example, $\langle t_2, 3\rangle$ indicates the time interval $[t_2, t_5]$. If a data user launches an interval skyline query $[t_2, t_5]$, $\mathcal{CS}_1$ stores the computed results under the cell after responding to the query:

- If the time step is less than $\gamma \cdot \log_2 n$, where $\gamma \geq n/k$ is the balance ratio, $\mathcal{CS}_1$ directly stores the computed skyline set $\mathcal{S}_{\text{sky}}$ under the cell.
- If the time step is larger than or equal to $\gamma \cdot \log_2 n$, $\mathcal{CS}_1$ stores the sorted indexes $\{\mathsf{E}(\delta_i^{\rho})\}, i \in [1, n], \rho \in [1, \lceil \log_2 n \rceil]$.

This strategy can balance the storage costs and computational costs. When another query request falls in the cell (the time complexity of locating cell is O(1)), $\mathcal{CS}_1$ can directly obtain the interval skyline or skip the SS protocol to compute the interval skyline. It is worth noting that we can store computed interval skyline for each cell. However, it may consume more storage overhead than the above strategy. This is because one of the characteristics of the time series data is high-dimensional. When the range of time interval $[t_{j_1} : t_{j_2}]$ is large, storing sorted indexes can limit the number of ciphertexts to $\log_2 n$ for each item and make it independent with the range of time interval. In the example of Fig. 6(a), if we suppose $n = 4$ and $\gamma = 1$, all the cells, whose time step is 1, will store the skyline set. Others will store the sorted index. Meanwhile, it is simple and efficient for the 2-dimensional table to support the continuous updates over time. Fig. 6(b) shows that t_1 is expired, while t_6 is added. It is worth noting that only the cell (time interval) has been queried, the computed results would be stored under the cell. Otherwise, the cell is empty. This mechanism is built according to the following facts: i) some cells may be queried frequently. ii) some cells may be expired before being queried.



(a) 2-dimensional table (b) Update

Fig. 6. Two-dimensional look-up table, where $w = 5$ ranges from t_1 to t_5.

After obtaining the encrypted skyline set, $\mathcal{CS}_1$ will generate a random number r and compute $\{r_i^j = \mathsf{H}(i||j, r) \mid i \in [1, k], j \in [1, d], r_i^j \in \{0, 1\}^{k_1}\}$. Then, $\mathcal{CS}_1$ adds these random noises to the corresponding skyline point: $\mathsf{E}(\mathrm{t}_i^j + r_i^j)$. Next, $\mathcal{CS}_1$ sends $\{\mathsf{E}(\mathrm{t}_i^j + r_i^j) \mid i \in [1, k], j \in [1, d]\}$ to $\mathcal{CS}_2$ and forwards r to the data user u_i via a secure channel. For $\mathcal{CS}_2$, it will recover $\{\mathrm{t}_i^j + r_i^j \mid i \in [1, k], j \in [1, d]\}$ with the secret key sk of SHE and returns them to u_i.

5.3.4 Data Recovery

Upon receiving r from $\mathcal{CS}_1$ and $\{\mathrm{t}_i^j + r_i^j \mid i \in [1, k], j \in [1, d]\}$ from $\mathcal{CS}_2$, u_i first computes $\{r_i^j = \mathsf{H}(i||j, r) \mid i \in [1, k], j \in [1, d]\}$. Then, u_i removes the random noises r_i^j from $\{\mathrm{t}_i^j + r_i^j \mid i \in [1, k], j \in [1, d]\}$. Finally, u_i can obtain the skyline point $\{\mathrm{t}_i^j \mid i \in [1, k], j \in [1, d]\}$ of the time interval $[t_{j_1} : t_{j_2}]$.

6 Security Analysis

In this section, we analyze the security of our proposed scheme. Specifically, we will prove that our secure skyline computation protocol can protect data records, query results, single-dimensional privacy, and access patterns.

Theorem 1. ***(Composition Theorem)****. If a protocol consists of several subprotocols, the protocol is secure as long as the subprotocols are secure and all the intermediate results are random or pseudo-random.*

We refer readers to [31] for the detailed proof of the composition theorem. In Fig. 7, we present the dependencies of our proposed secure protocols. It is clear that we should first prove that the SBT and SEQ subprotocols are secure, which can demonstrate the security of SDC and SHDC protocols according to Theorem 1. After analyzing the security of SS, SDC, and SHDC, we can obtain the security of our secure skyline computation protocol.

First of all, we briefly review the security model for securely realizing an ideal functionality in the presence of non-colluding semi-honest adversaries [31], [32].

Real-world execution. The real-world execution of a protocol Π takes place between $\{\mathcal{CS}_1, \mathcal{CS}_2\}$ and adversaries

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3153759, IEEE Transactions on Dependable and Secure Computing

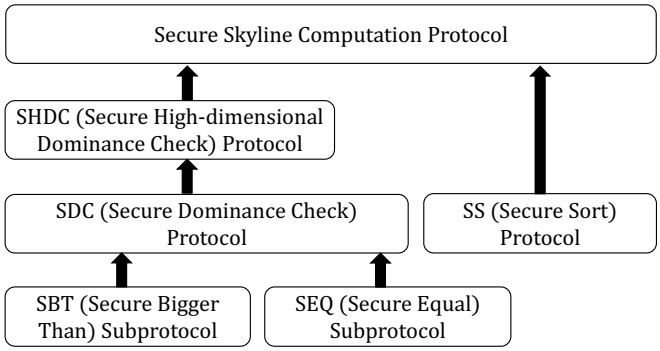


Fig. 7. Dependencies of secure protocols

$\{\mathcal{A}_1, \mathcal{A}_2\}$, who corrupt $\mathcal{CS}_1$ and $\mathcal{CS}_2$, respectively. With the input x_i and auxiliary input z_i $(i = 1, 2)$, e.g., the length of ciphertexts, the view of each party in the real-world execution protocol Π in the presence of adversary $\mathcal{A}_1$ ($\mathcal{A}_2$) is defined as

