Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Privacy-Preserving and Robust Skyline Querying over Incomplete Data in Sharded Blockchain Networks

Kening Zhang
Hong Kong Polytechnic University
C.K.M. Lee
Hong Kong Polytechnic University
Yung Po Tsang
yungpo.tsang@polyu.edu.hk

Hong Kong Polytechnic University

Research Article

Keywords: Skyline query, Incomplete data, Missing-value imputation, Privacy preservation, Homomorphic encryption, Sharded blockchain, Web 3.0

Posted Date: November 19th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11615078/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Privacy-Preserving and Robust Skyline Querying over Incomplete Data in Sharded Blockchain Networks

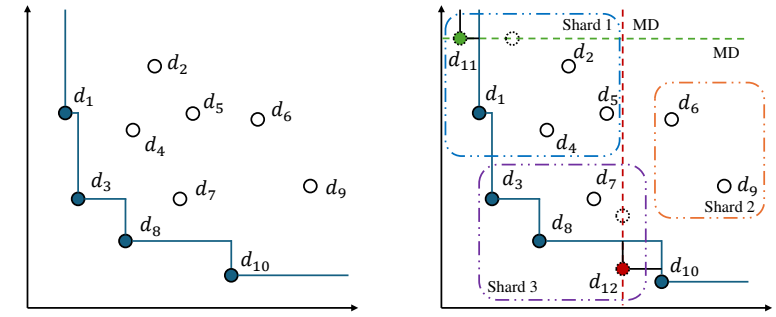
Abstract

Decentralised applications (DApps) in Web 3.0 increasingly need multi-criteria queries such as location-based skylines. These queries return the records near a user that are not dominated on any criterion, for example rentals that are close, cheap and well reviewed. In a sharded blockchain, supporting such queries is hard for three reasons: records contributed by many parties have missing attribute values, records and queries are sensitive, and results must be assembled across shards. This paper proposes a framework that addresses all three challenges. First, the Dimensional Imputation and Enhancement network (DIME) repairs missing values before encryption. It is a mask-aware denoising autoencoder trained only on observed values and enhanced with positional encodings and neighbourhood statistics of the records' locations. Second, the data owner builds an Encrypted Shard-Tree for every shard, a padded R-tree whose entries are Paillier ciphertexts. On this index, a secure shard-based dominance protocol and a secure branch-and-bound search compute each shard's skyline with the help of a key-holding full node that only sees blinded, shuffled values. A constant-round protocol then merges the shards' results. We prove the protocols correct, characterise their leakage and evaluate a full implementation with real cryptography on two real and three synthetic datasets. With 30% missing values, DIME reduces the imputation error by up to 15% relative to MissForest on real data, whereas skyline semantics for incomplete data attain F_1 scores as low as 0.02. Every secure query returned the exact skyline. The secure search was up to 7.5 times faster than an index-free secure scan and 25 times faster than a fully oblivious protocol, and its latency grew only 5.7-fold when the data grew 25-fold.

Keywords: Skyline query, Incomplete data, Missing-value imputation, Privacy preservation, Homomorphic encryption, Sharded blockchain, Web 3.0

1 Introduction

Web 3.0 moves data storage and computation from centralised platforms to networks of mutually distrusting nodes [1], which avoids the performance bottlenecks and single points of failure of centralised databases [2]. Blockchains provide the shared ledger of this ecosystem and have enabled a large class of decentralised applications, or DApps,



(a) Traditional skyline query over certain data

(b) Skyline query over MD data in sharing blockchains

Fig. 1: Comparison of a traditional skyline query over complete data and a skyline query over data with missing dimensions. Missing attribute values change which records appear to be Pareto-optimal.

in finance, healthcare, data sharing and peer-to-peer marketplaces [3–5], as well as decentralised learning systems that train models without a central coordinator [6]. As these applications grow, users increasingly need analytical queries rather than simple key look-ups. A tenant on a decentralised rental platform wants listings that are close to a target location and also cheap and well reviewed, and a DeFi user wants liquidity pools that offer low fees, low slippage and high volume at the same time. The skyline operator [7] is the standard tool for such multi-criteria decisions. It returns all records that are not dominated by any other record, that is, the Pareto-optimal choices, without asking the user to weight the criteria. When one of the criteria is the distance to a query location, the operator becomes a location-based or spatial skyline [8, 9].

Supporting skyline queries inside a sharded blockchain raises three intertwined difficulties, which Fig. 1 illustrates.

Records in a decentralised network are contributed by many independent parties, collected by heterogeneous devices and synchronised across shards under network churn, so some attribute values are inevitably missing [10–13]. In the New York City Airbnb data used in our evaluation, for example, 27–35% of the price and review attributes are missing. Missing values break the transitivity of dominance. If dominance is evaluated only on commonly observed attributes, as skyline algorithms designed for incomplete data do [10], cyclic dominance can eliminate almost all genuine Pareto-optimal records. In our experiments this approach attains an F_1 score of only 0.02 on the California housing data with 30% missing values.

Records are commercially or personally sensitive, and the query location reveals where the user is. Shard nodes are run by third parties and should learn neither the data nor the query nor the result. Encrypting the data with order-preserving or order-revealing encryption [14] keeps comparisons cheap, but it discloses the order and equality of values to the evaluating node, and this leakage enables inference attacks on real databases [15]. Fully oblivious protocols such as that of Liu et al. [16] hide

everything, but they touch every record for every result, so their cost grows with the product of the dataset size and the skyline size.

Sharding partitions the network into committees that process disjoint parts of the state in parallel [17–20]. A skyline over the whole dataset must therefore be assembled from per-shard results, and every cross-shard step adds communication rounds and latency. Distributed skyline algorithms [21, 22] solve the merge in plaintext but offer no confidentiality.

As Table 1 summarises, existing work addresses at most two of these challenges at a time. Secure skyline schemes [16, 23–26] assume complete data and a centralised cloud. Blockchain-based query schemes [9, 27–30] focus on verifiability or auditing and also assume complete data. Skyline algorithms for incomplete data [10–12, 31] operate on plaintext.

This paper proposes a framework for privacy-preserving and robust location-based skyline querying over incomplete data in sharded blockchain networks. The data owner first repairs missing values with a learned imputation model and then builds an encrypted index for every shard. Shards answer encrypted queries in parallel with the help of a key-holding full-node entity, and their partial results are merged securely. Our contributions are as follows.

- **Problem and framework.** We formalise privacy-preserving location-based skyline queries over incomplete data in a sharded blockchain. The system model separates on-chain coordination from off-chain secure computation, and we give a semi-honest threat model and a precise leakage profile for each party in Sections 3 and 4.
- **DIME.** Section 5 presents the Dimensional Imputation and Enhancement network. It is a mask-aware denoising autoencoder that is trained only on observed entries and enhanced with spatial context, namely a sinusoidal encoding of the location and neighbourhood statistics of each attribute. On two real datasets DIME attains the lowest imputation error at every missing rate under MCAR, up to 15% below MissForest, the strongest baseline, and it yields the most accurate skylines in most settings. It also attains the lowest error on data with naturally occurring missing values.
- **ES-Tree and SSD.** Section 6 introduces the Encrypted Shard-Tree, a padded R-tree whose entries are Paillier ciphertexts and which stores a precomputed encrypted attribute sum for every entry. On top of it, the Secure Shard-based Dominance protocol decides dominance between encrypted entries. It reveals a single bit per candidate to the shard and nothing but blinded, shuffled values to the key holder.
- **SBSQ and SMSB.** We develop the Secure Blockchain-based Skyline Query algorithm, a secure branch-and-bound search over the ES-Tree, and the Skyline Merging algorithm for Sharded Blockchains, which merges per-shard skylines in a constant number of rounds. Sections 7 and 8 prove their correctness and analyse their complexity and security.
- **Evaluation.** We implement the complete system with real 1024-bit and 2048-bit Paillier cryptography. Section 9 evaluates it on two real and three synthetic datasets against six imputation methods, an incomplete-data skyline algorithm, two secure skyline baselines and four plaintext skyline algorithms. Every secure query in the evaluation returned exactly the plaintext skyline.

The remainder of the paper is organised as follows. Section 2 reviews related work. Section 3 introduces preliminaries and the problem definition, and Section 4 presents the system and threat models. Sections 5 to 7 describe DIME, the ES-Tree with its secure primitives, and the query protocols. Section 8 analyses security, Section 9 reports the experimental evaluation, Section 10 discusses limitations, and Section 11 concludes the paper.

2 Related Work

The skyline operator was introduced to databases by Börzsönyi et al. [7], who proposed the block-nested-loop (BNL) and divide-and-conquer algorithms; the underlying problem of finding maximal vectors dates back to Kung et al. [32]. Sort-filter-skyline [33] presorts the data by a monotone scoring function so that a record that survives the filter is final. Index-based methods are progressive: the nearest-neighbour algorithm of Kossmann et al. [34] recursively partitions the space around nearest neighbours, and branch-and-bound skyline (BBS) [35] visits R-tree [36] entries in ascending order of their minimum distance to the origin, which makes it I/O-optimal. Song et al. [37] recently studied index structures dedicated to skyline queries. When one criterion is the distance to a query point, the problem becomes a location-based or spatial skyline [8], which is the query type studied in this paper. In distributed settings, each site computes a local skyline and a coordinator merges them [21, 22, 38]; edge-computing variants further reduce latency [39]. The skyline operator is also used inside networked systems, for example to select routing paths by several criteria at once in Internet of Medical Things deployments [40]. None of these algorithms protects the data or the query.

Khalefa et al. [10] defined dominance over incomplete records on the dimensions observed in both records. Under this definition dominance is no longer transitive and may be cyclic, so a record can be eliminated by a record that is itself dominated; the authors proposed the ISkyline algorithm to process such data efficiently. Subsequent work studied top-k dominating queries [31], classification-tree-based processing [11], incomplete data streams [12] and further pruning strategies [13]. All of these methods change the query semantics instead of repairing the data, and all of them operate on plaintext. We show in Section 9.2 that, on real data, the semantics of [10] recovers only a small fraction of the skyline of the complete data, which motivates imputing the data before encryption.

Liu et al. [16] proposed the first secure skyline protocol on a two-server cloud platform based on Paillier encryption [41]. Their protocol repeatedly selects the record with the minimum attribute sum obliviously and obliviously eliminates the records it dominates, so it hides the access pattern at the price of touching every record in every iteration. SecSkyline [23] reduces the cost with lightweight secret-sharing-based primitives. Cuzzocrea et al. [24] process skylines over encrypted data without order-preserving encryption. Chen et al. [25] achieve security against malicious adversaries in the honest-majority setting, and Hao et al. [26] support multi-party skyline queries for online medical diagnosis. The secure k-nearest-neighbour protocol of Elmehdwi et al. [42] introduced the secure multiplication and secure minimum sub-protocols

Table 1: Comparison with representative related work. A check mark indicates that a capability is supported and a dash that it is not.

Work	Encrypted data and query	Incomplete data	Blockchain	Sharded / distributed
Khalefa et al. [10]; Yuan et al. [11]	–	✓	–	–
Balke et al. [21]; Hose and Vlachou [22]	–	–	–	✓
Liu et al. [16]; Zheng et al. [23]	✓	–	–	–
Chen et al. [25]; Hao et al. [26]	✓	–	–	–
Xu et al. [27]	–	–	✓	–
Wang et al. [9]; Zeng et al. [29]	✓	–	✓	–
This work	✓	✓	✓	✓

that are widely reused in this line of work, and function-secret-sharing-based protocols have recently been applied to secure kNN classification [43]. Schemes that use order-preserving or order-revealing encryption [14, 44] are efficient but disclose order relations, which enables inference attacks [15]. All of these schemes assume complete data and a centralised or fixed multi-server deployment.

vChain [27] supports verifiable Boolean range queries over blockchain databases, and GEM^2-tree [28] reduces the gas cost of authenticated range queries. Wang et al. [9] support secure and verifiable location-based skyline queries on dynamic encrypted datasets using a blockchain-backed index. Zeng et al. [29] combine encrypted skyline queries with a blockchain-based audit architecture, Yin et al. [30] provide secure and traceable query services for the Internet of Things, and Tian et al. [45] use a blockchain to assist privacy-preserving image retrieval over multiple clouds. Sharding protocols such as Elastico [17], OmniLedger [18] and RapidChain [19], as well as account-reconfiguration schemes [46], scale throughput by partitioning the state; Liu et al. [20] survey their building blocks. To the best of our knowledge, no prior scheme supports privacy-preserving skyline queries across the shards of a sharded blockchain, nor over incomplete data.

