Privacy-Preserving and Resilient Skyline Querying in Sharded Blockchain Networks for Web 3.0

No Author Given

No Institute Given

Abstract. In recent years, Web 3.0 and decentralized applications (DApps) have gained significant traction, driving the demand for robust querying systems that can effectively manage complex data interactions. Skyline querying, which identifies optimal data points based on multiple criteria, is crucial in this context. However, existing skyline query systems fail to address the challenges posed by missing data dimensions in distributed environments, severely compromising query accuracy. To this end, this paper introduces a novel querying and security model tailored for Web 3.0, in which we first develop a comprehensive framework for skyline queries in this context. Then, we propose a dimensional imputation and enhancement network (DIME) based on autoencoders to address missing dimension (MD) problem. Next, we define the encrypted shard-tree (ES-Tree) index along with the secure shard-based dominance protocol (SSD), culminating in the development of the blockchain-based skyline querying algorithm (SBSQ) and the skyline merging algorithm of sharded blockchain (SMSB) for multi-sharded environments. Experimental results demonstrate the effectiveness of our approaches in enhancing query accuracy and resilience in sharded blockchain networks.

Keywords: Web 3.0 · DApps · blockchain · skyline query · missing dimension.

1 Introduction

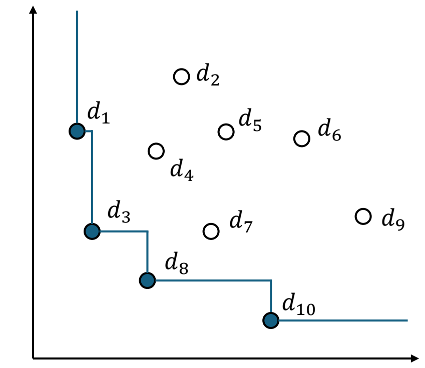
With the rapid advancement of Web 3.0 [1], decentralized technologies have emerged as transformative forces in data storage and management. As a core pillar of Web 3.0, blockchain technology has reshaped data interaction mechanisms while introducing new opportunities in privacy protection and data security, catalyzing the widespread adoption of decentralized applications (DApps) [2,3,4]. However, traditional centralized cloud computing architectures, which rely on singular computational resources and centralized storage, face pronounced challenges in handling the explosion of applications and data [5]. These limitations include significant performance bottlenecks and increased privacy risks, making it difficult to meet user demands for data efficiency and security.

In contrast, Web 3.0 distributes computational and storage tasks across globally dispersed nodes, thereby mitigating single-point failures and performance bottlenecks. In this context, DApps have found extensive applications across finance, social media, and content platforms. For instance, decentralized finance (DeFi) platforms leverage smart contracts and distributed nodes to facilitate global financial transactions [6]. However, the decentralized nature of DApps introduces new challenges, particularly concerning missing dimensions (MD) in data, which arise due to the inherent complexity and wide distribution of network nodes [7,8,9]. This issue is prevalent across Web 3.0 and DApp use cases, where data faults, especially during network fluctuations or high transaction volumes, can result in incomplete information. Such disruptions in DeFi applications, for example, may prevent users from accessing real-time asset or transaction data, thereby compromising query accuracy and informed decision-making.

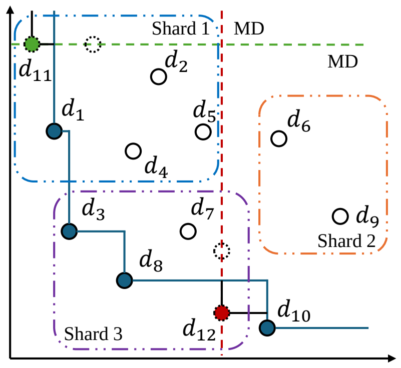
To enhance the scalability of Web 3.0 and DApp infrastructures, sharded blockchain architectures have been widely adopted. By partitioning the network into multiple shards that can process transactions in parallel, sharded blockchain significantly increases system throughput, meeting the high-demand requirements of large-scale data querying [10]. However, the sharded structure adds complexity to data synchronization and inter-shard communication, exacerbating MD issues. It is noteworthy that, although Decentralized Skyline Querying (DSQ) demonstrates significant efficiency in multidimensional data filtering by enabling users to swiftly identify optimal solutions that meet specific preferences, DSQ has yet to become a foundational technology in Web 3.0 and DApps [11]. This is largely due to the substantial technical challenges that arise when implementing DSQ within a sharded blockchain environment. Firstly, MD disrupt skyline query performance by limiting the ability to accurately compare data points across all relevant attributes, often resulting in incomplete dominance relationships. Additionally, the architecture of sharded blockchains necessitates parallel processing and integration of multidimensional data across multiple shards, which introduces complexities for DSQ in maintaining data integrity and consistency during cross-shard aggregation and multidimensional data handling. Existing research primarily focuses on privacy protection and efficiency improvements in blockchains, with limited studies addressing the applicability of DSQ in sharded environments and the challenges of MD repair.

Therefore, this study proposes an innovative approach to integrate sharded blockchain and DSQ by designing an MD repair mechanism to ensure the accuracy and reliability of decentralized queries, thereby enhancing query robustness and user experience in DApps. The contribution of this paper includes:

- We introduce a novel querying and security model tailored for Web 3.0, beginning with the development of a comprehensive framework for skyline queries specifically designed for this context.
- We address the issue of MD by proposing a recovery mechanism named the dimensional imputation and enhancement network (DIME) based on autoencoders, which enhances the reliability of skyline queries by effectively data reconstruction.
- We propose the encrypted shard-tree (ES-Tree) index along with the secure shard-based dominance protocol (SSD), culminating in the development of



(a) Traditional skyline query over certain data



(b) Skyline query over MD data in sharing blockchains

Fig. 1. Comparison of traditional skyline queries and skyline queries with MD.

the blockchain-based skyline querying algorithm (SBSQ) and merging algorithm of sharded blockchain (SMSB) for multi-sharded environments.
- Experimental results demonstrate the effectiveness of these contributions in enhancing query accuracy and resilience in sharded blockchain networks.