$\text{REAL}_{\Pi,\mathcal{A}_i,z_i}(x_i) \stackrel{def}{=} \text{OUT}_i^{\Pi}, i = 1, 2.$

Ideal-world execution. In the ideal world execution, there is an ideal functionality $\mathcal{F}$ for a function f, and the servers interact only with $\mathcal{F}$. Here, the view of each party in an ideal-world execution in the presence of independent simulators $\{\mathtt{Sim}_1, \mathtt{Sim}_2\}$ is defined as

$\text{IDEAL}_{\mathcal{F},\mathtt{Sim}_i,z_i}(x_i) \stackrel{def}{=} \text{OUT}_i^{\mathcal{F}}, i = 1, 2.$

In the above definitions, OUT_i is the output of parties.

Definition 4 (Security against semi-honest adversaries)**.** *Let $\mathcal{F}$ be a deterministic functionality and Π be a protocol between two parties (servers). We say that Π securely realizes $\mathcal{F}$ if there exists $\{\mathtt{Sim}_1, \mathtt{Sim}_2\}$ of PPT (Probabilistic Polynomial Time) transformations (where $\mathtt{Sim}_i = \mathtt{Sim}_i(\mathcal{A}_i), i = 1, 2$) such that for semi-honest PPT adversaries $\{\mathcal{A}_1, \mathcal{A}_2\}$, for all x_i and z_i, for each party holds:*

$$REAL_{\Pi,\mathcal{A}_i,z_i}(x_i) \stackrel{c}{\approx} IDEAL_{\mathcal{F},\mathtt{Sim}_i,z_i}(x_i)$$

where $\stackrel{c}{\approx}$ compactly denotes computational indistinguishability.

6.1 Privacy Preservation of SBT and SEQ Subprotocols

In this subsection, we use Definition 4 to show that the SBT subprotocol is secure, i.e., it can preserve the privacy of plaintexts.

Theorem 2. *The SBT subprotocol securely determines the bigger than relation in the presence of semi-honest (non-colluding) adversaries $\{\mathcal{A}_1, \mathcal{A}_2\}$.*

Proof. Here, we show how to construct the independent simulators: $\{\mathtt{Sim}_1, \mathtt{Sim}_2\}$.

$\mathtt{Sim}_1$ randomly chooses $\{m_1', m_2', \delta'\}$ and simulates $\mathcal{A}_1$ as follows. It first generates ciphertexts $\{\mathsf{E}(m_1'), \mathsf{E}(m_2'), \mathsf{E}(\delta')\}$ by the encryption algorithm of SHE. Then, it outputs $\mathcal{A}_1$'s entire view. In the real execution, $\mathcal{A}_1$ receives the ciphertext of $\{m_1, m_2, \delta\}$ whereas $\mathtt{Sim}_1$ gives $\{\mathsf{E}(m_1'), \mathsf{E}(m_2'), \mathsf{E}(\delta')\}$ to $\mathcal{A}_1$ in the ideal execution. The semantic security of SHE [20] guarantees that the views of $\mathcal{A}_1$ in the real and the ideal executions are indistinguishable.

$\mathtt{Sim}_2$ runs $\mathcal{A}_2$ by randomly choosing θ' and sending it to $\mathcal{A}_2$, which is the view of $\mathcal{A}_2$ in the ideal execution. In the real execution, $\mathcal{A}_2$ receives $\theta = s \cdot r_1 \cdot (m_1 - m_2) - s \cdot r_2$. Since r_1, r_2 and s are randomly generated, the views of $\mathcal{A}_2$ in the real and the ideal executions are indistinguishable. Note that, since θ' is randomly chosen, it means μ' will also be randomly generated. $\mathcal{A}_2$ cannot distinguish the views in μ' and μ. □

From the above proof, we can see that the SBT subprotocol ensures the security of the plaintext of $\{m_1, m_2\}$ and the result δ, i.e., preserving the privacy of inputs and order relations. In the SBT subprotocol, since $r_2 < r_1$, and s is randomly generated, $\mathcal{CS}_2$ cannot infer whether $m_1 = m_2$ when $m_1 = m_2$. Therefore, the SBT subprotocol is privacy-preserving.

Theorem 3. *The SEQ subprotocol securely determines the equality relation in the presence of semi-honest (non-colluding) adversaries $\{\mathcal{A}_1, \mathcal{A}_2\}$.*

Proof. Similar to the proof of Theorem 2, we can adopt Definition 4 to show that the SEQ subprotocol is secure. That is, it can preserve the privacy of plaintexts and equality relations. □

6.2 Privacy Preservation of SS, SDC, and SHDC Protocols

Here, we show that our SS protocol is secure, i.e., it can preserve the privacy of plaintexts.

Theorem 4. *The SS protocol securely sorts $\{\mathsf{E}(\vec{\mathtt{x}}_i) = (\mathsf{E}(\mathtt{x}_i^1), \mathsf{E}(\mathtt{x}_i^2), \cdots, \mathsf{E}(\mathtt{x}_i^d)) \mid \mathtt{x}_i^j \in \mathcal{M}, i \in [1, n], j \in [1, d]\}$ in the presence of semi-honest (non-colluding) adversaries $\{\mathcal{A}_1, \mathcal{A}_2\}$.*

Proof. Since $\mathcal{CS}_1$ always processes data records over their ciphertexts, $\mathcal{A}_1$ receives encrypted data in both views. Similar to $\mathcal{A}_1$'s views in the SBT subprotocol, the semantic security of SHE guarantees that the views of $\mathcal{A}_1$ in the real and the ideal executions are indistinguishable.