Rubin [47] classified missingness mechanisms as missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR); Little and Rubin [48] give a comprehensive treatment. Classical imputers include mean substitution, k-nearest-neighbour imputation [49], multivariate imputation by chained equations (MICE) [50] and the random-forest-based MissForest [51], which is among the strongest methods in recent benchmarks [52]. Fuzzy matching approaches have also been proposed [53]. Deep imputers include the denoising-autoencoder-based MIDA [54, 55] and the adversarial GAIN [56]. These methods treat records as exchangeable rows and ignore the spatial organisation of the data, even though nearby records tend to have similar attribute values [57]. DIME exploits this property explicitly.

3 Preliminaries and Problem Definition

Table 2 summarises the notation used in this paper.

Table 2: Summary of notation.

Symbol	Meaning
$\mathcal{D} = \{p_1, \ldots, p_n\}$	dataset; $p = (p.x, p.y, p.a_1, \ldots, p.a_k)$, $d = k + 2$
$q = (q.x, q.y)$	query location
$\delta(p, q)$	squared Euclidean distance between p and q
$p \prec_q p'$	p dominates p' with respect to q
$\mathrm{SKY}_q(\mathcal{D})$	location-based skyline of $\mathcal{D}$ for q
$\widetilde{\mathcal{D}}$, M	incomplete dataset and its mask; $M_{ij} = 1$ marks a missing entry
$\mathcal{D}_1, \ldots, \mathcal{D}_m$	region-based partition of $\mathcal{D}$ over m shards $\mathcal{S}_1, \ldots, \mathcal{S}_m$
$\mathcal{T}_i$, c	ES-Tree of shard $\mathcal{S}_i$ and its node capacity
$e.\ell$, $e.u$, $e.A$, $e.\sigma$	spatial corners, attribute lower bounds and attribute sum of entry e
$\mathrm{key}_q(e)$	search key $\delta_{\min}(e, q) + e.\sigma$
$[\![x]\!]$	Paillier ciphertext of x under the public key pk
$\mathcal{F}$	full-node entity holding the secret key sk
κ	statistical security parameter, $\kappa = 40$

3.1 Location-Based Skyline

A record $p \in \mathcal{D}$ has a location $(p.x, p.y)$ and k non-spatial attributes $p.a_1, \ldots, p.a_k$. Without loss of generality, smaller attribute values are preferred; attributes for which larger is better are negated during preprocessing. All values are scaled to integers in $[0, 2^{16})$.

Definition 1 (Dominance with respect to a query) Given a query location q, a record p dominates a record p', written $p \prec_q p'$, if $\delta(p, q) \leq \delta(p', q)$, $p.a_j \leq p'.a_j$ for all $j \in \{1, \ldots, k\}$, and at least one of these $k + 1$ inequalities is strict.

Definition 2 (Location-based skyline) The skyline of $\mathcal{D}$ with respect to q is $\mathrm{SKY}_q(\mathcal{D}) = \{p \in \mathcal{D} \mid \nexists\, p' \in \mathcal{D} : p' \prec_q p\}$.

The squared distance is used instead of the Euclidean distance because both induce the same order and the squared distance can be evaluated with additions and multiplications only. For a set of records summarised by a minimum bounding rectangle with spatial corners ℓ, u and attribute lower bounds A, we use the minimum squared distance

$$\delta_{\min}(e, q) = \sum_{t \in \{x, y\}} \Delta_t^2, \qquad \Delta_t = [q.t < e.\ell_t]\,(e.\ell_t - q.t) + [q.t > e.u_t]\,(q.t - e.u_t), \quad (1)$$

where $[\cdot]$ is the Iverson bracket. A key property used by all progressive skyline algorithms is that a monotone function of the comparison vector orders records consistently with dominance.

Lemma 1 *Let* $\mathrm{key}_q(p) = \delta(p,q) + \sum_{j=1}^{k} p.a_j$. *If* $p \prec_q p'$, *then* $\mathrm{key}_q(p) < \mathrm{key}_q(p')$. *Moreover, if every component of the comparison vector of* p *is no larger than that of* p', *then* $p \prec_q p'$ *if and only if* $\mathrm{key}_q(p) < \mathrm{key}_q(p')$.

Proof If $p \prec_q p'$, every summand of $\mathrm{key}_q(p)$ is at most the corresponding summand of $\mathrm{key}_q(p')$ and at least one is strictly smaller, so the sum is strictly smaller. Conversely, if all components are no larger and the sums differ, at least one component must be strictly smaller. □

The second part of Lemma 1 lets the secure dominance test replace the "at least one strict" condition, which would otherwise require a disjunction over $k+1$ encrypted bits, by a single comparison of keys.

3.2 Incomplete Data

An incomplete dataset $\widetilde{\mathcal{D}}$ is a copy of $\mathcal{D}$ in which the entries with $M_{ij} = 1$ are missing. Locations are assumed to be always observed, because a record without a location cannot be placed in a shard. We consider the three mechanisms of Rubin [47]. Under MCAR the mask M is independent of the data. Under MAR the probability that a value is missing depends on observed values, which in our experiments means on the location. Under MNAR it depends on the missing value itself; in our experiments, larger and therefore worse values are more likely to be missing.

Definition 3 (Robust skyline over incomplete data) Given $\widetilde{\mathcal{D}}$ and q, the goal is to return a set R that is as close as possible to $\mathrm{SKY}_q(\mathcal{D})$, the skyline of the unobserved complete data. Quality is measured by the precision, recall and F_1 score of R with respect to $\mathrm{SKY}_q(\mathcal{D})$.

3.3 Paillier Cryptosystem

The Paillier cryptosystem [41] has public key $pk = N = PQ$ for large primes P, Q, and plaintext space $\mathbb{Z}_N$. With generator $g = N+1$, a message x is encrypted as $[\![x]\!] = (1 + xN)\, r^N \bmod N^2$ for random r. It is semantically secure under the decisional composite residuosity assumption, abbreviated DCR, and it is additively homomorphic:

$$[\![x]\!] \cdot [\![y]\!] = [\![x+y]\!], \qquad [\![x]\!]^{\lambda} = [\![\lambda x]\!], \qquad [\![x]\!]^{-1} = [\![-x]\!]. \tag{2}$$

Negative values are represented by their residues modulo N; a decrypted value larger than $N/2$ is interpreted as negative. Following Damgård et al. [58], we generate the randomiser as $r^N := h_N^{\alpha}$, where $h_N = (-x^2)^N \bmod N^2$ is fixed and α is a fresh $\lceil \log_2 N \rceil/2$-bit exponent; with a fixed-base precomputation table this makes encryption about ten times faster. Because the randomiser does not depend on the message, it can be computed before a query arrives. We report this offline cost separately from the online query latency.

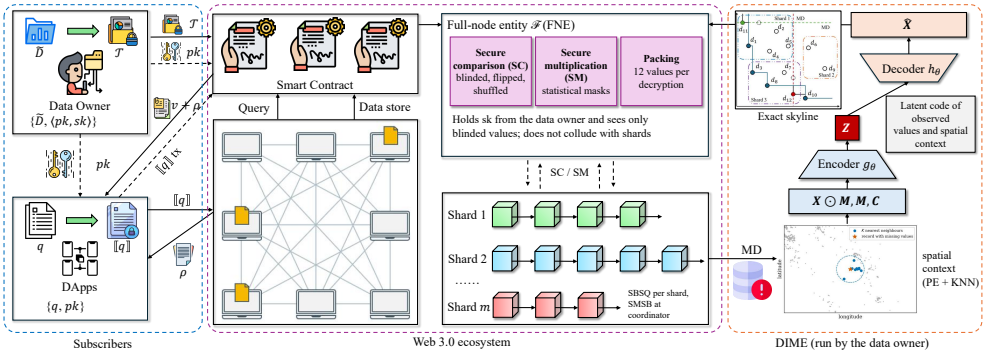


Fig. 2: System model. On the left, the data owner holds the incomplete data $\widetilde{\mathcal{D}}$ and the key pair $\langle pk, sk\rangle$ and publishes the encrypted ES-Trees $\mathcal{T}$, while an authorised DApp encrypts its query location as $[\![q]\!]$ and recovers the result from the masked values $v+\rho$ and the masks ρ. In the middle, smart contracts record index digests and query transactions, the peer-to-peer storage network holds the encrypted indexes, the shards run SBSQ on their ES-Trees and the coordinator merges the local skylines with SMSB. The full-node entity $\mathcal{F}$ holds sk and assists with secure comparison, secure multiplication and packing, but only sees blinded values. On the right, DIME is run by the data owner before encryption and repairs records with missing dimensions from their observed values and their spatial context.

3.4 Problem Statement

Given an incomplete dataset $\widetilde{\mathcal{D}}$ partitioned over m shards and an encrypted query $[\![q]\!]$ issued by an authorised DApp, the goal is to return to the DApp an approximation R of $\mathrm{SKY}_q(\mathcal{D})$ in the sense of Definition 3. Three requirements apply. First, R must equal $\mathrm{SKY}_q(\widehat{\mathcal{D}})$ exactly, where $\widehat{\mathcal{D}}$ is the imputed dataset. Second, the shards and the key holder may learn nothing beyond the well-defined leakage specified in Section 8. Third, the query latency must scale with the number of shards and the dataset size.

4 System Model and Threat Model

4.1 Entities

Figure 2 shows the five types of entities in the system.

- **Data owner.** The data owner (DO) collects the records and repairs missing values with DIME as described in Section 5. It partitions the repaired dataset $\widehat{\mathcal{D}}$ by region into $\mathcal{D}_1, \ldots, \mathcal{D}_m$, builds and encrypts an ES-Tree $\mathcal{T}_i$ for every region following Section 6, and generates the Paillier key pair (pk, sk). It publishes pk, sends sk to the full-node entity, uploads each $\mathcal{T}_i$ to distributed storage and records a digest of each $\mathcal{T}_i$ on chain, so that shards can verify the integrity of the index they retrieve.

- **DApp.** An authorised DApp acts as the query user. It encrypts its query location as $[\![q]\!] = ([\![q.x]\!], [\![q.y]\!])$ under pk, submits it to a shard and finally reconstructs the plaintext skyline from masked values.
- **Shards $\mathcal{S}_1, \ldots, \mathcal{S}_m$.** Each shard is a committee of blockchain nodes responsible for one region and executes the secure query protocol on its encrypted index. The shard that receives a query acts as the coordinator: it forwards the query to the other shards involved and runs the merge.
- **Full-node entity $\mathcal{F}$.** $\mathcal{F}$ holds sk and assists the shards in the interactive sub-protocols. In a deployment it can be operated by an independent consortium member or inside a trusted execution environment, and it can be replicated so that every shard has its own instance.
- **Distributed storage.** The storage layer keeps the encrypted indexes and serves node requests from shards, as common Web 3.0 storage layers do. It is assumed to return correct data, because its responses are checked against the on-chain digests.

4.2 Workflow

The blockchain is used for coordination and integrity: index digests, query submissions and result digests are recorded as transactions. The cryptographic computation is performed off chain by the shard committees, because Paillier operations on 2048-bit moduli are far too expensive to execute inside a smart contract. A query proceeds in four steps. The DApp first submits $[\![q]\!]$ to a coordinator shard. Every relevant shard $\mathcal{S}_i$ then runs SBSQ on $\mathcal{T}_i$ together with $\mathcal{F}$ and obtains an encrypted local skyline R_i. Next, the coordinator runs SMSB on $R_1, \ldots, R_m$ with $\mathcal{F}$ and obtains the encrypted global skyline R. Finally, the coordinator and $\mathcal{F}$ deliver R to the DApp by masked decryption.