The remaining contents are conducted as follows. Section 2 introduces the total framework of the proposed model. Section 3 presents the dimensional imputation and enhancement network. The shard-tree index and query protocol are proposed in Section 4 and 5. Experimental results are demonstrated in Section 6. Section 7 expresses the main research works. Section 8 concludes the paper.

2 Proposed Model

2.1 System Model

Under the Web 3.0 ecosystem, we propose a secure skyline querying architecture using a sharded blockchain network with m shards, as illustrated in the figure. Each entity's function is described as follows:

1. **Subscriber:** The Data owner (DO) generates a key pair $\langle pk, sk\rangle$ for the Paillier encryption scheme and a key SK for the ORE encryption scheme. It partitions the plaintext dataset $\mathbf{X}$ by region and constructs a secure index $I = \{I_1, \ldots, I_i, \ldots, I_m\}$ for the data points in each shard using both Paillier and ORE encryption algorithms. Here, I_i represents the secure index for the data points covered by shard i. The DO uploads I to the distributed storage in the blockchain network and securely transmits the public key pk to the data store, while sharing pk and SK with the DApps (decentralized applications) and $\langle pk, sk\rangle$ with the Ethereum Virtual Machine (EVM).

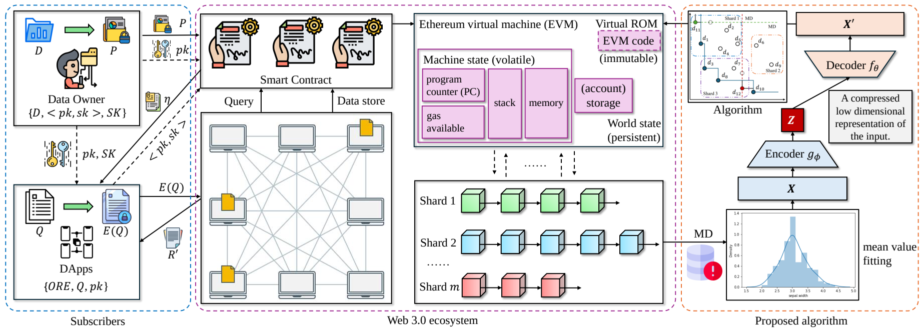


Fig. 2. The framework of the proposed system model.

2. **DApp:** As an authorized subscriber, the DApp submits a query $E(Q)$ encrypted using the Paillier encryption scheme to the network. After receiving the response R' from the EVM and the auxiliary result η, the DApp calculates the final skyline result locally.
3. **Shard:** Each shard i performs a secure skyline query on the secure index I_i within its region with assistance from the EVM for enhanced security. For multi-region queries that involve data from multiple shards, the shard collaborates with the EVM, returning R' and η to the DApp.
4. **Distributed Storage:** The distributed storage within the Web 3.0 ecosystem holds the secure index I of the dataset $\mathbf{X}$ and distributes it across corresponding shards. It also assists shards in handling multi-region queries and returns encrypted results to the shards, maintaining data privacy and security.

2.2 Security Model

This paper assumes that the EVM and shards operate in a "semi-honest" manner, meaning they will follow the protocol to execute designated operations honestly, while also attempting to gain access to the original dataset and the user's query. However, they will not alter or tamper with the data or results maliciously. Therefore, based on the secure simulation model [12], this scheme protects the following three aspects of privacy: (1) Data Privacy: Neither the EVM nor any shard can gain any information about the original dataset $\mathbf{X}$. (2). Query Privacy and Unlinkability: Any information related to the query Q remains hidden from all parties involved. Additionally, the query-processing entities (EVM or shards) cannot track the query's access path to gather additional information, thus ensuring the query's unlinkability. (3)Result Privacy: Only the corresponding DApp can access the query result R. In other words, the security requirement of this protocol mandates that the simulated view $\text{Sim}^A(\cdot)$ is computationally indistinguishable from the real execution view $\text{Real}^A(\cdot)$. Definitions are as follows:

Definition 1. Given a security parameter $\epsilon \in \mathbb{N}$, and assuming that all polynomial-time adversaries A only have a negligible probability of success, the proposed scheme is considered secure if there exists an efficient simulator S such that for any encrypted query q and encrypted dataset $\mathcal{P}$, the following holds:

$$\left|\Pr\left[\mathrm{Real}^{A}(\epsilon, \mathcal{P}, q)\right] - \Pr\left[\mathrm{Sim}^{A}(\epsilon, \mathcal{P}, q)\right]\right| \leq \mathrm{negl}(\epsilon) \tag{1}$$

Problem Definition. Given a dataset $\mathbf{X} = \{P_1, \ldots, P_n\}$, where each data point $P_i \in \mathbf{X}$ $(1 \leq i \leq n)$ consists of shared attributes across multiple shards (represented as $P_i[1], P_i[2], \ldots, P_i[d]$).

Definition 2 (Shard-based Dominance). Given two arbitrary data points P_1 and P_2 in d-dimensional space, P_1 dominates P_2 (or P_2 is dominated by P_1), denoted $P_1 \prec P_2$, if and only if $\forall\, i \in \{1, \ldots, d\}$, $P_1[i] \leq P_2[i]$ and there exists at least one j such that $P_1[j] < P_2[j]$.

Definition 3 (Shard-Dominance Relative to Query). Given a shared query Q and two arbitrary data points P_1 and P_2, P_1 is said to dominate P_2 with respect to Q, denoted $P_1 \prec_Q P_2$, if (1) $P_1 \prec P_2$ and (2) P_1 is closer in terms of shared attribute similarity to Q than P_2.

Definition 4 (Secure Sharing-based Skyline Query). Given a secure index I generated from the dataset $\mathbf{X}$ and an encrypted, shared query $E(Q)$, the skyline result R regarding $E(Q)$ satisfies:

$$\forall P_i, P_j \in R\ (i \neq j),\ E(P_i) \not\prec_Q E(P_j),$$

and for $\forall P_k \in \mathbf{X} \setminus R$, there exists $P_r \in R$ such that $E(P_r) \prec_Q E(P_k)$. Furthermore, the security of the sharing-based skyline query satisfies Definition 1. The set of skyline results is denoted as $\mathrm{SKY}(I)$ or $\mathrm{SKY}(\mathbf{X})$.