$\mathtt{Sim}_2$ runs $\mathcal{A}_2$ as follows. First, $\mathtt{Sim}_2$ randomly chooses $\{\vec{\mathtt{x}'}_i = (\mathtt{x}'^1_i, \mathtt{x}'^2_i, \cdots, \mathtt{x}'^d_i) \mid \mathtt{x}'^j_i \in \mathcal{M}, i \in [1, n], j \in [1, d]\}$, and calculates $\sigma_i' = \sum_{j=1}^{d} \mathtt{x}'^j_i, i \in [1, n]$. After choosing $n + 1$ random numbers $\{r_0, r_1, \cdots, r_n\}$ satisfying $r_0 > r_i, i \in [1, n]$, it computes $\{\hat{\sigma}_i' = r_0 \cdot \sigma_i' + r_i \mid i \in [1, n]\}$ and sends them to $\mathcal{A}_2$, which is the view of $\mathcal{A}_2$ in the ideal execution. In the real execution, $\mathcal{A}_2$ receives $\{\hat{\sigma}_i = r_0 \cdot \sigma_i + r_i \mid i \in [1, n]\}$, where $\sigma_i = \sum_{j=1}^{d} \mathtt{x}_i^j$. Since both $\mathtt{x}_i^j$ and $\mathtt{x}'^j_i \in \mathcal{M}$, and $n + 1$ random numbers are chosen to perturb σ_i and σ_i', the views of $\mathcal{A}_2$ in the real and the ideal executions are indistinguishable. □

The above proof shows that our SS protocol can preserve the privacy of plaintexts $\{\vec{\mathtt{x}}_i = (\mathtt{x}_i^1, \mathtt{x}_i^2, \cdots, \mathtt{x}_i^d) \mid \mathtt{x}_i^j \in \mathcal{M}, i \in [1, n], j \in [1, d]\}$ and results $\{\mathsf{E}(\hat{\delta}_i^\rho) \mid 1 \leq \rho \leq \lceil \log_2 n \rceil, i \in [1, n]\}$ (proved in $\mathcal{A}_1$'s views).

Next, we prove that the SDC and SHDC protocols are secure, i.e., they can protect inputs, results, and single-dimensional privacy.

Theorem 5. *The SDC and SHDC protocols securely determine dominance relations without leaking inputs, the protocol results, and the single-dimensional privacy.*

Proof. SDC: As shown in Fig. 7, the SDC protocol consists of the SBT and SEQ subprotocols. Recalling Section 5.1.2, before running SEQ and SBT subprotocols, the SDC protocol

generates $\{\theta^i \mid i \in [1,d]\}$, α, and β. Since all of these values are random, and these subprotocols are secure (Theorem 2 and Theorem 3), we can prove that the SDC protocol is secure according to Theorem 1. That is, the SDC protocol can protect the plaintext of data records and protocol result from leaking. Besides, since the SDC protocol uses the flip-coin mechanism to randomly adjust the order relation of two values, it can protect the single-dimensional privacy, i.e, the order or equality relation of each dimension's values. Thus, the SDC protocol is privacy-preserving.

SHDC: If our SHDC protocol determines the dominance relation of two data records by leaf nodes, it has the same security as the SDC protocol. On the other hand, if the dominance relation is determined by non-leaf nodes, $\mathcal{CS}_1$ and $\mathcal{CS}_2$ can know whether one data record does not dominate the other one. It is a trivial leakage, denoted as $\mathcal{L} = \{\mathrm{x}_{i_1} \not\succ \mathrm{x}_{i_2} \mid i_1 \neq i_2\}$, because $\mathcal{CS}_1$ and $\mathcal{CS}_2$ only know there exists one dimension $j \in [1,d]$ that $\mathrm{x}_{i_1}^j < \mathrm{x}_{i_2}^j$. Since neither $\mathcal{CS}_1$ nor $\mathcal{CS}_2$ knows which dimension has such the order relation, the single-dimensional privacy is protected in our SHDC protocol. Thus, our SHDC protocol is privacy-preserving. □

6.3 Privacy Preservation of The Proposed Secure Skyline Computation Protocol

In this subsection, we first prove that our secure skyline computation protocol can preserve the privacy of inputs, the protocol results, and the single-dimensional privacy. Then, we show that our secure skyline computation protocol can hide access patterns.

Theorem 6. *The secure skyline computation protocol securely computes skyline points without leaking inputs, the protocol results, and the single-dimensional privacy.*

Proof. From Algorithm 2, we can see that all of the intermediate results are random. According to Theorem 1, our secure skyline computation protocol can ensure the plaintext of data records and protocol results without leaking. For single-dimensional privacy, it only involves the SHDC protocol. Although there is a trivial leakage $\mathcal{L}$, we have proved that the single-dimensional privacy is preserved in our SHDC protocol. Therefore, our secure skyline computation protocol can preserve the single-dimensional privacy. □

Theorem 7. *The secure skyline computation protocol can hide the information about which data records in $\{\mathsf{E}(\vec{\mathrm{x}}_i) \mid i \in [1,n]\}$ are selected as skyline points.*

Proof. We prove Theorem 7 by showing neither $\mathcal{CS}_1$ nor $\mathcal{CS}_2$ knows which data records are selected as skyline points.

For $\mathcal{CS}_1$. From Algorithm 2, there are two cases for a data record to be added into the skyline set.