4.3 Threat Model

We adopt the semi-honest, or honest-but-curious, model, which is standard for secure skyline and secure kNN protocols [16, 23, 42]. Shards, $\mathcal{F}$ and the storage layer follow the protocols but try to learn as much as possible about the data, the query and the result from what they observe. We assume that $\mathcal{F}$ does not collude with any shard. This non-collusion assumption is shared by all two-party designs in which one party holds the decryption key. The DApp is authorised and is entitled to learn the query result. Integrity attacks by malicious shards, such as returning wrong results, are outside the scope of this paper. They can be addressed with verifiable query processing [9, 27], which is complementary to our design.

4.4 Design Goals

- **Robustness:** the returned skyline should be close to the skyline of the complete data, even when a substantial fraction of the attribute values is missing.
- **Exactness:** the secure protocols must return exactly $\mathrm{SKY}_q(\widehat{\mathcal{D}})$, so cryptography must not introduce approximation errors.
- **Confidentiality:** data privacy means that neither the shards nor $\mathcal{F}$ learn the records, query privacy means that nobody except the DApp learns q, and result

privacy means that only the DApp learns the attribute values of the result. Any information revealed to shards or $\mathcal{F}$ must be captured by an explicit leakage function, which Section 8 defines.

- **Efficiency and scalability:** the cost should grow with the number of index entries visited rather than with the dataset size, and it should decrease as shards work in parallel.

5 Dimensional Imputation and Enhancement Network

DIME repairs the missing attribute values before the data are encrypted. Imputation has to happen on plaintext: once a value is encrypted, a learned model can no longer be evaluated on it at reasonable cost. DIME is therefore run by the DO, which owns the plaintext data. Its design rests on two observations. First, a model that is trained to reconstruct placeholders for missing values learns to reproduce the placeholders, so the training signal must come from observed values only. Second, the records in our setting are geo-referenced, and by Tobler's first law of geography [57] nearby records tend to have similar attribute values. Generic imputers treat records as exchangeable rows and miss this signal.

5.1 Input Representation

Let $\widetilde{\mathcal{D}}$ be normalised to $[0,1]$ column-wise and let $m_i \in \{0,1\}^d$ be the observation indicator of record i, so that $m_{ij} = 1 - M_{ij}$. The encoder receives three groups of features.

- **Observed values with mask.** $\tilde{x}_i \odot m_i$ together with m_i, so that the network can tell a true zero from a missing value.
- **Positional encoding.** Following Tancik et al. [59], the location is mapped to $\phi(x,y) = [\sin(2^f\pi x), \cos(2^f\pi x), \sin(2^f\pi y), \cos(2^f\pi y)]_{f=0}^{F-1}$ with $F = 6$ frequencies, which lets a small network represent spatial variation at several scales.
- **Neighbourhood statistics.** For each record, the $K = 10$ nearest records by location are retrieved with a k-d tree. For every attribute j, DIME computes the mean of the observed values among these neighbours, $\bar{a}_{ij} = \sum_{t\in\mathcal{N}(i)} m_{tj}\tilde{x}_{tj} / \sum_{t\in\mathcal{N}(i)} m_{tj}$, and the fraction of neighbours that observe it, $\rho_{ij} = \sum_{t\in\mathcal{N}(i)} m_{tj}/K$. If no neighbour observes attribute j, the column mean is used instead.

We denote the spatial context $c_i = [\phi(x_i, y_i), \bar{a}_{i\cdot}, \rho_{i\cdot}]$. The context is computed from observed values only and can therefore be used both during training and during inference.

5.2 Architecture

DIME is an autoencoder [60] with a context-conditioned decoder:

$$z_i = g_\theta\big([\tilde{x}_i \odot m_i,\ m_i,\ c_i]\big), \qquad \hat{x}_i = h_\theta\big([z_i,\ c_i]\big), \tag{3}$$

Algorithm 1 DIME training and imputation, run by the DO

Input: normalised incomplete dataset $\widetilde{\mathcal{D}}$, mask M, corruption rate π
Output: repaired dataset $\widehat{\mathcal{D}}$

split the observed attribute entries into training entries m^{tr} and 10% validation entries
compute c_i for every record from the training entries
repeat
  for each mini-batch B **do**
    $\tilde{m}_i \leftarrow m_i^{tr} \odot \text{Bernoulli}(1-\pi)$ for attributes, $\tilde{m}_{i,\{x,y\}} \leftarrow 1$, for $i \in B$
    update θ with the gradient of Eq. (4) restricted to B
  end for
until the validation error has not improved for 25 epochs
recompute c_i from all observed entries; $\hat{x}_i \leftarrow h_\theta([g_\theta([\tilde{x}_i \odot m_i, m_i, c_i]), c_i])$
return $\widehat{\mathcal{D}}$ with $\widehat{\mathcal{D}}_{ij} = \tilde{x}_{ij}$ if $m_{ij} = 1$ and $\text{clip}(\hat{x}_{ij}, 0, 1)$ otherwise

where g_θ and h_θ are three-layer perceptrons with GELU activations, a hidden width of 128 and a latent dimension of $\max(4, d)$. Feeding the context to the decoder as well as to the encoder lets the latent code concentrate on the information carried by the record's own observed values.

5.3 Training Objective

Training follows the denoising principle [55], adapted to incomplete data. In every mini-batch, each observed attribute entry is additionally hidden from the encoder with probability $\pi = 0.2$; let $\tilde{m}_i \leq m_i$ be the resulting corrupted mask. The loss is computed on all observed entries, including the ones that were hidden from the input:

$$\mathcal{L}(\theta) = \frac{\sum_i \left\| m_i \odot \left(h_\theta([g_\theta([\tilde{x}_i \odot \tilde{m}_i, \tilde{m}_i, c_i]), c_i]) - \tilde{x}_i \right) \right\|_2^2}{\sum_i \|m_i\|_1}. \tag{4}$$

The entries that are observed but hidden teach the network to predict unseen values from the observed ones and from the spatial context, and restricting the loss to observed entries prevents it from fitting placeholders. Locations are never hidden. We hold out 10% of the observed attribute entries as a validation set and optimise Eq. (4) with Adam [61], using a learning rate of 2×10^{-3} that is halved whenever the validation error plateaus and a batch size of 256. Training stops early when the validation error has not improved for 25 epochs. Algorithm 1 summarises the procedure.

5.4 Complexity

Computing the neighbourhood statistics takes $O(n \log n)$ time with a k-d tree. One training epoch costs $O(n\,(d + |c|)\,H + nH^2)$ for hidden width H. Inference is a single forward pass per record, so new records can be imputed incrementally without retraining. In our experiments, training on 5,000 records took between a few seconds and a

few tens of seconds on a laptop CPU. Because imputation runs once per data release and outside the query path, it adds no latency to queries.

6 Encrypted Shard-Tree and Secure Primitives

6.1 ES-Tree Construction

For every region $\mathcal{D}_i$, the DO bulk-loads an R-tree [36] over all d dimensions with the Sort-Tile-Recursive algorithm, STR for short [62], using node capacity c. It then encrypts the tree entry by entry as shown in Algorithm 2.

- A data entry for record p stores $[\![p.x]\!], [\![p.y]\!]$, the attribute ciphertexts $[\![p.a_1]\!], \ldots, [\![p.a_k]\!]$ and the precomputed encrypted attribute sum $[\![p.\sigma]\!] = [\![\sum_j p.a_j]\!]$.
- A node entry e stores its spatial corners $[\![e.\ell_x]\!], [\![e.\ell_y]\!], [\![e.u_x]\!], [\![e.u_y]\!]$, the encrypted lower bounds $[\![e.A_1]\!], \ldots, [\![e.A_k]\!]$ of its subtree's attributes, the encrypted sum $[\![e.\sigma]\!] = [\![\sum_j e.A_j]\!]$ and a pointer to its child node.

We call the precomputed sum $[\![e.\sigma]\!]$ the shared non-spatial aggregate and denote it by $E_{S_{NS}}$. It lets a shard form the search key $[\![\mathrm{key}_q(e)]\!] = [\![\delta_{\min}(e, q)]\!] \cdot [\![e.\sigma]\!]$ with a single homomorphic addition instead of $k - 1$.

Padding.

If nodes were only partially filled, a shard could tell nodes apart by their fan-out and use this to link queries that visit the same node. The DO therefore pads every node to exactly c entries with dummy entries. A dummy entry is located at (F, F) with $F = 2^{18}$, which lies outside the data domain, and all its attributes are set to the maximum value $2^{16} - 1$. The ciphertexts of dummy entries are indistinguishable from those of real entries.

Lemma 2 *For every query q in the data domain, every real record dominates every dummy entry.*

Proof Every real coordinate and every query coordinate lies in $[0, 2^{16})$, so $\delta(p, q) < 2^{33} < 2(F - 2^{16})^2 \leq \delta_{\min}(\mathrm{dummy}, q)$. Every real attribute is at most the dummy's attribute value. Hence the distance inequality is strict and all attribute inequalities hold. □

By Lemma 2, dummy entries never enter a skyline, so padding does not affect correctness; Section 9.5 quantifies its cost. A tree over n_i records with capacity c has $O(n_i/c)$ nodes. Each data entry carries $3 + k$ ciphertexts and each node entry $5 + k$.

6.2 Secure Sub-Protocols

All sub-protocols run between a shard $\mathcal{S}$, which holds ciphertexts and pk, and $\mathcal{F}$, which holds sk. They are batched, so that one call processes a list of inputs in a single round trip.

Algorithm 2 ES-Tree construction for shard $\mathcal{S}_i$, run by the DO

Input: repaired integer region $\widehat{\mathcal{D}}_i$, capacity c, public key pk
Output: encrypted index $\mathcal{T}_i$

build an STR R-tree over $\widehat{\mathcal{D}}_i$ in all d dimensions with capacity c
for each node v in the tree **do**
  for each entry e of v **do**
    if e is a data entry for p **then**
      $e^\star \leftarrow (\llbracket p.x \rrbracket, \llbracket p.y \rrbracket, \llbracket p.a_1 \rrbracket, \ldots, \llbracket p.a_k \rrbracket, \llbracket \sum_j p.a_j \rrbracket)$
    else
      $e^\star \leftarrow (\llbracket e.\ell_x \rrbracket, \llbracket e.\ell_y \rrbracket, \llbracket e.u_x \rrbracket, \llbracket e.u_y \rrbracket, \llbracket e.A_1 \rrbracket, \ldots, \llbracket e.A_k \rrbracket, \llbracket \sum_j e.A_j \rrbracket, ptr)$
    end if
  end for
  pad v to c entries with encrypted dummy entries, see Lemma 2
end for
upload $\mathcal{T}_i$ to distributed storage; record digest($\mathcal{T}_i$) on chain

Secure comparison.

The secure comparison protocol SC takes pairs $(\llbracket a \rrbracket, \llbracket b \rrbracket)$ and returns $[a \le b]$, either in plaintext to $\mathcal{S}$ or encrypted, in which case nobody learns it. For each pair, $\mathcal{S}$ draws a random bit f, an odd κ-bit integer r and $r' \in [0, r)$, and computes

$$\llbracket v \rrbracket = \begin{cases} \llbracket a-b \rrbracket^{2r} \cdot \llbracket r'-r \rrbracket = \llbracket r(2(a-b)-1)+r' \rrbracket & f=0, \\ \llbracket b-a \rrbracket^{2r} \cdot \llbracket r'+r \rrbracket = \llbracket r(2(b-a)+1)+r' \rrbracket & f=1. \end{cases} \tag{5}$$

Since $2(a-b)-1$ is odd and $0 \le r' < r$, the sign of v equals the sign of $2(a-b)-1$ when $f=0$ and its opposite when $f=1$, and v is never zero. $\mathcal{S}$ randomly permutes the batch and sends it to $\mathcal{F}$. $\mathcal{F}$ decrypts each v, determines $s = [v < 0]$, and returns s in plaintext, or $\llbracket s \rrbracket$ in encrypted mode. $\mathcal{S}$ undoes the permutation and the flip: $[a \le b] = s \oplus f$, which in encrypted mode is computed homomorphically as $\llbracket s \rrbracket$ or $\llbracket 1 \rrbracket \cdot \llbracket s \rrbracket^{-1}$. Because f is uniform and secret, the sign that $\mathcal{F}$ observes is a uniformly random bit.