3 Dimensional Imputation and Enhancement Network

To address the issue of MD in multi-dimensional datasets, we propose a dimensional imputation and enhancement network (DIME) to model underlying data patterns and reconstruct missing dimensions with minimal error, thereby supporting robust skyline operations across all dimensions. The process flow of DIME is as follows:

Let $\mathbf{X} = \{\mathbf{x}_i \in \mathbb{R}^d \mid i = 1, \ldots, N\}$ be a dataset with N samples and d dimensions, where missing values are represented by a binary mask matrix $\mathbf{M} \in \{0,1\}^{N \times d}$. Specifically, $\mathbf{M}_{ij} = 1$ if dimension j of sample i is missing. We denote the dataset with missing values as $\mathbf{X}_{\mathrm{mi}}$.

For each missing element $\mathbf{x}_{ij}$, mean imputation is applied, yielding an initial approximation, where the imputed dataset is denoted by $\mathbf{X}_{\mathrm{im}}$, and the normalized dataset is denoted $\mathbf{X}_{\mathrm{nor}}$:

$$\mathbf{x}_{ij} = \frac{1}{|\{k \mid \mathbf{M}_{kj} = 0\}|} \sum_{k=1}^{N} \mathbf{x}_{kj} \cdot (1 - \mathbf{M}_{kj}) \tag{2}$$

The imputed data $\mathbf{X}_{\text{im}}$ is normalized to the range $[0, 1]$ to ensure consistent feature scaling:

$$\mathbf{x}_{ij} = \frac{\mathbf{x}_{ij} - \min(\mathbf{x}_j)}{\max(\mathbf{x}_j) - \min(\mathbf{x}_j)} \tag{3}$$

Then, we employ an autoencoder model $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$ to reconstruct the full dimensionality of $\mathbf{X}_{\text{nor}}$. The autoencoder consists of an encoder-decoder structure, $f_\theta(\mathbf{x}) = h_\theta(g_\theta(\mathbf{x}))$, where:

- **Encoder**: $g_\theta : \mathbb{R}^d \to \mathbb{R}^k$, mapping input data to a latent representation in a reduced dimension k.
- **Decoder**: $h_\theta : \mathbb{R}^k \to \mathbb{R}^d$, reconstructing the original input from the latent space.

The model is trained by minimizing the reconstruction loss on available data:

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - f_\theta(\mathbf{x}_i)\|^2 \tag{4}$$

Here, the Mean Squared Error (MSE) serves as the loss function, and optimization is conducted using the Adam optimizer. Following training, the network reconstructs the missing entries by applying the autoencoder to the normalized dataset. For each missing element $\mathbf{x}_{ij}$ where $\mathbf{M}_{ij} = 1$, the predicted value is:

$$\mathbf{x}_{ij} = [f_\theta(\mathbf{x}_i)]_j \tag{5}$$

The reconstructed dataset $\mathbf{X}_{\text{com}}$ now contains imputed values across all dimensions, ready for skyline querying. DIME leverages latent representations to enhance imputation fidelity, thus offering robust, accurate dimensional recovery crucial for skyline operations in distributed and decentralized data environments.

4 Encrypted Shard-Tree Index and Protocol

4.1 Tree Index

To ensure the unlinkability of secure queries, this paper proposes an encrypted index structure called the ES-Tree. In this structure, each leaf node points to the data objects within the dataset $\mathbf{X}$ that correspond to a particular shard. Non-leaf nodes consist of index objects, i.e., minimum bounding rectangle (MBR), where pointer points to the children of the node and MBR represents the minimum bounding rectangle in d-dimensional space, defined as $\text{MBR} = (\text{mbr}_1, \text{mbr}_2, \dots, \text{mbr}_d)$. Here, $(\text{mbr}_1, \text{mbr}_2)$ represents the coordinates of the bottom-left corner in spatial dimensions, while $(\text{mbr}_3, \dots, \text{mbr}_d)$ are the bounds in non-spatial dimensions. Each non-leaf node in the encrypted, semi-hierarchical ES-Tree's second layer (also referred to as a semi-hierarchical shard node) has the form $(\text{MBR}, B(\cdot))$, where $B(\cdot)$ is a function calculating the score for each child node.

Given a dataset $\mathbf{X}$ and a query point Q, with X and Y as spatial attributes (i.e., $P[1]$ and $P[2]$), we partition $\mathbf{X}$ into shards where each node has a maximum capacity of $c = 3$. After DIME repairs the data, ES-Tree can efficiently perform skyline queries while protecting privacy.

The ES-Tree leverages a combination of semi-hierarchical structure and mixed encryption to enhance data privacy and security. The data object for each leaf node in the ES-Tree index is represented as follows:

$$\langle E(P[1]), E(P[2]), E_{S_{NS}}(P), \langle E(\mathrm{Enc}(P[3])), \ldots, E(\mathrm{Enc}(P[d]))\rangle\rangle,$$

where $E(\cdot)$ represents the Paillier encryption scheme [13], $\mathrm{Enc}(\cdot)$ represents ORE encryption [14]. Through pre-computation, $E_{S_{NS}}(P) = E\left(\sum_{i=3}^{d} P[i]\right)$, the ES-Tree can compute partial aggregations efficiently without storing $E(P[i])$ for $3 \leq i \leq d$. For non-leaf nodes, the index object is represented as

$$\langle E(\mathrm{mbr}_1), E(\mathrm{mbr}_2), E_{S_{NS}}(\mathrm{MBR})\rangle, \langle \mathrm{Enc}(\mathrm{mbr}_3), \ldots, \mathrm{Enc}(\mathrm{mbr}_d)\rangle, \mathrm{pointer}/B(\cdot) \tag{6}$$

where $E_{S_{NS}}(\mathrm{MBR}) = E\left(\sum_{i=3}^{d} \mathrm{mbr}_i\right)$, which improves both storage and computational efficiency.