Case 1: A data record that is not dominated by any skyline point is a skyline point (line 22 in Algorithm 2). Since our SS protocol returns the encrypted sort index $\{\mathsf{E}(\delta_i^\rho)\}$, and the data record is computed from $\sum_{l=1}^{n} \mathsf{E}(\mathrm{x}_l^j) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil}(b_l^\rho \odot \mathsf{E}(\delta_i^\rho)), j \in [1,d], i \in [1,n]$, $\mathcal{CS}_1$ cannot link the computed data record to the data record in the dataset $\{\mathsf{E}(\vec{\mathrm{x}}_i) = (\mathsf{E}(\mathrm{x}_i^1), \mathsf{E}(\mathrm{x}_i^2), \cdots, \mathsf{E}(\mathrm{x}_i^d)) \mid i \in [1,n]\}$. Therefore, $\mathcal{CS}_1$ does not know which item is selected as skyline point in this case.

Case 2: A data record that has the maximum sum value is a skyline point (line 12 in Algorithm 2). Similarly, the data point is calculated by $\sum_{l=1}^{n} \mathsf{E}(\mathrm{x}_l^j) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil}(b_l^\rho \odot \mathsf{E}(\delta_1^\rho)), j \in [1,d]$ (line 11). Consequently, $\mathcal{CS}_1$ does not know which item is selected as skyline point in this case.

For $\mathcal{CS}_2$. From Algorithm 2, we can see that $\mathcal{CS}_2$ can get information when running the SS and SHDC protocols.

SS: $\mathcal{CS}_2$ obtains the perturbed sum value of each data record. Besides, before sending encrypted sum values to $\mathcal{CS}_2$, $\mathcal{CS}_1$ randomly permutes them with the XOR gate. Therefore, $\mathcal{CS}_2$ cannot link these sum values to the corresponding data records.

SHDC: For the non-leaf nodes, $\mathcal{CS}_2$ can learn the information about *one data record does not dominate the other one* by observing whether $\theta = r_1 \cdot (\sigma_2^l - \sigma_1^l) \cdot (\sigma_2^r - \sigma_1^r) + r_2 > 0$, where σ_i^l and σ_i^r $(i = 1, 2)$ are partial sum values (see details in Section 5.1.3). However, $\mathcal{CS}_2$ cannot link the partial sum values to the corresponding data record due to the perturbation and permutation. When reaching leaf nodes, our SHDC protocol will invoke the SDC protocol. However, Theorem 5 shows that $\mathcal{CS}_2$ learns nothing from the SDC protocol. Therefore, $\mathcal{CS}_2$ does not know which items are selected as skyline points. □

7 Performance Evaluation

In this section, we evaluate the performance of our proposed scheme. From Section 5.3, we know that the data reporting and data recovery phases only involve data encryption and noise removal, respectively, which are efficient and intuitive in terms of performance. Therefore, in this section, we focus on the performance of the interval skyline search phase.

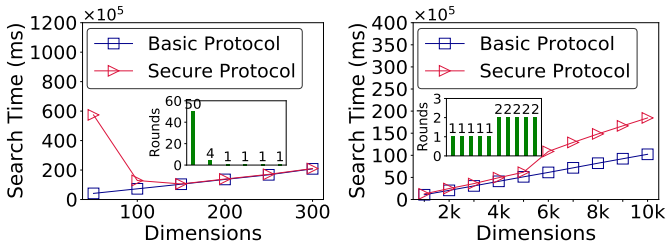
Experimental setting: We implemented our scheme in Java and executed it on a machine with 16 GB memory, 3.4 GHz Intel(R) Core(TM) i7-3770 processors, and Ubuntu 16.04 OS. In our experiments, we adopt two real-world time series datasets and illustrate the detailed information in Table 2. For simplicity sake, we denote these two datasets as **Gas** and **Electricity**, respectively. It is worth noting there are around 100,000 time stamps in the **Electricity** dataset. We filter out the missing values (the value of two consecutive time stamps is 0), and the effective timestamps are 10,303.

TABLE 2
Real-world datasets used in our experiments

Name	Time series (n)	Timestamps (d)
Greenhouse Gas Observing Network Data Set [33]	2912	327
Electricity Load Diagrams Data Set [34]	315	10303

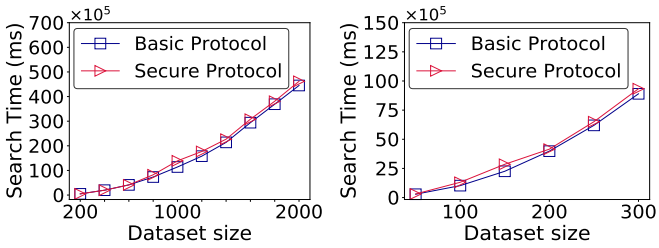
To ensure privacy, we set $\tau = 4$ in our SHDC protocol, i.e., the number of dimensions of non-leaf nodes should be larger than 4. For security parameters of SHE, we let $k_0 = 4096, k_1 = 40, k_2 = 160$ in our evaluations. The reasons for choosing these values as security parameters are i) k_1 is related to the space of plaintexts. In our experiments, since the space (2^{20}) can fully cover the plaintext value, we let $k_1 = 2 \times 20 = 40$ bits; ii) k_2 indicates the size of the secret key. We set $k_2 = 160$ bits to guarantee the security of the secret key; iii) for k_0, it is related to the ciphertext

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3153759, IEEE Transactions on Dependable and Secure Computing



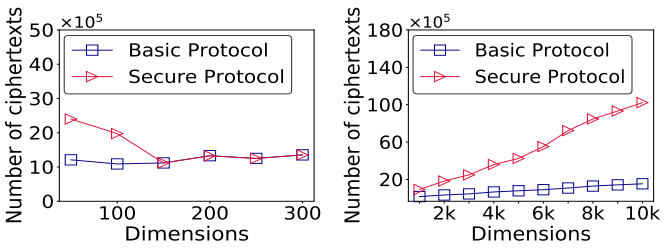
(a) **Gas** Dataset (b) **Electricity** Dataset

Fig. 8. Computational costs varying with d.