Secure multiplication.

For the secure multiplication protocol SM, given pairs $(\llbracket a \rrbracket, \llbracket b \rrbracket)$ with $|a|, |b| < 2^\ell$, $\mathcal{S}$ draws masks r_a, r_b uniformly from $[0, 2^{\ell+\kappa})$ and sends $\llbracket a + r_a \rrbracket, \llbracket b + r_b \rrbracket$. $\mathcal{F}$ decrypts them and returns $\llbracket (a+r_a)(b+r_b) \rrbracket$. $\mathcal{S}$ then computes $\llbracket ab \rrbracket = \llbracket (a+r_a)(b+r_b) \rrbracket \cdot \llbracket a \rrbracket^{-r_b} \cdot \llbracket b \rrbracket^{-r_a} \cdot \llbracket -r_a r_b \rrbracket$. This is the masking technique of [42], with statistical rather than perfect masks, which keeps all exponents short.

Plaintext packing.

The values sent to $\mathcal{F}$ by SC and SM are much shorter than the plaintext space. $\mathcal{S}$ therefore packs them into slots of $w = \ell + \kappa + 3$ bits by Horner's rule, $\llbracket V \rrbracket = \prod_t \llbracket v_t + 2^{w-2} \rrbracket^{2^{wt}}$. The offset 2^{w-2} keeps every slot non-negative, so a single decryption

Algorithm 3 SSD, batched over candidates

Input: candidates E and, for each $e \in E$, a set S_e of skyline records, all encrypted
Output: for each e, the bit $\Phi_e = [\exists s \in S_e : s \text{ dominates every record in } e]$, revealed to $\mathcal{S}$

1: for every e and $s \in S_e$, form the $k+2$ pairs $(\delta(s,q), \delta_{\min}(e,q))$, $(s.a_j, e.A_j)_{j=1}^{k}$, $(\mathrm{key}_q(s)+1, \mathrm{key}_q(e))$
2: $[\![b_{s,e,t}]\!] \leftarrow \mathsf{SC}(\text{all pairs}, \textit{encrypted})$ ▷ round 1
3: $[\![\beta_{s,e}]\!] \leftarrow \prod_t [\![b_{s,e,t}]\!]$ ▷ number of satisfied conditions
4: $[\![\varphi_{s,e}]\!] \leftarrow \mathsf{SC}\big(([\![k+2]\!], [\![\beta_{s,e}]\!]), \textit{encrypted}\big)$ ▷ round 2: all $k+2$ satisfied
5: $[\![\Sigma_e]\!] \leftarrow \prod_{s \in S_e} [\![\varphi_{s,e}]\!]$
6: $\Phi_e \leftarrow \mathsf{SC}\big(([\![1]\!], [\![\Sigma_e]\!]), \textit{plaintext}\big)$ ▷ round 3: only this bit is revealed
7: **return** $\{\Phi_e\}_{e \in E}$

by $\mathcal{F}$ recovers $\lfloor (|N|-2)/w \rfloor$ values, which is 12 values for $|N| = 1024$ and $\ell = \kappa = 40$. As Section 9.5 shows, packing reduces the key holder's decryption work and the uplink traffic substantially.

6.3 Secure Shard-Based Dominance

The secure shard-based dominance protocol SSD decides, for a candidate entry e and a set S of current skyline records, whether some $s \in S$ dominates every record represented by e. All inputs are encrypted: $[\![\delta(s,q)]\!]$, $[\![s.a_j]\!]$ and $[\![\mathrm{key}_q(s)]\!]$ for each $s \in S$, and $[\![\delta_{\min}(e,q)]\!]$, $[\![e.A_j]\!]$ and $[\![\mathrm{key}_q(e)]\!]$ for the candidate. For a data entry, $e.A_j = p.a_j$ and $\delta_{\min} = \delta$.

Lemma 3 (Correctness of SSD) *$\Phi_e = 1$ if and only if there exists $s \in S_e$ with $s \prec_q p$ for every record p in the subtree of e. For a data entry p, this means $s \prec_q p$.*

Proof $\varphi_{s,e} = 1$ if and only if $\delta(s,q) \le \delta_{\min}(e,q)$, $s.a_j \le e.A_j$ for all j, and $\mathrm{key}_q(s) < \mathrm{key}_q(e)$. By Lemma 1, the first $k+1$ conditions together with the last one are equivalent to s dominating the comparison vector $(\delta_{\min}(e,q), e.A_1, \ldots, e.A_k)$. Every record p below e satisfies $\delta(p,q) \ge \delta_{\min}(e,q)$ and $p.a_j \ge e.A_j$, so s dominates p by transitivity of the componentwise order. Conversely, a data entry is its own comparison vector. Finally, $\Phi_e = [\sum_s \varphi_{s,e} \ge 1]$. □

Only Φ_e is revealed to the shard. The individual comparison outcomes and the identity of the dominating record stay encrypted. For a batch with $|E|$ candidates and $|S|$ skyline records, SSD uses $|E||S|(k+3) + |E|$ comparisons in three rounds.

7 Secure Skyline Query Processing

7.1 Secure Distance Computation

When a shard expands a node, it computes the encrypted squared distances, or minimum squared distances, for all c entries of the node in one batch.

- *Data entries.* $[\![\delta(p,q)]\!] = \mathsf{SM}([\![p.x - q.x]\!], [\![p.x - q.x]\!]) \cdot \mathsf{SM}([\![p.y - q.y]\!], [\![p.y - q.y]\!])$, where the differences are obtained homomorphically.
- *Node entries.* For $t \in \{x, y\}$, the encrypted indicators $[\![[q.t < e.\ell_t]]\!] = \mathsf{SC}([\![q.t + 1]\!], [\![e.\ell_t]\!])$ and $[\![[q.t > e.u_t]]\!] = \mathsf{SC}([\![e.u_t + 1]\!], [\![q.t]\!])$ are computed in encrypted mode. SM then evaluates $[\![\Delta_t]\!]$ of Eq. (1), and a final SM squares it.

Because the indicators remain encrypted, the shard does not learn whether q lies inside, to the left of or to the right of a bounding rectangle. The batch takes one round for data entries and three rounds for node entries, using 2 multiplications per data entry and 4 comparisons and 6 multiplications per node entry.

7.2 Secure Blockchain-Based Skyline Query

The secure blockchain-based skyline query algorithm SBSQ, given as Algorithm 4, is a secure counterpart of BBS [35]. Shard $\mathcal{S}_i$ keeps a queue Ψ of candidate entries sorted by key_q and a list R_i of local skyline records. Each candidate also records how many skyline records it has already been checked against, so that dominance is never re-tested against the same record.

Theorem 4 (Correctness of SBSQ) *SBSQ returns exactly* $\mathrm{SKY}_q(\widehat{\mathcal{D}}_i)$.

Proof We first show soundness. By Lemma 3, a data entry p is appended to R_i only after SSD has confirmed that no record already in R_i dominates it. Suppose some record p' dominates p. By Lemma 1, $\mathrm{key}_q(p') < \mathrm{key}_q(p)$, and every ancestor entry of p' has a key no larger than $\mathrm{key}_q(p')$. The queue is processed in non-decreasing key order, and every entry inserted later has a key no smaller than the entry that was expanded to produce it. Hence p', or a record that dominates p' and therefore p, is popped and appended before p is popped, so p would have been discarded. For completeness, note that by Lemma 3 an entry is discarded only if some skyline record dominates every record below it, so no skyline record is ever discarded. By Lemma 2, dummy entries are dominated by every real record and therefore never enter R_i. □

During the search, the shard learns the outcome of the key comparisons used to order Ψ and the dominance bit Φ_e of every candidate. All distances, attribute values and the query remain encrypted.

7.3 Skyline Merging for Sharded Blockchains

After each shard has computed its local skyline, the coordinator merges the local results. Merging relies on the following property of skylines over a partition [21, 22].

Lemma 5 (Mergeability) *For a partition $\widehat{\mathcal{D}} = \widehat{\mathcal{D}}_1 \cup \cdots \cup \widehat{\mathcal{D}}_m$, a record $p \in \mathrm{SKY}_q(\widehat{\mathcal{D}}_i)$ belongs to $\mathrm{SKY}_q(\widehat{\mathcal{D}})$ if and only if no record of $\bigcup_{j \neq i} \mathrm{SKY}_q(\widehat{\mathcal{D}}_j)$ dominates p. Consequently* $\mathrm{SKY}_q(\widehat{\mathcal{D}}) = \mathrm{SKY}_q\big(\bigcup_i \mathrm{SKY}_q(\widehat{\mathcal{D}}_i)\big)$.

Algorithm 4 SBSQ on shard $\mathcal{S}_i$

Input: encrypted index $\mathcal{T}_i$, encrypted query $[\![q]\!]$
Output: encrypted local skyline R_i

$R_i \leftarrow \emptyset$; $\Psi \leftarrow$ INSERT$(\emptyset,$ EXPAND$(\text{root}(\mathcal{T}_i)))$
while $\Psi \neq \emptyset$ **do**
  $e \leftarrow$ pop the first entry of Ψ
  if e has not been checked against all of R_i **and** $\mathsf{SSD}(e,$ unchecked part of $R_i) = 1$ **then**
    continue ▷ e is dominated
  end if
  if e is a data entry **then**
    append e to R_i
  else
    $\Psi \leftarrow$ INSERT$(\Psi,$ EXPAND$(e))$
  end if
end while
return R_i

function EXPAND(v) ▷ fetch child node v from storage
  compute $[\![\delta_{\min}(e,q)]\!]$ for all entries e of v as in Section 7.1; $[\![\text{key}_q(e)]\!] \leftarrow [\![\delta_{\min}(e,q)]\!] \cdot [\![e.\sigma]\!]$
  if $R_i \neq \emptyset$ **then** remove every e with $\mathsf{SSD}(e, R_i) = 1$ ▷ insertion-time pruning
  end if
  return the remaining entries
end function
function INSERT$(\Psi,\ C)$
  sort C with one batched round of pairwise plaintext-mode SC on keys
  locate every $e \in C$ in Ψ by parallel binary search with batched SC in $\lceil \log_2(|\Psi|+1) \rceil$ rounds
  return the merged queue
end function

Proof If some $p' \in \widehat{\mathcal{D}}_j$ dominates p, then either $p' \in \text{SKY}_q(\widehat{\mathcal{D}}_j)$, or some $p'' \in \text{SKY}_q(\widehat{\mathcal{D}}_j)$ dominates p' and hence, by transitivity of dominance over complete records, also p. Records in $\widehat{\mathcal{D}}_i$ cannot dominate p because p is in the local skyline. The converse is immediate. □

Lemma 5 turns the merge into a single batched dominance test. For every local skyline record p of shard i, the coordinator calls $\mathsf{SSD}(p, \bigcup_{j \neq i} R_j)$ and keeps p if the result is 0. All tests are independent, so the whole merge takes three rounds regardless of the number of shards and uses $\sum_i |R_i|(u - |R_i|)(k+3)$ comparisons, where $u = \sum_i |R_i|$. This procedure is the skyline merging algorithm for sharded blockchains, SMSB. Unlike iterative merging based on repeated secure minimum computation, it does not need to order the union of the local skylines. The merge relies on the transitivity argument in Lemma 5, and this is where imputation pays off a second

time: over incomplete records dominance is not transitive [10], so the lemma would no longer hold.

7.4 Result Retrieval

For each record p of the global skyline and each of its d encrypted values $[\![v]\!]$, the coordinator draws ρ uniformly from $\mathbb{Z}_N$ and sends $[\![v+\rho]\!]$ to $\mathcal{F}$ and ρ to the DApp over a secure channel. $\mathcal{F}$ decrypts $v+\rho$ and forwards it to the DApp, which outputs $v = (v+\rho)-\rho \bmod N$. $\mathcal{F}$ observes uniformly random values, the coordinator observes only ciphertexts, and only the DApp learns p.