The dashed lines represent that the path for accessing nodes is transparent, meaning the EVM and shards cannot derive additional information by observing access patterns. Therefore, in the semi-hierarchical structure, we calculate function $B(\cdot)$ based on the input encrypted matrix M and node bucket NB as follows:

$$C_{ij} = \prod_{k=1}^{\theta} \mathrm{SM}(M_i[k], \mathrm{NB}_{k,j}) \tag{7}$$

$$B(M, \mathrm{NB}) = \begin{bmatrix} C_{11} & \ldots & C_{1c} \\ \vdots & \ddots & \vdots \\ C_{\theta 1} & \ldots & C_{\theta c} \end{bmatrix} \tag{8}$$

where θ denotes the number of index objects in the parent node, $M = [\mathrm{IV}_1^T, \ldots, \mathrm{IV}_\theta^T]^T$, and $\mathrm{NB} = [E(O_1)^T, \ldots, E(O_\theta)^T]^T$, where $\mathrm{SM}(\cdot,\cdot)$ represents the secure multiplication based on the Paillier encryption scheme [15]. To ensure query unlinkability, IV_i, along with $E(1)$ and $E(0)$, forms each leaf node of the encrypted subtree. The result of $B(\cdot)$ is a matrix in which each row represents a shard-based encrypted leaf node that corresponds to a parent node index object.

To resist inference attacks based on leaf node capacity (i.e., when some leaf nodes are not fully populated, allowing the EVM or shard nodes to deduce if two queries are accessing the same node by observing node capacity), the blockchain shards with less data generates suitable noise tuples to pad nodes that lack sufficient data points. The dashed entries represent the added noise tuples. Since the padded data points will be dominated by at least one true skyline point, they do not affect the accuracy of the query results.

It is noteworthy that the ES-Tree index for a dataset $\mathbf{X}$ is generally maintained by the data owner. Although the semi-blind structure in the ES-Tree introduces additional computational overhead, the skyline query based on the ES-Tree only requires dominance operations on a subset of data points, without the need to traverse the entire dataset.

4.2 Secure Shard-Based Dominance Protocol for ES-Tree

The purpose of the secure shard-based dominance (SSD) protocol in the ES-Tree is to compute the dominance relationship between an encrypted object e and an encrypted skyline point s within a shard. Assume the ES-Tree contains an encrypted index I with objects e and skyline points s, and the query Q is encrypted as $E(Q)$. The full node entity (FNE) holds a private/public key pair (sk, pk), ensuring that details about e.MBR (or $e.P$) and $s.P$ are not exposed to the shard nodes or FNE.

The shard computes the Manhattan distances d_s from s to $E(Q)$ and d_e from e to $E(Q)$, with noise padding. The dominance metric β_1 is computed as follows:

$$\beta_1 = \prod_{i=1}^{2} d_i^{2^{\overline{\lambda}(2-i)}} \tag{9}$$

where $\overline{\lambda} = \|N\|/2$. Subsequently, the shard sends β_1 to FNE. Upon receiving β_1, FNE decrypts it as β_1' and encapsulates it in a masked comparison for $\langle d_1, d_2 \rangle$, calculated as:

$$d_i = \frac{\beta_i'}{2^{\overline{\lambda}(2-i)}} \tag{10}$$

Using $\beta_{i+1}' = \beta_i' - d_i \cdot 2^{\overline{\lambda}(2-i)}$, FNE securely compares d_1 and d_2, then sends f back to the shard. If $f = 1$ (i.e., $d_1 < d_2$), it implies that s is closer to Q than e; otherwise, e is closer to Q. If e is a data object, FNE assists in decrypting $\langle E(\text{Enc}(e.P[3])), \ldots, E(\text{Enc}(e.P[d])) \rangle$ to retrieve non-spatial attribute values. If e is an index object, FNE directly compares e with s to obtain $\wp$. Finally, the shard computes the dominance relationship $\Phi = f \wedge \wp_1 \wedge \cdots \wedge \wp_{d-2}$.

5 Secure Skyline Query Protocol for Shard-Based Querying

5.1 Single-Blockchain Region

Preprocessing The DO generates a secure index I based on the ES-Tree and uploads it to the distributed blockchain shards, and then assigns each shard S_i to handle the data within its allocated region. When the DApp sends a query $E(Q)$ to a shard S_i, S_i securely calculates $E(Q)$'s distance to the encrypted objects e within I_i, using $E(\text{dist}(e, Q))$. When e is a data object, S_i uses the secure squared Manhattan distance (SSMD) function to calculate the distance

$E(\mathrm{dist}^2(e,Q))$ [16]. When e is an index object, S_i calculates the distance based on e's MBR relative to Q, allowing for a more efficient approximation. Specifically, if the spatial coordinates of Q (i.e., $Q[1], Q[2]$) do not fall within e.MBR's spatial boundaries, then S_i calculates the distance as $E(\mathrm{dist}^2(e,Q)) = \mathrm{SSMD}(e, E(Q))$; if Q falls within e.MBR in one dimension only (e.g., $Q[1] \in e.\mathrm{MBR}[1]$), S_i calculates $E(\mathrm{dist}^2(e,Q)) = E(|e.\mathrm{mbr}_2 - Q[2]|^2)$. Otherwise, S_i calculates the full squared distance using $E(\mathrm{dist}^2(e,Q)) = E(|e.\mathrm{mbr}_2 - Q[2]|^2 + |e.\mathrm{mbr}_1 - Q[1]|^2)$. Next, S_i masks this computed distance with noise for security, represented by $\widetilde{\tau} = E(\mathrm{dist}^2(e,Q))^{\hbar^2}$, and sends it to the FNE. Here, the noise $\hbar$ is fixed for each query round to maintain security. Upon receiving $\widetilde{\tau}$, FNE decrypts and retrieves it, then returns $E(\sqrt{\widetilde{\tau}})$. Finally, FNE uses $E(\mathrm{dist}(e,Q)) = E(\sqrt{\widetilde{\tau}})$ for secure dominance calculation, ensuring that the modified distance value is suitable for determining shard dominance relationships while maintaining privacy.