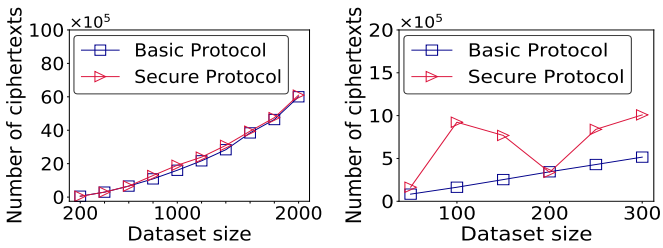
(a) **Gas** Dataset (b) **Electricity** Dataset

Fig. 9. Computational costs varying with n.



(a) **Gas** Dataset (b) **Electricity** Dataset

Fig. 10. Communication costs varying with d.



(a) **Gas** Dataset (b) **Electricity** Dataset

Fig. 11. Communication costs varying with n.

size. We set $k_0 = 4096$ bits to support more homomorphic multiplication-I operations. Please see Section 3.2 for detailed information about the plaintext space, secret key size, and ciphertext size.

7.1 Performance of Our Proposed Scheme

In this subsection, we evaluate the computational cost and the communication cost of the basic and secure skyline computation protocols by varying the number of time series n and timestamps (dimensions) d on **Gas** and **Electricity** datasets. Here, we respectively denote these two protocols as basic protocol and secure protocol in the following.

• *Computational cost of searching interval skyline.* Fig. 8 depicts the search time varying with d, in which Fig. 8(a) shows the evaluation over **Gas** dataset and $n = 1000$, while Fig. 8(b) shows the evaluation over **Electricity** dataset and $n = 100$. In Fig. 8(a), the search time of the secure protocol is more than the basic protocol when d is small. It is reasonable since, in the **Gas** dataset, the number of skyline points is not changed when we vary d. When d is small, it means that $\mathcal{CS}_1$ has a high probability of obtaining the dominance relations from the leaf nodes, leading to more rounds to get the whole skyline points. However, when d is large, $\mathcal{CS}_1$ may get the dominance relation from non-leaf nodes, leading to fewer rounds. When the number of rounds is 1, the computational cost of the secure protocol is equivalent to using the SFS algorithm and has a similar search time to the basic protocol. The trend of Fig. 8(b) is also related to the number of rounds. To clearly show the reason, we plot the number of rounds in the corresponding reduced figure.

Fig. 9 plots the search time varying with n, in which Fig. 9(a) shows the evaluation over **Gas** dataset and $d = 100$, while Fig. 9(b) shows the evaluation over **Electricity** dataset and $d = 1000$. In both figures, since the secure protocol computes the skyline points with one round in the most cases, it means that dominance relations are determined by the non-leaf nodes of our SHDC protocol. Therefore, our secure protocol has a similar search time as the basic protocol.

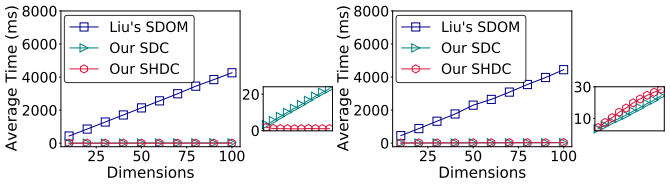
• *Communication cost of searching interval skyline.* Fig. 10 and Fig. 11 depict the communication cost of the secure and basic protocols, in which we vary n and d, respectively. For simplicity sake, we measure the communication cost by the number of ciphertexts sent between two servers: $\mathcal{CS}_1$ and $\mathcal{CS}_2$. This is reasonable since counting the number of ciphertexts is equivalent to counting the number of bytes that can be easily calculated with: *the number of bytes = the number of ciphertexts* $\times 2k_0/8$, where $k_0 = 4096$. Besides, since the size of plaintexts is significantly smaller than that of ciphertexts, we do not consider the communication cost of plaintexts in our SS protocol.

Fig. 10(a) and Fig. 10(b) show the similar trends to Fig. 8(a) and Fig. 8(b), respectively. That is because the communication cost is also related to the number of rounds. However, Fig. 11 shows a different trend. In Fig. 11(a), since the numbers of skyline points and rounds are not changed when n is increasing over the **Gas** dataset, the communication cost of the secure and basic protocol is close. In Fig. 11(b), the secure protocol has more communication cost than the basic protocol. That is because the number of skyline points k would affect the communication costs if k is relatively small compared to the whole dataset. In our protocols, each skyline point is used to check the dominance relation with candidate points. Therefore, the more skyline points will result in the more communication costs. In the experiment of Fig. 11(b), the number of skyline points is $\{2, 6, 4, 1, 2, 2\}$ for the dataset size from 50 to 300. Thus, the trend in Fig. 11(b) is reasonable. Note that, when n is 200, the number of skyline points is 1. In this case, the communication cost of the secure protocol should be similar to that of basic protocol (See Algorithms 1 and 2). Fig. 11(b) demonstrates the correctness of our analysis.