7.5 Complexity Analysis

Let V_i be the number of entries that shard $\mathcal{S}_i$ visits, $s_i = |R_i|$ and $\bar{\psi}$ the average queue length.

- *Computation.* Distance computation costs $O(V_i)$ calls to SC and SM. Insertion-time pruning and pop-time dominance tests cost $O(V_i s_i (k+3))$ comparisons in the worst case, and queue maintenance costs $O(V_i(c+\log\bar{\psi}))$ comparisons. Each comparison costs the shard $O(1)$ modular exponentiations with κ-bit exponents. It costs $\mathcal{F}$ one twelfth of a decryption and, in encrypted mode, one encryption. SMSB costs $O(u^2(k+3))$ comparisons.
- *Communication.* Each comparison sends one twelfth of a ciphertext upstream and, in encrypted mode, one ciphertext downstream. Each expansion needs $O(1+\log\bar{\psi})$ rounds, and SMSB needs three rounds.
- *Parallelism.* Shards run concurrently, so the query latency is $\max_i T(\mathsf{SBSQ}_i) + T(\mathsf{SMSB})$ rather than the sum of the per-shard times.

For comparison, a secure scan without an index, which we call SecScan, costs $O(n\,|W|(k+3))$ comparisons for a window W, and a fully oblivious protocol following Liu et al. [16] costs $O(n\,|\mathrm{SKY}|(k+3))$ comparisons and $O(n\,|\mathrm{SKY}|\,d)$ multiplications, because every iteration touches every record. In SBSQ the cost is governed by V_i, which the branch-and-bound order keeps far below n_i.

8 Security Analysis

We follow the simulation paradigm for semi-honest adversaries [63, 64] with explicit leakage functions, as is customary for practical encrypted query processing [65]. A protocol is secure with respect to leakage $\mathcal{L}$ against a party P if there is a probabilistic polynomial-time simulator that, given only the input and output of P and the leakage $\mathcal{L}$, produces a view that is computationally indistinguishable from the real view of P.

Definition 4 (Leakage-based security) Let $\mathrm{View}_P(\mathcal{D}, q)$ be the view of party P in a real execution. The scheme is $\mathcal{L}_P$-secure against a semi-honest P if there exists a PPT simulator Sim_P such that for all $\mathcal{D}, q$,

$$\left|\Pr[\mathcal{A}(\mathrm{View}_P(\mathcal{D},q))=1] - \Pr[\mathcal{A}(\mathrm{Sim}_P(\mathcal{L}_P(\mathcal{D},q)))=1]\right| \le \mathrm{negl}(\lambda) \tag{6}$$

for every PPT distinguisher $\mathcal{A}$, where λ is the security parameter.

8.1 Leakage Functions

Leakage to a shard.

The leakage $\mathcal{L}_{\mathcal{S}_i}(\mathcal{D}, q)$ to shard $\mathcal{S}_i$ consists of five items:

- the size information n_i and c together with the shape of $\mathcal{T}_i$, which is determined by n_i and c because nodes are padded;
- the access pattern, that is, the sequence of tree nodes that are expanded;
- the outcomes of the key comparisons used to maintain the queue, which reveal the relative order of the search keys of visited entries;
- the dominance bit Φ_e of every candidate;
- the positions of the entries that form the local skyline.

For the coordinator, the leakage additionally includes the sizes $|R_j|$ of the local skylines and, for each local skyline record, the bit that tells whether it survives the merge.

Leakage to $\mathcal{F}$.

The leakage $\mathcal{L}_{\mathcal{F}}(\mathcal{D}, q)$ consists of the number of sub-protocol calls and their batch sizes. It also contains, for each comparison, the absolute difference $|a - b|$ of the compared values, given as an unordered multiset per batch that cannot be linked to records or attributes. This item is an upper bound on what $\mathcal{F}$ actually observes, which is only the blinded magnitude $r\,|2(a - b) \mp 1| + r'$ for a secret random r.

8.2 Security Against the Full-Node Entity

Theorem 6 *Under the DCR assumption, the protocol suite consisting of SBSQ, SMSB and result retrieval is $\mathcal{L}_{\mathcal{F}}$-secure against a semi-honest $\mathcal{F}$ that does not collude with any shard.*

Proof sketch $\mathcal{F}$ receives four kinds of messages. In SC, each decrypted slot is $v = \pm r(2(a - b) - 1) + r'$ with a secret uniform flip, so the sign of v is a uniform bit that is independent of a and b, and the batch is randomly permuted. Given the multiset of values $|a - b|$ from $\mathcal{L}_{\mathcal{F}}$, the simulator samples fresh f, r and r', forms the same distribution of slot values, permutes them uniformly and packs them. In SM, each decrypted value is $a + r_a$ with r_a uniform in $[0, 2^{\ell+\kappa})$ and $|a| < 2^{\ell}$, so its distribution is within statistical distance $2^{-\kappa}$ of the uniform distribution on that interval, and the simulator samples uniform values. In result retrieval, $\mathcal{F}$ decrypts $v + \rho$ with ρ uniform in $\mathbb{Z}_N$, which is perfectly uniform. Every other message that $\mathcal{F}$ receives is a packed ciphertext of such values, whose plaintext the simulator already knows, and the messages that $\mathcal{F}$ sends are fresh encryptions that it computes itself. A hybrid argument over the polynomially many messages bounds the distinguishing advantage by a negligible function of λ plus a polynomial multiple of $2^{-\kappa}$. □

8.3 Security Against Shards

Theorem 7 *Under the DCR assumption, the protocol suite is $\mathcal{L}_{\mathcal{S}_i}$-secure against a semi-honest shard $\mathcal{S}_i$, including the coordinator, that does not collude with $\mathcal{F}$.*

Proof sketch A shard never holds sk. Its view consists of three parts: the ciphertexts of the index, of the query and of every encrypted output returned by $\mathcal{F}$; the plaintext bits returned by SC in plaintext mode; and the access pattern on distributed storage. The simulator replaces every ciphertext by an encryption of 0, which is indistinguishable by the semantic security of Paillier with the DJN randomiser [58] under DCR. It produces the plaintext bits from the comparison outcomes, dominance bits and skyline positions contained in $\mathcal{L}_{\mathcal{S}_i}$. Bits that the shard un-flips locally remain consistent, because the simulator knows the outputs and samples the flips itself. The access pattern is generated from the size, shape and access-pattern items of $\mathcal{L}_{\mathcal{S}_i}$. Each replacement changes the view by a negligible amount, and the shard's own randomness is sampled honestly. □

8.4 Discussion of the Leakage

What is protected.

By Theorems 6 and 7, neither a shard nor $\mathcal{F}$ learns any record value, the query location or any value of the result. In contrast to designs based on order-revealing encryption [14, 44], a shard cannot compare two arbitrary stored values. It learns only the order of query-dependent keys of the entries it visits, and these keys combine the encrypted distance to an unknown query with the attribute sum.

Access pattern.

Like most efficient index-based secure query schemes, SBSQ reveals which nodes are expanded. Padding removes the fan-out side channel, but repeated queries from nearby locations still tend to visit overlapping nodes, and access-pattern leakage can enable inference attacks when an adversary has auxiliary knowledge [66]. Applications that need complete access-pattern hiding can run the fully oblivious variant evaluated in Section 9.4, which was 11 to 25 times slower in our experiments with a gap that widens as n grows, or they can place the index in an oblivious RAM [67]. Our evaluation quantifies this trade-off between privacy and efficiency.

Magnitude leakage to $\mathcal{F}$.

Multiplicative blinding hides the sign and the identity of each compared pair, but it reveals the rough magnitude of $|a - b|$ to $\mathcal{F}$. Replacing SC with a bitwise comparison protocol such as DGK [68] would remove this leakage at a higher cost.

Collusion.

If $\mathcal{F}$ colludes with a shard, the shard obtains the decryption key and all confidentiality is lost. This limitation is shared by all two-party designs with a single key holder. It can be mitigated by distributing sk among several $\mathcal{F}$ replicas with threshold Paillier decryption, so that no single node can decrypt.

9 Experimental Evaluation

The evaluation answers four questions. Section 9.2 examines whether DIME recovers missing values, and with them the skyline, more accurately than existing imputers and than skyline semantics for incomplete data. Section 9.3 analyses which components of

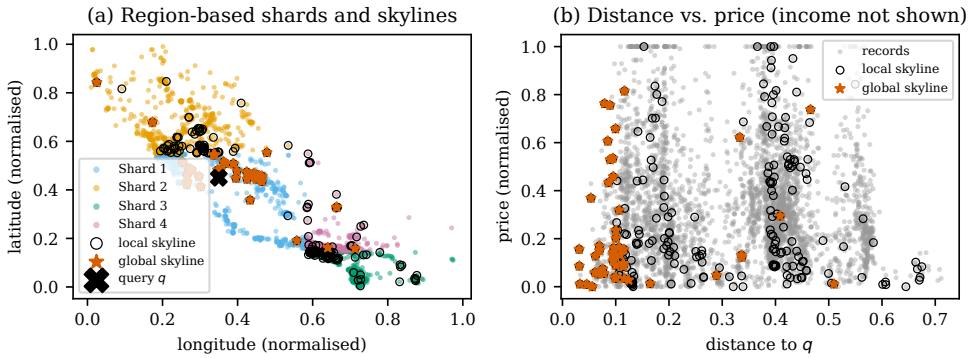


Fig. 3: California data partitioned into four region-based shards, shown for 4,000 sampled records with price and income as the $k = 2$ attributes. Panel (a) shows the locations, the query q, the local skylines of the shards as circles and the global skyline as stars. Panel (b) shows the same records in the space of distance to q and price. Records that are optimal within their shard but not globally are removed by SMSB.

DIME matter and how sensitive it is to its hyper-parameters. Section 9.4 compares SBSQ and SMSB with secure baselines and studies how they scale with the data size, the number of attributes, the number of shards and the index parameters. Section 9.5 measures what the individual protocol optimisations contribute.

9.1 Experimental Setup

Datasets.

Table 3 summarises the datasets. The California dataset is the California Housing data [69], which describes 20,640 census block groups with their longitude and latitude. We use five of its attributes: the median house value and the average occupancy, where lower values are better, the median income and the average number of rooms, where higher values are better, and the house age, where lower is better. The NYC Airbnb dataset contains the 29,918 New York City listings of the Inside Airbnb snapshot of 14 September 2026 [70]. Its attributes are the nightly price and the minimum number of nights, where lower is better, and three review scores for rating, location and value, where higher is better. In the raw data 29.1% of these attribute values are missing. The controlled experiments use the 14,337 complete listings with injected missing values, and Section 9.2.1 uses the raw data with its natural missing values. IND, COR and ANT are the standard independent, correlated and anti-correlated skyline benchmarks [7], here with uniformly distributed locations. Figure 3 shows the California data partitioned into four region-based shards together with the local and global skylines of one query. The skewed attributes, namely price, occupancy, rooms and minimum nights, are log-transformed. All attributes are clipped at their 1st and 99th percentiles, and all values are min–max scaled and quantised to $[0, 2^{16})$.

Table 3: Datasets. The column k gives the number of non-spatial attributes used by default; the real datasets provide up to five.

Dataset	Records	k	Missing values	Source
California	20,640	3	injected	census block groups [69]
NYC Airbnb, complete	14,337	3	injected	Inside Airbnb [70]
NYC Airbnb, raw	29,918	4	natural, 29.1%	Inside Airbnb [70]
IND, COR, ANT	2,000–50,000	1–4	injected	synthetic [7]

Missing values.

Missing values are injected into the attributes, never into the locations, at rates between 10% and 50%, with 30% as the default. We use the three mechanisms defined in Section 3. MCAR is the default. Under MAR the probability that a value is missing increases with the longitude of the record, and under MNAR it increases with the value itself.

Imputation baselines.