Secure Blockchain-Based Skyline Query Protocol The SBSQ protocol initializes with an empty skyline result set $\mathbb{R} = \emptyset$ and a priority queue Ψ. The priority queue stores objects from the ES-Tree shard I_i, leveraging the secure minimum (SMIN) algorithm to keep the minimum distance object at the top of Ψ [15]. For a data object e, the shard S_i calculates the secure minimum distance $\mathrm{mindist}_E(e,Q)$ to $E(Q)$ as:

$$\mathrm{mindist}_E(e,Q) = E(\mathrm{dist}(e,Q)) \times E(S_{NS}(P))^{\hbar} \tag{11}$$

When e is an index object, S_i calculates $\mathrm{mindist}_E(e,Q) = E(\mathrm{dist}(e,Q)) \times E_{S_{NS}}(\mathrm{MBR})^{\hbar}$, where the plaintext value of $\mathrm{mindist}_E(\cdot)$ is scaled by a factor $\hbar$. The query proceeds as follows: S_i inserts the root node sRoot (containing $\mathrm{mindist}_E(\cdot)$) into Ψ and retrieves the top object e from Ψ. If e is an index object, S_i uses the SSD protocol in Section 5.2 to verify whether e is dominated by any skyline points already in $\mathbb{R}$. If e is dominated, it is pruned from Ψ; otherwise, if e is a non-dominated index object and meets the semi-blind condition, S_i calculates $B(\cdot)$ to retrieve e's child nodes e_i. Each child e_i is inserted into Ψ only if it is not dominated by any points in $\mathbb{R}$; otherwise, it is pruned, thus reducing the number of candidates in Ψ and improving the efficiency of the SMIN calculations. If e is a data object and is not dominated by any skyline points in $\mathbb{R}$, S_i inserts e into R with its corresponding values $\langle E(P[1]), E(P[2])\rangle, \mathrm{mindist}_E, \langle E(\mathrm{Enc}(P[3])), \ldots, E(\mathrm{Enc}(P[d]))\rangle$. The algorithm continues until $\Psi = \emptyset$.

Result Retrieval Once shard S_i obtains the encrypted skyline results $\mathbb{R}$ (where $E(s) \in \mathbb{R}$), it computes the encrypted response $\eta'[j] = E(s[j]) + \Psi[j] = E_{sk}(s[j]) \times E_{sk}(\Psi[j])$ for each coordinate $s[j]$, with random noise $\Psi[j]$ added to each $s[j]$ (where $1 \leq j \leq 2$). The encrypted response, including the noisy values Ψ and non-spatial attributes $\langle \mathrm{Enc}(P[3]), \ldots, \mathrm{Enc}(P[d])\rangle$, is sent to the DApp. The DApp forwards η' to the FNE. The FNE decrypts each $\eta'[j]$ by calculating $\eta[j] = \mathbf{X}_{sk}(\eta'[j])$ and returns $\eta[j]$ to the DApp. Finally, the DApp computes

$s[j] = \eta[j] - \Psi[j]$ to obtain the spatial coordinates of the skyline result. Using the shared secret key SK, the DApp decrypts and retrieves the non-spatial attribute values as part of the final result.

5.2 Multi-Shard Blockchain Regions

When the DApp query $E(Q)$ targets data across multiple shards (including query regions distributed across various shards), the shard S_i not only performs secure skyline queries on its local secure index I_i for $E(Q)$ but also forwards $E(Q)$ to other relevant shards within the ES-Tree network. Each shard performs a secure skyline query on its respective secure index and sends the partial results back to the FNE, which aggregates the results from multiple shards.

The FNE performs a secure merge of the skyline results based on the secure indices and returns the merged skyline result to the original shard S_i, which then forwards the final aggregated result to the DApp. Similar to the process in Section 5.1, this protocol uses secure indexing and merging to ensure that each shard's data remains private. The following section introduces the skyline merging algorithm of sharded blockchain (SMSB) to efficiently combine results from multiple shards for a comprehensive skyline result.

First, this protocol defines skyline mergeability in Definition 5. According to this criterion, the FNE can efficiently merge all outputs from individual SBSQ calculations into a single secure skyline computation, which we denote as the secure skyline merge.

Definition 5. Given a dataset $\mathbf{X} = \mathbf{X}_1 \cup \cdots \cup \mathbf{X}_u \cup \cdots \cup \mathbf{X}_U$, where $\mathbf{X}_u$ is the u-th shard of data, the skyline result for $\mathbf{X}$ is defined as $\mathrm{SKY}(\mathbf{X}) = \mathrm{SKY}(\mathrm{SKY}(\mathbf{X}_1) \cup \cdots \cup \mathrm{SKY}(\mathbf{X}_U))$.

In the SMSB algorithm, the FNE merges the skyline results from each shard into a new dataset $\mathbf{X}'$ and computes the secure minimum distance ξ of each object's $\mathrm{mindist}_E(\cdot)$ using the SMIN algorithm. The FNE then performs a secure integer comparison to evaluate if $\mathrm{mindist}_E(\cdot)$ for each object in $\mathbf{X}'$ is equal to ξ, obtaining the encrypted comparison result ϖ. The FNE sends ϖ to the shard S_i, which decrypts $\mathbf{X}_{sk}(\varpi_u) = 1$ if the corresponding object's $\mathrm{mindist}_E(\cdot)$ matches ξ and returns the index of the matching object to FNE. Next, the FNE adds the corresponding object $E(\mathfrak{P}_{\min})$ to the intermediate skyline result $\hat{\mathfrak{R}}$. If more than one $\mathbf{X}_{sk}(\varpi_u) = 1$ occurs, the FNE selects the first matching object and uses the SSD to determine if any other objects in $\mathbf{X}'$ are dominated by $\mathfrak{P}_{\min}$, removing any dominated objects from $\mathbf{X}'$ and $\mathfrak{P}_{\min}$ from further comparison. This process continues until $\mathbf{X}' = \emptyset$.

After FNE returns the aggregated skyline result $\hat{\mathfrak{R}}$ to the original shard S_i, S_i performs the SMSB algorithm to further refine $\hat{\mathfrak{R}}$ using its secure index and eventually obtain the final skyline result $\mathfrak{R}$. Lastly, S_i sends $\mathfrak{R}$ back to the DApp in a manner similar to Section 5.1.