7.2 Comparing Dominance Check Protocols

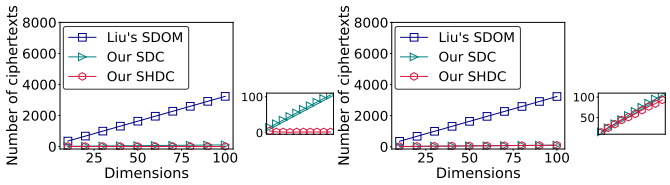
In the process of skyline search, the core component is the secure dominance check protocol that can determine whether two encrypted data records have a dominance relation. In this paper, we introduce a secure dominance check

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3153759, IEEE Transactions on Dependable and Secure Computing



(a) **Gas** Dataset (b) **Electricity** Dataset

Fig. 12. Computational costs varying with d.



(a) **Gas** Dataset (b) **Electricity** Dataset

Fig. 13. Communication costs varying with d.

(SDC) protocol and propose a secure high-dimensional dominance check (SHDC) protocol to determine the dominance relation. Both of them can preserve the privacy of plaintexts and single-dimensional privacy. In [12], Liu et al. also proposed a secure dominance check protocol, hereafter we denote it as Liu's SDOM, which can be used in our scheme. As a result, in this subsection, we will compare the SDC, SHDC, and Liu's SDOM protocols in terms of computational and communication costs.

However, Liu's SDOM protocol employs the Paillier encryption, which is more expensive than the SHE scheme used in our protocol. Thus, to be fair, we implemented Liu's SDOM protocol with SHE and compare these protocols based on the same cryptographic primitive. It is worth noting that: i) by selecting suitable security parameters, SHE and Paillier can have the same security strength; ii) all of these three protocols (SDC, SHDC, and Liu's SDOM) can be achieved with Paillier or SHE; iii) the SDC and SHDC protocols have the same security level as Liu's SDOM if the same underlying encryption scheme is adopted. In addition, although the scheme in [35] is more efficient than that of in [12], it is still reasonable to compare the SDC and SHDC protocols with Liu's SDOM protocol in [12]. This is because the work in [35] uses the same secure protocols as [12] and applies the parallelization technology at the dataset level. Therefore, evaluating the performance of [12] is equivalent to evaluating the performance of [35] if other comparison works can also use the designed parallelization technology.

• *Comparing computational costs.* From Section 5.1.2, 5.1.3, and [12], we know that the computational cost of these three protocols is only related to the number of dimensions. Accordingly, Fig. 12 depicts average execution costs of the SDC, our SHDC, and Liu's SDOM varying with the number of dimensions from 10 to 100. Fig. 12(a) and Fig. 12(b) show the evaluations over **Gas** and **Electricity** datasets, respectively. Both figures show that our protocol can improve the efficiency of determining dominance relation by at least two orders of magnitude compared to Liu's SDOM protocol. That is because: i) the secure comparison subprotocol (SBT) used in our protocol is more efficient than that of Liu's SDOM [36]; ii) our designed algorithm is more efficient. That is, we determine whether $\forall j \in [1, d]$ $\mathbf{x}_1^j \geq \mathbf{x}_2^j$ by testing whether $\alpha = \beta$ (see details in Section 5.1.2), while Liu's SDOM protocol checks all dimensions one by one.

It is interesting that the computational cost of our SHDC protocol is much less than the SDC protocol in Fig. 12(a), whereas it is slightly more computationally expensive in Fig. 12(b). This is reasonable since most of the data records in the **Gas** dataset have no dominance relation with each other and thus make our SHDC protocol determine the dominance relation by the non-leaf nodes of DC-tree instead of leaf nodes. It can greatly improve performance and, it is our intention to design such a high-dimensional protocol. While for the **Electricity** dataset, there exist several data records that can dominate others. As a result, our SHDC protocol determines dominance relations by leaf nodes of DC-tree, i.e., invoking SDC protocol, in most cases. Therefore, the computational cost of SHDC protocol is approximately equivalent to *time cost of checking non-leaf nodes* + *time cost of SDC protocol*. Thus, for the **Electricity** dataset, our SHDC protocol takes more time than the SDC protocol.

• *Comparing communication costs.* Fig. 13 plots the number of transferred ciphertexts of these three protocols varying with the number of dimensions, in which Fig. 13(a) shows the communication cost over the **Gas** dataset, while Fig. 13(b) is over the **Electricity** dataset. Both figures show that our protocol entails at least 23× improvement in the communication cost compared to Liu's SDOM protocol. The reason is that it needs $d + 5$ ciphertexts to achieve the SDC protocol, while it is $2(d+1)(l+1)$ for Liu's SDOM protocol, where d is the number of dimensions and l is the largest bit length of values in the evaluated dataset. Meanwhile, we can see that our SHDC protocol is always better than the SDC protocol in both datasets. For the **Gas** dataset, since checking dominance relation in non-leaf nodes only needs to send one ciphertext to $\mathcal{CS}_2$, our SHDC protocol is significantly more efficient in the communication cost when determining dominance relations by non-leaf nodes. For the **Electricity** dataset, although the dominance relation is determined by leaf nodes, and additional communication costs, i.e., the transmitted ciphertexts of non-leaf nodes, are incurred, the average transmitted ciphertexts of our SHDC protocol are slightly less than that of the SDC protocol. It is because the benefits of determining dominance relations by non-leaf nodes outweigh the incurred costs. This trend will be more obvious when the number of dimensions is large, as shown in the reduced figure in Fig. 13(b).