We compare DIME with six imputers: mean imputation; kNN imputation [49] with $k = 5$; MICE [50] with Bayesian ridge regression and 10 iterations; MissForest [51] with random forests of 30 trees of maximum depth 12 and 5 iterations; GAIN [56] with the default hyper-parameters of its authors; and MIDA [54]. For fairness, MIDA uses the same validation-based early stopping as DIME. We also evaluate the incomplete-data skyline semantics of Khalefa et al. [10], referred to as ISkyline, which computes the skyline directly on the incomplete data without imputation.

Query baselines.

SecScan uses the same cryptographic primitives as our scheme but no index. It processes all encrypted records with a secure block-nested loop and thereby isolates the benefit of the ES-Tree. The oblivious baseline re-implements the protocol structure of Liu et al. [16] on our primitives: it repeatedly selects the secure minimum obliviously and then obliviously eliminates the records it dominates, which hides the access pattern. As lower bounds that offer no privacy, we also report four plaintext algorithms: BNL [7]; BBS [35]; DSQ, a distributed skyline that merges local skylines [21, 22]; and GNN, which gathers the data of all shards before running BNL.

Metrics.

Imputation quality is measured by the RMSE on the injected missing cells in normalised units. Skyline quality is measured by precision, recall and F_1 with respect to the skyline of the complete data, as in Definition 3, averaged over 20 query locations per run. Each accuracy configuration is repeated with 5 random seeds, which change both the sample and the missing-value mask, and we report the mean and the standard deviation. For the secure protocols we report the online time, the number of interaction rounds, the communication volume and the index size. The online time is the wall-clock time minus the time spent on generating randomisers, which can be

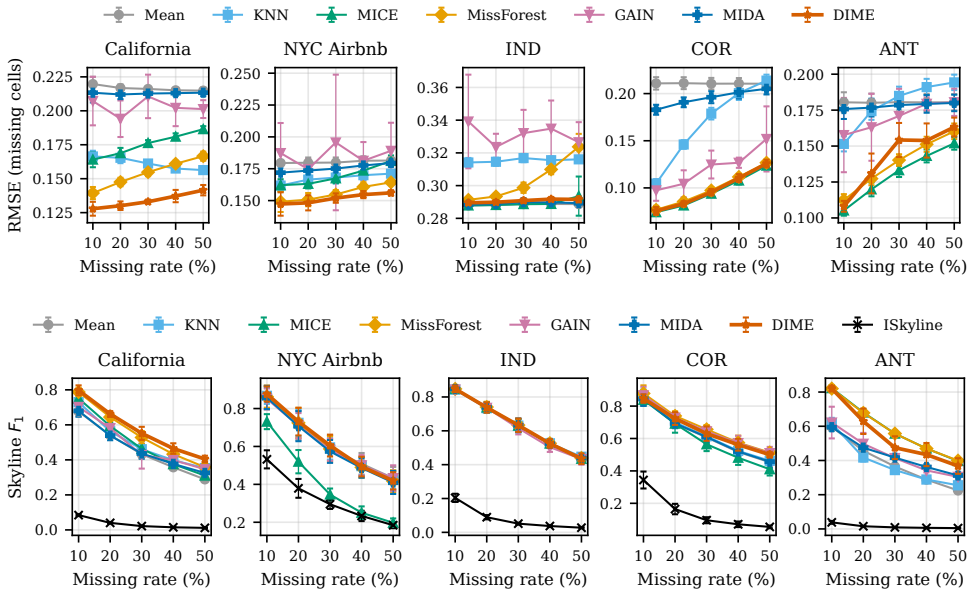


Fig. 4: Imputation RMSE in the top row, where lower is better, and skyline F_1 in the bottom row, where higher is better, versus the missing rate under MCAR. Error bars show one standard deviation over 5 seeds.

precomputed. Each efficiency configuration is averaged over 3 query locations. With m shards, the latency is the maximum time over the shards plus the time of the merge.

Implementation.

The system is implemented in Python 3.11. Paillier encryption with CRT decryption and the DJN randomiser is built on gmpy2, DIME and GAIN use PyTorch [71], and the other imputers use scikit-learn [72]. Unless stated otherwise, the efficiency experiments use $|N| = 1024$, $\kappa = 40$, $c = 8$, $m = 4$, $k = 2$ and $n = 10{,}000$, and the accuracy experiments use $k = 3$ and $n = 5{,}000$. All experiments ran on a laptop with a four-core Intel Core i7-1165G7 CPU at 2.8 GHz and 16 GB of RAM under Windows 11. Shards and $\mathcal{F}$ run as single-threaded code on the same machine, and several configurations run concurrently, so the absolute times are conservative. Every secure query in every experiment returned exactly the plaintext skyline of the imputed data, which was checked automatically.

9.2 Imputation and Skyline Accuracy

Figure 4 and Table 4 report the results under MCAR.

Real data.

DIME attains the lowest RMSE on both real datasets at every missing rate. On California it reduces the RMSE relative to MissForest, the strongest baseline, by 8%

Table 4: Accuracy at 30% MCAR missing values as mean ± standard deviation over 5 seeds. Best values are in bold.

Method	California	NYC Airbnb	IND	COR	ANT
RMSE ↓					
Mean	0.216 ± 0.002	0.180 ± 0.004	**0.289 ± 0.002**	0.211 ± 0.006	0.181 ± 0.006
KNN	0.161 ± 0.002	0.168 ± 0.005	0.317 ± 0.003	0.179 ± 0.006	0.185 ± 0.006
MICE	0.176 ± 0.002	0.167 ± 0.004	**0.289 ± 0.002**	**0.094 ± 0.003**	**0.133 ± 0.005**
MissForest	0.155 ± 0.003	0.155 ± 0.004	0.299 ± 0.003	0.097 ± 0.003	0.140 ± 0.004
GAIN	0.211 ± 0.016	0.196 ± 0.053	0.332 ± 0.014	0.125 ± 0.015	0.171 ± 0.019
MIDA	0.213 ± 0.002	0.175 ± 0.004	0.290 ± 0.002	0.196 ± 0.006	0.178 ± 0.007
DIME	**0.133 ± 0.002**	**0.152 ± 0.004**	0.291 ± 0.001	0.095 ± 0.003	0.154 ± 0.012
F_1 ↑					
Mean	0.432 ± 0.023	0.587 ± 0.057	0.631 ± 0.039	0.607 ± 0.035	0.365 ± 0.015
KNN	0.461 ± 0.013	0.595 ± 0.065	0.631 ± 0.038	0.614 ± 0.036	0.344 ± 0.012
MICE	0.463 ± 0.028	0.346 ± 0.032	0.631 ± 0.038	0.564 ± 0.042	**0.557 ± 0.018**
MissForest	0.528 ± 0.037	0.587 ± 0.068	**0.634 ± 0.036**	**0.656 ± 0.027**	**0.557 ± 0.017**
GAIN	0.428 ± 0.078	0.582 ± 0.058	0.615 ± 0.037	0.644 ± 0.028	0.414 ± 0.054
MIDA	0.437 ± 0.030	0.574 ± 0.060	0.633 ± 0.041	0.609 ± 0.036	0.417 ± 0.010
DIME	**0.552 ± 0.037**	**0.605 ± 0.058**	0.633 ± 0.040	0.630 ± 0.044	0.475 ± 0.067
ISkyline	0.022 ± 0.002	0.295 ± 0.025	0.051 ± 0.006	0.096 ± 0.019	0.009 ± 0.001

at 10% missing values and by 14–15% at 30–50%; at 30% the two methods reach 0.133 and 0.155. On NYC Airbnb the reduction is smaller, between 1% and 5%. The advantage carries over to the skyline. DIME has the highest F_1 on California at every rate, for example 0.552 against 0.528 for MissForest at 30%, and on NYC Airbnb up to 30% missing values. At 40–50% missing values on NYC Airbnb, all imputers except MICE are within 0.03 F_1 of each other, and mean imputation and GAIN are marginally ahead. MICE is competitive in RMSE on NYC Airbnb but attains a much lower F_1, 0.346 against 0.605 for DIME at 30%. Its linear predictions shrink extreme values towards the mean, which changes which records are Pareto-optimal. RMSE alone is therefore not a sufficient criterion for choosing an imputer for skyline queries.

Synthetic data.

The synthetic benchmarks have no spatial structure, which is the main source of information for DIME. On IND the attributes are independent, so no method can do better than the mean. Mean imputation, MICE, MIDA and DIME are within 0.002 RMSE of each other, whereas kNN, MissForest and GAIN fit noise and reach between 0.299 and 0.332, and the skyline F_1 of all imputers lies between 0.615 and 0.634. On COR, DIME, MICE and MissForest are within 0.003 RMSE of each other, and MissForest has the best F_1. On ANT the linear dependence among the attributes is captured best by MICE, with an RMSE of 0.133, and by MissForest with 0.140. DIME comes third with 0.154, and its F_1 of 0.475 trails both. These results confirm that the advantage of DIME comes from exploiting spatial autocorrelation.

Imputation versus incomplete-data semantics.

ISkyline attains an F_1 of only 0.022 on California and 0.295 on NYC Airbnb at 30% missing values, far below every imputation method. Its precision is high, but its recall collapses: when dominance is evaluated on commonly observed attributes, records are

Table 5: RMSE and F_1 at 30% missing values under different missingness mechanisms, each cell giving RMSE / F_1 averaged over 5 seeds. Best values are in bold.

	California			NYC Airbnb		
Method	MCAR	MAR	MNAR	MCAR	MAR	MNAR
Mean	0.216 / 0.432	0.215 / 0.461	0.250 / 0.465	0.180 / 0.587	0.181 / 0.596	0.226 / 0.789
KNN	0.161 / 0.461	0.158 / 0.512	0.207 / 0.408	0.168 / 0.595	0.169 / 0.595	0.206 / 0.773
MICE	0.176 / 0.463	0.180 / 0.513	0.183 / 0.576	0.167 / 0.346	0.167 / 0.350	0.207 / 0.439
MissForest	0.155 / 0.528	0.162 / 0.550	0.159 / 0.639	0.155 / 0.587	0.158 / 0.602	**0.189** / **0.790**
GAIN	0.211 / 0.428	0.205 / 0.488	0.209 / 0.511	0.196 / 0.582	0.194 / 0.585	0.212 / 0.771
MIDA	0.213 / 0.437	0.211 / 0.480	0.243 / 0.437	0.175 / 0.574	0.176 / 0.588	0.221 / 0.761
DIME	**0.133** / **0.552**	**0.138** / **0.580**	**0.144** / **0.645**	**0.152** / **0.605**	**0.154** / **0.607**	0.190 / 0.784
ISkyline	– / 0.022	– / 0.049	– / 0.025	– / 0.295	– / 0.281	– / 0.303

eliminated through cyclic dominance chains and almost all true skyline records are lost. This supports our decision to repair the data before querying.

Missingness mechanisms.

Table 5 reports the results at 30% missing values under the three mechanisms. DIME has the lowest RMSE and the highest F_1 in five of the six settings. The exception is MNAR on NYC Airbnb, the hardest setting because the missing values are systematically the larger ones. There DIME and MissForest are tied in RMSE at 0.190 and 0.189, and MissForest and mean imputation are marginally ahead in F_1. As expected, MNAR increases the RMSE of every method, because the observed values are no longer representative of the missing ones.

9.2.1 Natural Missing Values

On the raw NYC Airbnb data with its natural missing values, the true values of the missing cells are unknown. We therefore hide a further 10% of the observed attribute values, impute a sample of 10,000 listings with $k = 4$ attributes, and score each method on the hidden cells. Table 6 shows that DIME again attains the lowest error, with an RMSE of 0.129 against 0.135 for MissForest and 0.144 for MICE. DIME is not the fastest method: it takes 38 s, compared with 25 s for MissForest and less than 1 s for MICE. Imputation, however, runs once per data release at the DO and is not on the query path.

9.3 Analysis of DIME

Ablation.

Table 7 removes one component at a time. The neighbourhood statistics are the most important spatial component: without them, the RMSE on California rises from 0.133 to 0.152. The positional encoding brings no measurable gain on its own when the neighbourhood statistics are present. Removing both spatial components, however, raises the RMSE to 0.171 on California and to 0.163 on NYC Airbnb, so the two components are partly redundant and either one recovers much of the spatial signal. The masked loss and the additional masking are essential on all datasets, including

Table 6: Imputation on the raw NYC Airbnb data, where 29.1% of the attribute values are naturally missing. Methods are scored on 10% additionally hidden observed cells with $n = 10{,}000$ and $k = 4$; RMSE is given as mean ± standard deviation over 5 seeds. Times were measured with six experiments running concurrently and are only comparable with each other.