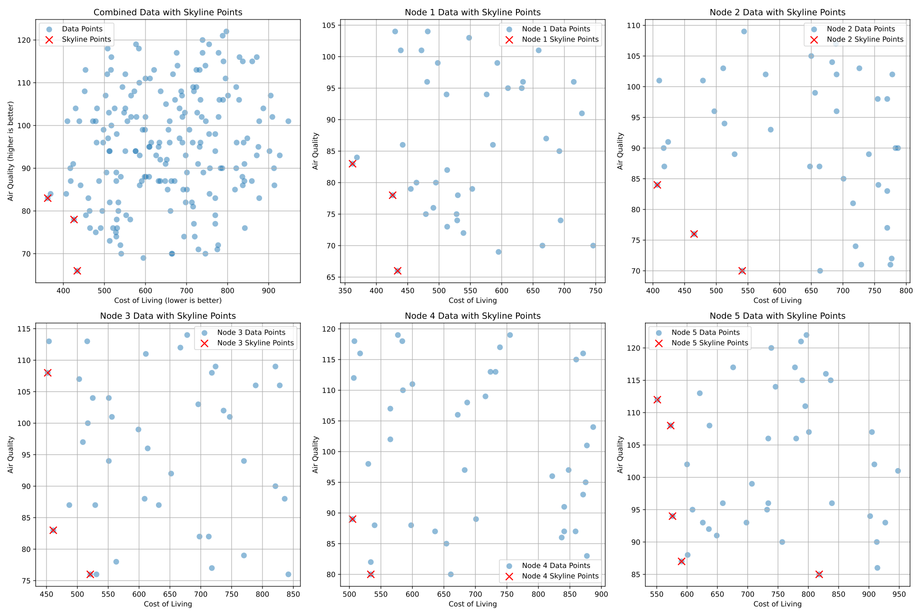


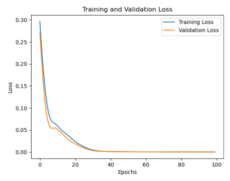
Fig. 3. Distribution of the 2-dimension dataset (5 shards, 200 data)

6 Experiments

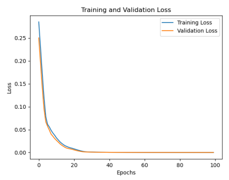
6.1 Experimental Setup

Datasets Fig. 3 shows the 2-dimension data of the dataset used in the experiments for better understanding. This part of dataset comprises 200 data points distributed across five shards, with each data point containing two dimensions: "Cost of Living" and "Air Quality." Each node includes labeled skyline points, representing locally optimal choices within the shard. The dataset simulates a distributed data environment in sharded blockchain networks, supporting experimental evaluation of privacy-preserving and resilient skyline querying for Web 3.0 applications.

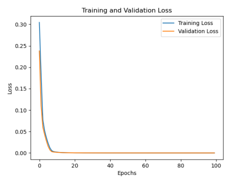
Competing Methods We compare the proposed algorithm with the following methods. (1) **BNL** [17] is a straightforward algorithm that compares every pair of data points to identify skyline points, ensuring correctness but with high computational cost, especially for large datasets. (2) **DSQ** [18] is designed for distributed environments, which splits the skyline query task across multiple nodes, computing partial skylines and merging them, offering better scalability and efficiency compared to brute force methods. (3) **NN** [19] uses R-tree indexing for efficient nearest-neighbor searches to find skyline points, reducing the search space and improving efficiency over brute force methods. (4) **GNN** first merges



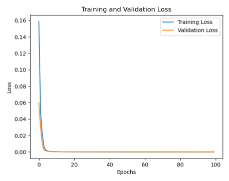
(a) Number of data $\| \boldsymbol{X} \|= 100$



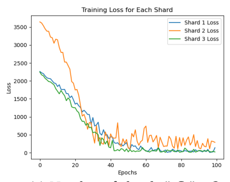
(b) Number of data $\| \boldsymbol{X} \|= 200$



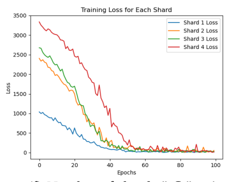
(c) Number of data $\| \boldsymbol{X} \|= 500$



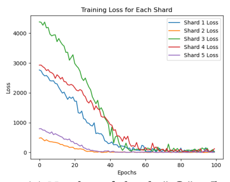
(d) Number of data $\| \boldsymbol{X} \|= 1000$



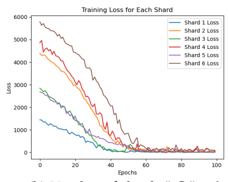
(e) Number of shards $\| \boldsymbol{S} \|= 3$



(f) Number of shards $\| \boldsymbol{S} \|= 4$

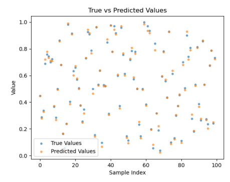


(g) Number of shards $\| \boldsymbol{S} \|= 5$

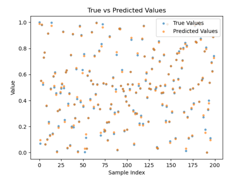


(h) Number of shards $\| \boldsymbol{S} \|= 6$

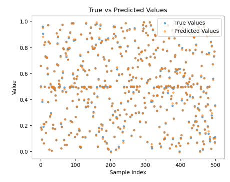
Fig. 4. Convergence of the loss function for each shard of data and the overall data.



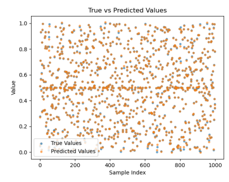
(a) Number of data $\| \boldsymbol{X} \|= 100$



(b) Number of data $\| \boldsymbol{X} \|= 200$



(c) Number of data $\| \boldsymbol{X} \|= 500$



(d) Number of data $\| \boldsymbol{X} \|= 1000$

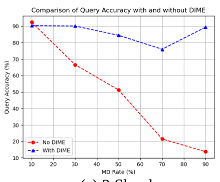
Fig. 5. Comparison of MD repaired results with true values in the data (2-dimension).

all data from distributed shards and then applies brute force to find the global skyline, ensuring correctness but with inefficiency due to the data merging step.