8 Related Work

Privacy-preserving skyline queries have attracted considerable attention in the database community [11]–[14], [37], [38]. In 2013, Bothe et al. [11] mapped the problem of computing skyline into determining the non-descending series that was computed with the scalar products among sub-vectors of tuples. However, since the simple matrix encryption was adopted to encrypt the sub-vectors of tuples, this scheme is not semantically secure, and an adversary can launch a known plaintext attack to infer the secret keys. Zaman et al. [37] proposed a secure skyline computation scheme in MapReduce. It aimed to support the multi-party secure computation and assumed a trusted party: Coordinator, who can obtain the order of data on each dimension. Since the Coordinator must know the order relations of each dimension to compute skyline, the skyline

computation approach in [37] cannot be applied to our scheme due to the security considerations. In 2017, Liu et al. [12] proposed a fully secure skyline computation scheme for dynamic skyline queries. This scheme can protect the plaintexts, single-dimensional privacy, and access patterns from leaking. However, it suffers from the performance issues due to the inefficient basic protocols. In [35], Liu et al. further presented a parallel version of [12] by designing and applying a parallelization technology. The work in [35] uses the same secure protocols as [12] and applies the parallelization technology to improve performance. Recently, Wang et al. [14] also designed a secure scheme to deal with dynamic skyline queries. This scheme adopted the order-revealing encryption (ORE) as the cryptographic primitive, leading to the information leakage in single-dimensional privacy. The work in [38] proposed a secure skyline computation scheme for user-defined skyline queries. It also has the problem in performance due to the expensive encryption. Besides, it extended a d-dimensional data record to $2d$ dimensions for skyline computation, which is hard to be applied to high-dimensional data. In 2019, Zheng et al. [13] designed a skyline computation protocol by determining dominance relations over encrypted data. However, their work did not consider the single-dimensional privacy and access patterns.

Although some works also considered secure skyline queries, they focused on other techniques and cannot be used in our scenario. In 2016, Chen et al. [39] devised three secure skyline query schemes. However, these schemes are for the location databases and aim to verification instead of privacy preservation. In [40], Wang et al. proposed a hardware-aided secure skyline query scheme, in which some "hard-to-compute" operations are shifted to SGX. Recently, Zeighami et al. [41] proposed an approach to use query result materialization for answering dynamic skyline queries on encrypted data, which focuses on the skyline result materialization and is not in line with our scenario.

9 Conclusion

In this paper, we have proposed an efficient and privacy-preserving interval skyline query scheme over encrypted time series data, which can preserve the privacy of plaintexts, single-dimensional privacy, and access patterns while ensuring efficiency. Specifically, we first specified the interval skyline query and introduced the symmetric homomorphic encryption (SHE). Then, we presented SBT and SEQ subprotocols and privacy-preserving logic gates. Based on these building blocks, we designed a secure sort (SS) protocol to sort the encrypted dataset. Further, to deal with high-dimensional time series data, we designed a DC-tree and proposed our SHDC protocol. With these protocols, we proposed our secure skyline computation protocol. Finally, we analyzed the security of our scheme and conducted extensive experiments to evaluate it, and the results illustrate that our scheme is efficient in both computation and communication.

Acknowledgements

This research was supported in part by NSERC Discovery Grants (04009), ZJNSF (LZ18F020003), NSFC (U1709217).

References

[1] P. Wang, H. Wang, and W. Wang, "Finding semantics in time series," in *SIGMOD*, 2011, pp. 385–396.
[2] Y. Sakurai, Y. Matsubara, and C. Faloutsos, "Mining and forecasting of big time-series data," in *SIGMOD*, 2015, pp. 919–922.
[3] L. Zhang, N. Alghamdi, M. Y. Eltabakh, and E. A. Rundensteiner, "Tardis: Distributed indexing framework for big time series data," in *ICDE*. IEEE, 2019, pp. 1202–1213.
[4] C. Sheng, W. Hsu, and M. L. Lee, "Mining dense periodic patterns in time series data," in *ICDE*. IEEE, 2006, pp. 115–115.
[5] N. Alghamdi, L. Zhang, H. Zhang, E. A. Rundensteiner, and M. Y. Eltabakh, "Chainlink: indexing big time series data for long subsequence matching," in *ICDE*. IEEE, 2020, pp. 529–540.
[6] B. Jiang and J. Pei, "Online interval skyline queries on time series," in *ICDE*. IEEE, 2009, pp. 1036–1047.
[7] H. Wang, C.-K. Wang, Y.-J. Xu, and Y.-C. Ning, "Dominant skyline query processing over multiple time series," *Journal of Computer Science and Technology*, vol. 28, no. 4, pp. 625–635, 2013.
[8] G. He, L. Chen, C. Zeng, Q. Zheng, and G. Zhou, "Probabilistic skyline queries on uncertain time series," *Neurocomputing*, vol. 191, pp. 224–237, 2016.
[9] E. Shi, T. H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *NDSS*, vol. 2. Citeseer, 2011, pp. 1–17.
[10] M. Harvan, S. Kimoto, T. Locher, Y.-A. Pignolet, and J. Schneider, "Processing encrypted and compressed time series data," in *ICDCS*. IEEE, 2017, pp. 1053–1062.
[11] S. Bothe, P. Karras, and A. Vlachou, "eskyline: Processing skyline queries over encrypted data," *VLDB*, vol. 6, no. 12, pp. 1338–1341, 2013.
[12] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure skyline queries on cloud platform," in *ICDE*. IEEE, 2017, pp. 633–644.
[13] Y. Zheng, R. Lu, B. Li, J. Shao, H. Yang, and K.-K. R. Choo, "Efficient privacy-preserving data merging and skyline computation over multi-source encrypted data," *Information Sciences*, vol. 498, pp. 91–105, 2019.
[14] W. Wang, H. Li, Y. Peng, S. S. Bhowmick, P. Chen, X. Chen, and J. Cui, "Scale: An efficient framework for secure dynamic skyline query processing in the cloud," in *International Conference on Database Systems for Advanced Applications*. Springer, 2020, pp. 288–305.
[15] B. Wang, Y. Hou, M. Li, H. Wang, and H. Li, "Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*, 2014, pp. 111–122.
[16] G. Chen, T.-H. Lai, M. K. Reiter, and Y. Zhang, "Differentially private access patterns for searchable symmetric encryption," in *INFOCOM*. IEEE, 2018, pp. 810–818.
[17] J. Ning, G. S. Poh, X. Huang, R. Deng, S. Cao, and E.-C. Chang, "Update recovery attacks on encrypted database within two updates using range queries leakage," *IEEE Transactions on Dependable and Secure Computing*, 2020.
[18] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *SIGSAC*, 2015, pp. 644–655.
[19] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: ramification, attack and mitigation." in *NDSS*. Citeseer, 2012.
[20] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity range query over encrypted time series data," *IEEE Transactions on Dependable and Secure Computing*, 2021.
[21] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *ICDE*, vol. 3, 2003, pp. 717–719.
[22] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data." in *NDSS*, 2015, p. 4325.
[23] J. Brickell and V. Shmatikov, "Privacy-preserving graph algorithms in the semi-honest model," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2005, pp. 236–252.
[24] H. Yu, X. Jia, H. Zhang, X. Yu, and J. Shu, "Psride: Privacy-preserving shared ride matching for online ride hailing systems," *IEEE Transactions on Dependable and Secure Computing*, 2019.
[25] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, "Privacy-preserving matrix factorization," in *Proceedings*