Method	RMSE	MAE	Time in s
Mean	0.1797 ± 0.0035	0.1265	0.0
KNN	0.1558 ± 0.0031	0.0999	10.5
MICE	0.1444 ± 0.0033	0.0949	0.4
MissForest	0.1354 ± 0.0028	0.0860	25.0
GAIN	0.1558 ± 0.0077	0.1074	20.0
MIDA	0.1625 ± 0.0034	0.1133	7.9
DIME	**0.1290 ± 0.0054**	0.0835	38.5

Table 7: Ablation of DIME at 30% MCAR. RMSE is given as mean ± standard deviation and F_1 as the mean over 5 seeds. PE denotes the positional encoding and NB the neighbourhood statistics.

	California		NYC Airbnb		COR	
Variant	RMSE	F_1	RMSE	F_1	RMSE	F_1
Full DIME	0.1330 ± 0.0016	0.552	0.1520 ± 0.0037	0.605	0.0954 ± 0.0035	0.630
w/o PE	0.1331 ± 0.0012	0.553	0.1514 ± 0.0038	0.597	0.0944 ± 0.0030	0.617
w/o NB	0.1517 ± 0.0106	0.527	0.1532 ± 0.0026	0.595	0.0957 ± 0.0037	0.626
w/o PE+NB	0.1713 ± 0.0020	0.501	0.1627 ± 0.0054	0.602	0.0942 ± 0.0037	0.640
w/o masked loss	0.1626 ± 0.0048	0.520	0.1607 ± 0.0040	0.581	0.1139 ± 0.0050	0.650
w/o extra masking	0.1583 ± 0.0070	0.454	0.1643 ± 0.0040	0.576	0.0989 ± 0.0041	0.644

COR. Without the masked loss, the network partly learns to reproduce the mean placeholders, and its RMSE rises to 0.163 on California and 0.114 on COR. Without the additional masking, the network is never trained to predict hidden values, and its RMSE on California rises to 0.158. On COR, variants with a higher RMSE sometimes obtain a slightly higher F_1, which again shows that RMSE and skyline accuracy are not perfectly aligned when the data have no spatial structure.

Sensitivity and shard-local training.

Figure 5 shows that DIME is insensitive to the hidden width: the RMSE varies by less than 0.004 between $H = 32$ and $H = 256$. The corruption rate matters only below $\pi = 0.2$, and from $\pi = 0.2$ to 0.4 the RMSE is flat. The right panel compares a single model trained by the DO on all regions with one model per shard that is trained only on its own region. The global model always has the lower RMSE, and the gap grows with the number of shards. On California, the RMSE of shard-local models rises from 0.138 for $m = 2$ to 0.152 for $m = 16$ and their F_1 drops from 0.544 to 0.463, whereas the global model stays at 0.133; only for $m = 2$ are the two variants within 0.003 in F_1.

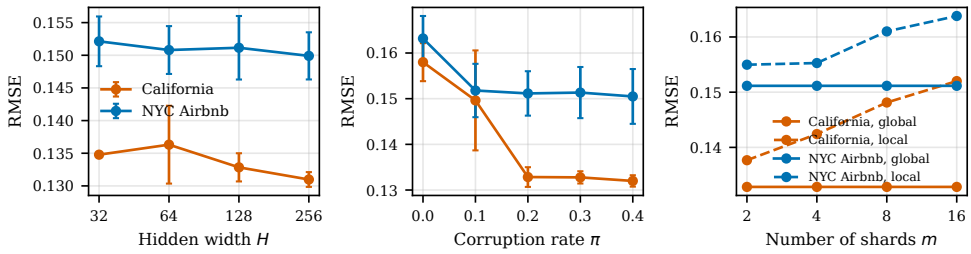


Fig. 5: Sensitivity of DIME to the hidden width H in the left panel and to the corruption rate π in the middle panel. The right panel compares one global model with one local model per shard. All results use 30% MCAR missing values and are averaged over 3 seeds.

Smaller regions provide less training data and fewer neighbours near region borders. This supports imputing at the DO, before the data are partitioned.

9.4 Efficiency of Secure Query Processing

Comparison with secure baselines.

Figure 6 compares SBSQ with SecScan and with the oblivious protocol on a single shard of IND data with $k = 2$. All three methods return the same exact skyline, but at very different costs.

- *Time.* SBSQ is 2.3 times faster than SecScan at $n = 250$ and 7.5 times faster at $n = 4{,}000$, where it needs 35.8 s against 268.3 s. It is 11 times faster than the oblivious protocol at $n = 250$ and 25 times faster at $n = 2{,}000$, where the two need 25.9 s and 641.6 s; we did not run the oblivious protocol at larger sizes. The number of index entries that SBSQ visits falls from 63% of the number of records at $n = 250$ to 18% at $n = 4{,}000$, so its advantage widens as the data grow.
- *Communication.* At $n = 2{,}000$, SBSQ transfers 15.7 MB, compared with 89.4 MB for SecScan and 232.5 MB for the oblivious protocol.
- *Rounds.* SecScan needs a round trip for every record and therefore the most rounds. The oblivious protocol needs relatively few rounds, because each iteration is a tournament of logarithmic depth over all records. At $n = 2{,}000$ it needs 873 rounds, fewer than the 1,400 rounds of SBSQ. On high-latency networks this narrows the gap, but each of its rounds carries far more data and computation.

The oblivious protocol pays this cost for complete access-pattern hiding, whereas SBSQ reveals the leakage described in Section 8.

Scalability with the data size.

Table 8 shows that the cost grows clearly sublinearly with n. A 25-fold increase of the data, from 2,000 to 50,000 records, raises the online latency only 5.7-fold, from 24.7 s to 140.3 s, because the number of visited entries grows roughly with the square root of n and the skyline size grows only logarithmically. We model a LAN with a round-trip

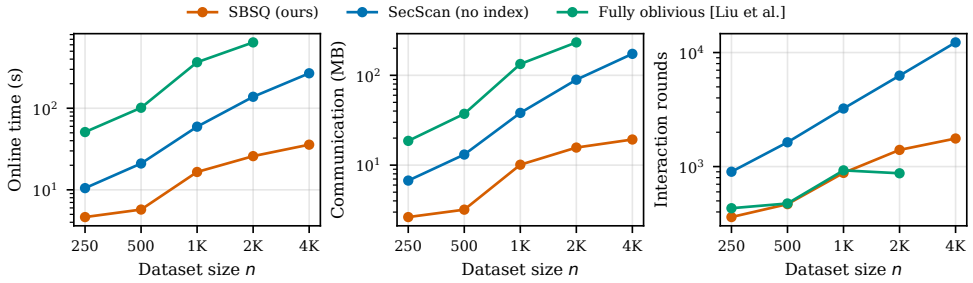


Fig. 6: Online time, communication and interaction rounds of the secure methods versus n on a single shard, for IND data with $k = 2$ and $|N| = 1024$. The oblivious protocol was run up to $n = 2{,}000$.

Table 8: Scalability with the dataset size on IND data with $k = 2$, $m = 4$ and $c = 8$, averaged over 3 queries. |SKY| is the skyline size and Visited the number of index entries visited by all shards. Latency is the online latency, that is, the maximum over the shards plus the merge, and LAN and WAN add the modelled network delay. Comm. is the total traffic, Build the time the DO needs to build and encrypt all ES-Trees, and Index the total index size. BBS and GNN are plaintext baselines.

n	|SKY|	Visited	Latency (s)	Merge (s)	LAN (s)	WAN (s)	Comm. (MB)	Build (s)	Index (MB)	BBS (ms)	GNN (ms)
2,000	33	931	24.7	12.8	25.2	65.1	28	2.6	3.1	14	209
5,000	45	1480	40.4	21.6	41.1	99.1	50	6.2	7.8	22	489
10,000	52	2077	59.4	29.4	60.4	136.6	77	12.8	15.5	40	1181
20,000	57	3011	99.0	42.9	100.5	207.1	125	25.5	30.9	67	2450
50,000	68	4384	140.3	53.3	142.8	321.1	206	63.9	77.0	83	5513

time of 0.5 ms and 1 Gbit/s of bandwidth, and a WAN with 50 ms and 100 Mbit/s. On a LAN the network adds less than 3%. On a WAN the round trips dominate, and the latency at $n = 10{,}000$ grows from 59 s to 137 s. Index construction by the DO is linear in n and costs about 1.3 ms and 1.5 KB per record, so the encrypted index for 50,000 records occupies 77 MB. Plaintext BBS needs 83 ms at $n = 50{,}000$, so privacy costs about three orders of magnitude in time. This is the expected price of Paillier-based secure computation and in line with the other secure baselines. $\mathcal{F}$ accounts for only 17–20% of the online computation, so most of the work falls on the shards, which can be scaled out.

Number of attributes.

The cost of every skyline algorithm grows with the dimensionality, because the skyline grows. On IND data with $n = 2{,}000$, the skyline has 8, 33, 96 and 192 records for $k = 1, \ldots, 4$, and Figure 7 shows that the latency grows from 2.9 s to 26.9 s, 123.9 s and 418.1 s. For $k \geq 3$ the merge dominates and accounts for 86 s and 334 s of the latency, because SMSB compares every local skyline record with every record of the

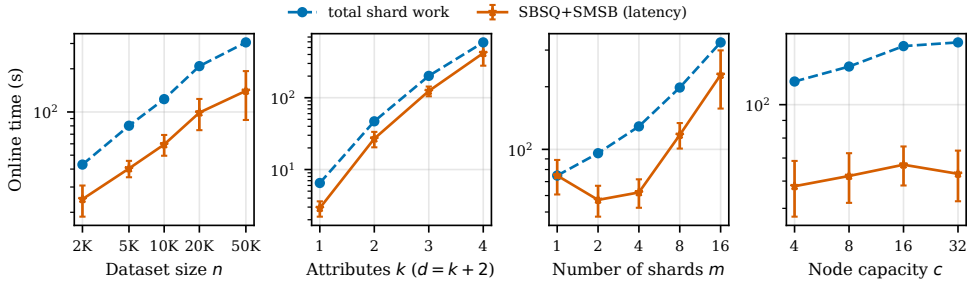


Fig. 7: Online latency as solid lines and total online work summed over all shards as dashed lines, plotted against the dataset size n, the number of attributes k, the number of shards m and the node capacity c. The defaults are $n = 10{,}000$, $k = 2$, $m = 4$ and $c = 8$ on IND data, and the panel for k uses $n = 2{,}000$.

Table 9: Cost per dataset for $n = 2{,}000$, $k = 2$ and $m = 4$, averaged over 3 queries.

Dataset	|SKY|	Visited	SSD pairs	Latency (s)	Comm. (MB)	BBS (ms)
IND	33	931	18,800	26.3	28	17
COR	22	536	6,198	8.5	10	14
ANT	109	1813	133,699	149.3	188	43
California	74	1336	67,147	89.3	95	33
NYC Airbnb	7	549	6,194	9.2	10	7

local skylines of the other shards. This quadratic merge is the main bottleneck of the current design for higher-dimensional data, and Section 10 discusses remedies.

Data distribution.

Table 9 reports the cost on each dataset for $n = 2{,}000$ and $k = 2$. As in plaintext, the cost is governed by the skyline size. Correlated data with 22 skyline records and NYC Airbnb with 7 are answered in about 9 s, independent data in 26 s, and California and anti-correlated data, whose skylines contain 74 and 109 records, in 89 s and 149 s.

Number of shards.