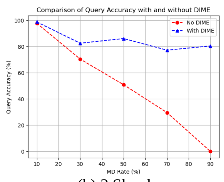
Implementation Details The implementation is carried out on a Windows 11 Enterprise system (23H2) system with Python version 3.9.5 and PyCharm version 2021.2.4 as the primary IDE. The required libraries include NumPy 1.20.3, Pandas 1.2.4, Scikit-learn 0.24.2, Matplotlib 3.4.3, SciPy 1.6.3, Rtree 0.9.7, and Pytest 6.2.4, which supports efficient computation, data manipulation, and visualization. The hardware configuration consists of a 12th Gen Intel(R) Core(TM) i7-12700KF 3.60 GHz, 32GB RAM, and a 512GB SSD.

6.2 Performance Evaluation

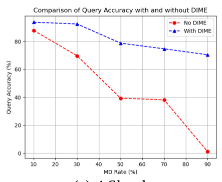
Convergence of DIME Fig. 4 illustrates the convergence of the loss function of DIME under varying data sizes and shard counts, evaluating its effectiveness in addressing the missing dimension problem. Fig. 4 (a) to Fig. 4 (d) show the training and validation loss curves for different data sizes: 100, 200, 500, and 1000 data points. As the dataset size increases, particularly for 500 and 1000 data points, the loss decreases more rapidly and stabilizes at a lower value. This



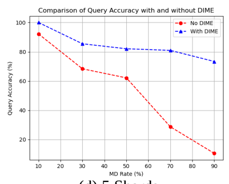
(a) 2 Shards



(b) 3 Shards

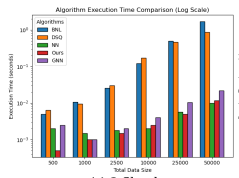


(c) 4 Shards

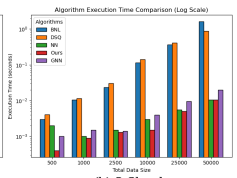


(d) 5 Shards

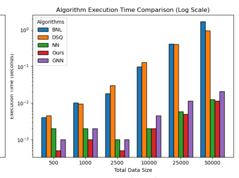
Fig. 6. Comparison of data repair with and without DIME.



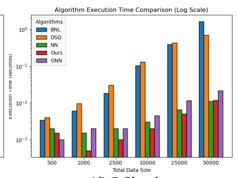
(a) 2 Shards



(b) 3 Shards



(c) 4 Shards



(d) 5 Shards

Fig. 7. Comparison of query times of different algorithms.

demonstrates that larger datasets facilitate quicker learning and more accurate imputation by the DIME. Across all data sizes, the validation loss converges closely to the training loss, indicating that the model generalizes well and does not overfit, even as the dataset grows. Fig. 4 (e) to Fig. 4 (h) depict the loss functions for different shard configurations (3, 4, 5, and 6 shards) while keeping the data size constant. Each shard's loss curve exhibits slight variations at the beginning but eventually converges to a stable value, suggesting that the DIME is able to effectively handle data imputation across different shards. With an increasing number of shards, the convergence in the early stages shows some fluctuations, particularly in higher shard counts, but all shards eventually achieve stable loss values. This indicates that the model is resilient to shard-based partitioning and converges effectively across distributed settings.

MD and True Value Fig. 5 presents a comparison between the true and predicted values for MD repair, illustrating the model's effectiveness in imputing missing values. The plots show the relationship between the true values and the predicted values across different dataset sizes. As the data size increases, the scatter plots become tighter, indicating that the model becomes more accurate in its predictions as the amount of data increases. In Fig. 5 (a), with 100 data points, the points are more scattered, showing larger deviations between the true and predicted values. As we move to Fig. 5 (b), (c), and (d), with 200, 500, and 1000 data points, the predicted values increasingly align with the true values, reflecting improved accuracy.

Repairing Accuracy Fig. 6 compares the query accuracy of data repair with and without the DIME under different shard configurations (2 to 5 shards) and

MD rates. As the MD rate increases from 10% to 90%, the query accuracy drops significantly for both scenarios (with and without DIME). This indicates that as the proportion of missing data increases, the task of repairing and querying data becomes more challenging, which results in lower accuracy. For all shard configurations, the query accuracy with DIME remains higher compared to without DIME. For example, in Fig. 6 (a), when the MD rate is 90%, the accuracy with DIME is significantly higher than without DIME, which shows that DIME helps maintain higher accuracy even in the presence of higher levels of missing data. As the number of shards increases, the difference in query accuracy between the scenarios with and without DIME becomes more pronounced, especially as the MD rate increases. In 6 (d), with 5 shards, the accuracy gap between the two scenarios is widest, highlighting that DIME plays a crucial role in maintaining data repair quality in more complex distributed environments.

Query Time Fig. 7 compares the execution times of various algorithms, including BNL, DSQ, NN, GNN, and Ours proposed method, across different shard configurations (2 to 5 shards) and varying data sizes. The comparison is presented in log scale to highlight the differences in execution times for large-scale datasets. As the number of shards increases from 2 to 5 (Fig. 7 (a) to (d)), the execution time for most algorithms tends to increase. This is due to the increased complexity of handling more shards in a distributed environment. However, our algorithm demonstrates relatively stable execution times compared to others as the number of shards grows, indicating better scalability and faster execution times, especially for larger data sizes. For instance, in Fig. 7 (a), our algorithm outperforms the other algorithms for smaller datasets, while it maintains relatively low execution times compared to other methods like BNL and DSQ, which experience higher execution times for larger datasets. This reflects the fact that other algorithms are less efficient at processing distributed data, which makes them less scalable to larger datasets or more fragments. GNN also performs well, but its execution time is still higher than ours, especially for larger data sizes and more shards.

7 Related Works

There has been some work related to the skyline query that has been proposed. Zheng et al. [20] introduces SecSkyline, a lightweight cryptographic framework for secure skyline queries over encrypted databases, achieving substantial query latency improvements. Cuzzocrea et al. [21] presents AOPF-SFS, an algorithm for efficient skyline query processing over encrypted data in cloud databases, along with the eSkyline prototype, which supports secure, order-independent query evaluations. Wang et al. [22] expresses efficient location-based skyline queries on dynamic encrypted datasets, using a blockchain-backed DSV-tree to ensure dataset confidentiality, authenticity, and forward privacy. Kertiou et al. [23] proposes the P2P-IoMT protocol, a dynamic routing solution for IoMT applications that improves network lifetime, energy efficiency, and packet delivery by using Skyline-based multi-criteria decision-making. Zeng et al. [24] presents a blockchain-based, privacy-preserving skyline query scheme that enhances computational efficiency and security using improved encryption and sparse matrix techniques. Yin et al. [24] introduces a blockchain-based framework for secure and traceable skyline queries, employing consensus and encryption to ensure both privacy and the ability to track query steps. Chen et al. [25] proposes the HMMSQ scheme, which enables efficient, privacy-preserving skyline queries on outsourced databases while detecting malicious server activities and significantly reducing query latency.