of the 2013 ACM SIGSAC conference on Computer & communications security, 2013, pp. 801–812.

[26] Y. Zheng, H. Duan, and C. Wang, "Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2475–2489, 2018.

[27] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. Deng, "Privacy-preserving federated deep learning with irregular users," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[28] H. Mahdikhani, R. Lu, Y. Zheng, J. Shao, and A. A. Ghorbani, "Achieving o $(\log^3 n)$ communication-efficient privacy-preserving range query in fog-based iot," *IEEE Internet of Things Journal*, pp. 5220–5232, 2020.

[29] Y. Guan, R. Lu, Y. Zheng, S. Zhang, J. Shao, and G. Wei, "Toward privacy-preserving cybertwin-based spatio-temporal keyword query for its in 6g era," *IEEE Internet of Things Journal*, 2021.

[30] S. Zhang, S. Ray, R. Lu, Y. Zheng, Y. Guan, and J. Shao, "Achieving efficient and privacy-preserving dynamic skyline query in online medical diagnosis," *IEEE Internet of Things Journal*, 2021.

[31] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[32] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multi-party computation." *IACR Cryptol. Eprint Arch.*, vol. 2011, p. 272, 2011.

[33] "Greenhouse gas observing network data set," https://archive.ics.uci.edu/ml/datasets/Greenhouse+Gas+Observing+Network, 2015.

[34] "Electricity load diagrams data set," https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014, 2015.

[35] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure and efficient skyline queries on encrypted data," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 7, pp. 1397–1411, 2018.

[36] T. Veugen, F. Blom, S. J. de Hoogh, and Z. Erkin, "Secure comparison protocols in the semi-honest model," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 7, pp. 1217–1228, 2015.

[37] A. Zaman, M. A. Siddique, and Y. Morimoto, "Secure computation of skyline query in mapreduce," in *International Conference on Advanced Data Mining and Applications*. Springer, 2016, pp. 345–360.

[38] 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 *TrustCom/BigDataSE*. IEEE, 2018, pp. 958–963.

[39] W. Chen, M. Liu, R. Zhang, Y. Zhang, and S. Liu, "Secure outsourced skyline query processing via untrusted cloud service providers," in *INFOCOM*. IEEE, 2016, pp. 1–9.

[40] J. Wang, M. Du, and S. S. Chow, "Stargazing in the dark: Secure skyline queries with sgx," in *International Conference on Database Systems for Advanced Applications*. Springer, 2020, pp. 322–338.

[41] S. Zeighami, G. Ghinita, and C. Shahabi, "Secure dynamic skyline queries using result materialization," in *ICDE*. IEEE, 2021, pp. 157–168.

Songnian Zhang received his M.S. degree from Xidian University, China, in 2016 and he is currently pursuing his Ph.D. degree in the Faculty of Computer Science, University of New Brunswick, Canada. His research interest includes cloud computing security, big data query and query privacy.

Suprio Ray is an Associate Professor with the Faculty of Computer Science, University of New Brunswick, Fredericton, Canada. He received a Ph.D. degree from the Department of Computer Science, University of Toronto, Canada, in 2015. His research interests include big data and database management systems, run-time systems for scalable data science, provenance and privacy issues in big data and query processing on modern hardware. E-mail: sray@unb.ca

Rongxing Lu (S'09-M'11-SM'15-F'21) is currently an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. He is a Fellow of IEEE. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise, and was the recipient of 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Chair of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee), and the founding Co-chair of IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC).

Yandong Zheng received her M.S. degree from the Department of Computer Science, Beihang University, China, in 2017 and she is currently pursuing her Ph.D. degree in the Faculty of Computer Science, University of New Brunswick, Canada. Her research interest includes cloud computing security, big data privacy and applied privacy.

Yunguo Guan is a PhD student of the Faculty of Computer Science, University of New Brunswick, Canada. His research interests include applied cryptography and game theory.

Jun Shao received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2008. He was a Post-Doctoral Fellow with the School of Information Sciences and Technology, Pennsylvania State University, Pennsylvania, PA, USA, from 2008 to 2010. He is currently a Professor with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His current research interests include network security and applied cryptography.