8 Conclusion

In this work, we proposed a novel approach to privacy-preserving and resilient skyline querying in sharded blockchain networks, focusing on addressing the challenges posed by missing data dimensions. Our method introduces a DIME, which effectively handles missing dimensions through autoencoders, alongside an ES-Tree index and a SSD protocol. We also developed the SBSQ algorithm, demonstrating the effectiveness of our approach in improving query accuracy and resilience in distributed systems. Experimental results confirm that our approach outperforms traditional skyline querying methods, providing scalability and efficiency in large-scale, distributed blockchain environments.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Rudman, R., Bruwer, R.: Defining web 3.0: opportunities and challenges. The electronic library **34**(1), 132–154 (2016)
2. Johnson, M., Jones, M., Shervey, M., Dudley, J.T., Zimmerman, N.: Building a secure biomedical data sharing decentralized app (dapp): tutorial. Journal of medical Internet research **21**(10), e13601 (2019)
3. Zheng, P., Jiang, Z., Wu, J., Zheng, Z.: Blockchain-based decentralized application: A survey. IEEE Open Journal of the Computer Society **4**, 121–133 (2023)
4. Beltrán, E.T.M., Pérez, M.Q., Sánchez, P.M.S., Bernal, S.L., Bovet, G., Pérez, M.G., Pérez, G.M., Celdrán, A.H.: Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. IEEE Communications Surveys & Tutorials (2023)
5. Islam, S., Apu, K.U.: Decentralized vs. centralized database solutions in blockchain: Advantages, challenges, and use cases. Global Mainstream Journal of Innovation, Engineering & Emerging Technology **3**(4), 58–68 (2024)
6. Zhou, L., Xiong, X., Ernstberger, J., Chaliasos, S., Wang, Z., Wang, Y., Qin, K., Wattenhofer, R., Song, D., Gervais, A.: Sok: Decentralized finance (defi) attacks. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 2444–2461. IEEE (2023)
7. Yuan, D., Zhang, L., Li, S., Sun, G.: Skyline query under multidimensional incomplete data based on classification tree. Journal of Big Data **11**(1), 72 (2024)

8. Bai, M., Han, Y., Yin, P., Wang, X., Li, G., Ning, B., Ma, Q.: S_ids: An efficient skyline query algorithm over incomplete data streams. Data & Knowledge Engineering **149**, 102258 (2024)
9. Rahman, M.S., Azharul Hasan, K.: Computing skyline query on incomplete data. In: International Conference on Big Data, IoT and Machine Learning. pp. 657–672. Springer (2023)
10. Liu, Y., Liu, J., Salles, M.A.V., Zhang, Z., Li, T., Hu, B., Henglein, F., Lu, R.: Building blocks of sharding blockchain systems: Concepts, approaches, and open problems. Computer Science Review **46**, 100513 (2022)
11. Chen, W.H., Lai, C.C.: Distributed edge computing combined with reinforcement learning for low-latency probabilistic skyline queries. In: 2024 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS). pp. 1–5. IEEE (2024)
12. Yao, A.C.C.: How to generate and exchange secrets. In: 27th annual symposium on foundations of computer science (Sfcs 1986). pp. 162–167. IEEE (1986)
13. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques. pp. 223–238. Springer (1999)
14. Lewi, K., Wu, D.J.: Order-revealing encryption: New constructions, applications, and lower bounds. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1167–1178 (2016)
15. Elmehdwi, Y., Samanthula, B.K., Jiang, W.: Secure k-nearest neighbor query over encrypted data in outsourced environments. In: 2014 IEEE 30th International Conference on Data Engineering. pp. 664–675. IEEE (2014)
16. Li, Z., Wang, H., Zhang, S., Zhang, W., Lu, R.: Secknn: Fss-based secure multiparty knn classification under general distance functions. IEEE Transactions on Information Forensics and Security **19**, 1326–1341 (2023)
17. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. Journal of the ACM (JACM) **22**(4), 469–476 (1975)
18. Hose, K., Vlachou, A.: A survey of skyline processing in highly distributed environments. The VLDB Journal **21**, 359–384 (2012)
19. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: VLDB'02: Proceedings of the 28th International Conference on Very Large Databases. pp. 275–286. Elsevier (2002)
20. Zheng, Y., Wang, W., Wang, S., Jia, X., Huang, H., Wang, C.: Secskyline: Fast privacy-preserving skyline queries over encrypted cloud databases. IEEE Transactions on Knowledge and Data Engineering **35**(9), 8955–8967 (2022)
21. Cuzzocrea, A., Karras, P., Vlachou, A.: Effective and efficient skyline query processing over attribute-order-preserving-free encrypted data in cloud-enabled databases. Future Generation Computer Systems **126**, 237–251 (2022)
22. Wang, Z., Zhang, L., Ding, X., Choo, K.K.R., Jin, H.: A dynamic-efficient structure for secure and verifiable location-based skyline queries. IEEE Transactions on Information Forensics and Security **18**, 920–935 (2022)
23. Kertiou, I., Laouid, A., Saber, B., Hammoudeh, M., Alshaikh, M.: A p2p multipath routing algorithm based on skyline operator for data aggregation in iomt environments. PeerJ Computer Science **9**, e1682 (2023)
24. Zeng, S., Hsu, C., Harn, L., Liu, Y., Liu, Y.: Efficient and privacy-preserving skyline queries over encrypted data under a blockchain-based audit architecture. IEEE Transactions on Knowledge and Data Engineering (2024)
25. Chen, Y., Liu, L., Chen, R., Fu, S., Yang, Y.: Honest-majority maliciously secure skyline queries on outsourced data. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management. pp. 344–353 (2024)