For $n = 10{,}000$ records, increasing the number of shards from 1 to 16 reduces the critical per-shard search time from 74.7 s to 14.1 s, because each shard searches a smaller tree. At the same time the sum of the local skyline sizes grows from 52 to 377, so the merge time grows from 0.2 s to 214.0 s. The resulting trade-off has its optimum at a small number of shards: the computation latency is lowest for $m = 2$, with 57.2 s against 74.8 s for a single shard. Sharding also shortens the sequential chain of round trips on the critical path, so the WAN latency is lowest for $m = 4$, with 139 s against 236 s for a single shard. Beyond eight shards, the merge outweighs the gains of parallel search. In a deployment the number of shards is dictated by the blockchain,

Table 10: Ablation of the protocol design for $n = 5{,}000$ and $k = 2$ on IND data, averaged over 3 queries. Latency and total work are online times, and the column $\mathcal{F}$ gives the online time spent by the full-node entity.

Variant	Latency (s)	Total work (s)	$\mathcal{F}$ (s)	Rounds	Comm. (MB)	Index (MB)	WAN (s)
Full design, $m = 4$	43.1	85.5	15.0	1134	49.9	7.81	101.8
w/o packing	77.3	154.8	126.2	1134	93.9	7.81	137.8
w/o insertion pruning	45.4	92.7	17.0	1721	55.2	7.81	133.6
w/o padding	41.8	82.4	14.9	1100	48.4	7.69	98.9
w/o $E_{S_{NS}}$ precomputation	40.7	81.9	14.2	1134	49.9	6.32	99.5
single shard, $m = 1$	44.7	44.7	8.0	2101	26.7	7.72	151.9

but the result shows that queries should be routed only to the shards whose regions can contribute to the skyline.

Node capacity and key size.

The node capacity has a moderate effect. Small nodes with $c = 4$ visit the fewest entries and give the lowest computation latency of 58.0 s, but they produce deeper trees and more rounds. For $c \geq 8$ the WAN latency is flat at about 139 s, so $c = 8$ is a good default. Increasing the Paillier modulus from 1024 to 2048 bits, as recommended for long-term security, raises the online latency at $n = 5{,}000$ by a factor of 2.9, from 36.1 s to 106.0 s. The index construction time grows by a factor of 5.1, from 6.5 s to 33.1 s, and the index size and the traffic double, from 7.8 MB to 15.6 MB and from 50 MB to 96 MB. The latency grows less than the roughly sixfold cost of a single 2048-bit exponentiation, because a 2048-bit plaintext holds twice as many packing slots, which halves the number of decryptions per comparison.

9.5 Ablation of the Protocol Optimisations

Table 10 removes one design element at a time.

- *Packing* is the most effective optimisation. Without it, $\mathcal{F}$ needs one decryption per comparison. Its online time grows 8.4-fold from 15.0 s to 126.2 s, the latency grows from 43.1 s to 77.3 s, and the traffic almost doubles.
- *Insertion-time pruning* saves 5% of the latency and 34% of the rounds, 1,134 against 1,721. This matters on a WAN, where the latency drops from 133.6 s to 101.8 s.
- *Padding* adds 72 dummy entries, which cost 3% of the latency and 1.6% of the index size. It hides node fan-outs at almost no cost.
- *The precomputed attribute sum* $[\![e.\sigma]\!]$ saves only $k-1$ cheap homomorphic additions per entry. This saving is not measurable, while the precomputed sum enlarges the index by 24% for $k = 2$. It is therefore worthwhile only for larger k or when shards are severely compute-constrained.
- *Sharding.* Four shards do almost twice the total work of a single shard, 85.5 s against 44.7 s, because local skylines contain records that are not globally optimal and must be merged. The computation latency is similar at 43.1 s and 44.7 s, but the WAN

latency is 33% lower, 101.8 s against 151.9 s, because the per-shard searches proceed in parallel with fewer sequential rounds.

On-chain cost.

The blockchain stores only coordination records. Under the gas schedule of Ethereum [73], anchoring one 32-byte index digest costs about 20,000 gas for the storage write. A query transaction carrying $[\![q]\!]$, which consists of two 256-byte ciphertexts for $|N| = 1024$, costs about $21{,}000 + 16 \cdot 512 \approx 29{,}000$ gas, and a result digest costs about as much as an index digest. These costs do not depend on n, because all data-dependent computation happens off chain.

10 Discussion

Why impute before encrypting?

An alternative to imputation is to change the query semantics, as incomplete-data skyline algorithms do [10]. Our results show that this is not a viable option in the setting studied here. Cyclic dominance eliminates almost every genuine skyline record. Moreover, the lack of transitivity would invalidate Lemma 5, so the local skylines of different shards could no longer be merged locally. Imputation restores a complete, transitive dominance relation, which both the progressive search and the constant-round merge rely on. The price is that the DO must run DIME on plaintext before encryption. This is consistent with our trust model, because the DO owns the data anyway.

Privacy–efficiency trade-off.

SBSQ deliberately reveals the access pattern and the order of the keys of the visited entries to the shard, in exchange for touching only a small fraction of the index. The fully oblivious baseline hides this information but, as Section 9.4 shows, is far more expensive. We consider the leakage acceptable when shards are semi-trusted committees of a permissioned or consortium chain, which is the typical deployment of data-sharing DApps. For open networks with stronger adversaries, the oblivious variant or an ORAM-backed index [67] should be used.

The merge bottleneck.

The experiments identify SMSB as the dominant cost when local skylines are large, which happens with many shards or many attributes. Its constant number of rounds is achieved by testing every local skyline record against all other local skylines. This takes $O(u^2)$ comparisons, which becomes expensive once the sum u of the local skyline sizes reaches a few hundred. Two remedies are natural. First, the coordinator can seed every shard with a few globally strong records, for example the local skyline records with the smallest keys, so that shards prune locally dominated entries before the merge. Second, queries can be routed only to shards whose root bounding rectangle is not dominated by such a seed. Both remedies trade additional rounds for fewer comparisons and fit directly into the protocols presented here.

Limitations.

First, we assume semi-honest, non-colluding shards and $\mathcal{F}$; extending the protocols to malicious adversaries, for example with verifiable computation [27] or honest-majority secret sharing [25], is left for future work. Second, as discussed in Section 8, the comparison protocol reveals the rough magnitude of compared differences to $\mathcal{F}$. Third, our evaluation runs all shards and $\mathcal{F}$ on a single machine, measures computation directly and models network latency analytically from the measured number of rounds and bytes. A deployment on a real sharded blockchain would add consensus latency for the on-chain coordination steps, which is independent of our protocols. Fourth, the ES-Tree is built for a static data release. Inserting a record requires re-encrypting one root-to-leaf path, and DIME can impute new records with a single forward pass, but we have not evaluated dynamic workloads. Finally, DIME relies on spatial autocorrelation. On data without spatial structure its advantage disappears. On IND it performs on par with mean imputation. On COR it performs on par with MICE and MissForest. On ANT, where the attributes depend linearly on each other, MICE and MissForest are more accurate. A practical deployment should therefore validate DIME against a regression-based imputer on held-out observed values, which DIME's validation split already provides.

11 Conclusion

This paper studied privacy-preserving location-based skyline queries over incomplete data in sharded blockchain networks. We showed that incomplete-data skyline semantics are not a viable answer to missing values in this setting: on real data with 30% missing values they attain F_1 scores of 0.02–0.30, and they break the transitivity on which an efficient cross-shard merge depends. We therefore proposed to repair the data before encryption with DIME, a mask-aware denoising autoencoder enhanced with spatial context. On real geo-referenced data, DIME reduced the imputation error by up to 15% relative to the strongest baseline and produced the most accurate skylines in most settings.

On the repaired data, the ES-Tree with padded, Paillier-encrypted entries, together with the SSD, SBSQ and SMSB protocols, answers queries exactly. Shards learn only a well-defined access-pattern leakage, and the key holder sees only blinded, shuffled values. An implementation with real cryptography returned the exact skyline in every experiment. It was 7.5 times faster than an index-free secure scan and 25 times faster than a fully oblivious protocol, and its cost grew sublinearly with the data size. Plaintext packing was the most effective optimisation, while the quadratic merge emerged as the main bottleneck for many shards or many attributes.

Future work includes seeding and routing strategies that reduce the merge cost, protocols that tolerate malicious shards, threshold decryption to remove the single key holder, and an evaluation on a deployed sharded blockchain with dynamic data.

References

[1] Rudman, R., Bruwer, R.: Defining web 3.0: opportunities and challenges. The

electronic library **34**(1), 132–154 (2016)

[2] 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)

[3] 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), 13601 (2019)

[4] 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)

[5] 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 (2023). IEEE

[6] 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)

[7] Borzsony, S., Kossmann, D., Stocker, K.: The Skyline operator. In: Proceedings 17th International Conference on Data Engineering, pp. 421–430. IEEE Comput. Soc, Los Alamitos, CA, USA (2001). https://doi.org/10.1109/icde.2001.914855

[8] Sharifzadeh, M., Shahabi, C.: The spatial skyline queries. In: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), pp. 751–762 (2006)

[9] 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)

[10] Khalefa, M.E., Mokbel, M.F., Levandoski, J.J.: Skyline Query Processing for Incomplete Data. In: 2008 IEEE 24th International Conference on Data Engineering, pp. 556–565. IEEE, Piscataway, NJ, USA (2008). https://doi.org/10.1109/icde.2008.4497464

[11] 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)

[12] 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)

[13] 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 (2023). Springer

[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] Naveed, M., Kamara, S., Wright, C.V.: Inference Attacks on Property-Preserving Encrypted Databases. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 644–655. ACM, New York, NY, USA (2015). https://doi.org/10.1145/2810103.2813651

[16] Liu, J., Yang, J., Xiong, L., Pei, J.: Secure Skyline Queries on Cloud Platform. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 633–644. IEEE, Piscataway, NJ, USA (2017). https://doi.org/10.1109/icde.2017.117

[17] Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A Secure Sharding Protocol For Open Blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 17–30. ACM, New York, NY, USA (2016). https://doi.org/10.1145/2976749.2978389

[18] Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., Ford, B.: OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 583–598. IEEE, Piscataway, NJ, USA (2018). https://doi.org/10.1109/sp.2018.000-5

[19] Zamani, M., Movahedi, M., Raykova, M.: RapidChain: Scaling Blockchain via Full Sharding. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 931–948. ACM, New York, NY, USA (2018). https://doi.org/10.1145/3243734.3243853

[20] 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)

[21] Balke, W.-T., Güntzer, U., Zheng, J.X.: Efficient Distributed Skylining for Web Information Systems, pp. 256–273. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24741-8_16

[22] Hose, K., Vlachou, A.: A survey of skyline processing in highly distributed environments. The VLDB Journal **21**, 359–384 (2012)

[23] 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)

[24] 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)

[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)

[26] Hao, W., Liu, S., Lv, C., Wang, Y., Wang, J.: Efficient and privacy-preserving multi-party skyline queries in online medical primary diagnosis. Journal of King Saud University - Computer and Information Sciences **35**(8), 101637 (2023) https://doi.org/10.1016/j.jksuci.2023.101637

[27] Xu, C., Zhang, C., Xu, J.: vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases. In: Proceedings of the 2019 International Conference on Management of Data, pp. 141–158. ACM, New York, NY, USA (2019). https://doi.org/10.1145/3299869.3300083

[28] Zhang, C., Xu, C., Xu, J., Tang, Y., Choi, B.: GEM^2-Tree: A gas-efficient structure for authenticated range queries in blockchain. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 842–853. IEEE, Piscataway, NJ, USA (2019). https://doi.org/10.1109/icde.2019.00080

[29] 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)

[30] Yin, B., Xu, B., Liu, Y., Wang, T.: Enabling secure and traceable query services for internet of things using blockchain. IEEE Transactions on Network and Service Management (2024)

[31] Miao, X., Gao, Y., Zheng, B., Chen, G., Cui, H.: Top-k Dominating Queries on Incomplete Data. IEEE Transactions on Knowledge and Data Engineering **28**(1), 252–266 (2016) https://doi.org/10.1109/tkde.2015.2460742

[32] 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)

[33] Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405), pp. 717–719. IEEE, Piscataway, NJ, USA (2003). https://doi.org/10.1109/icde.2003.1260846

[34] 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 (2002). Elsevier

[35] Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Transactions on Database Systems **30**(1), 41–82 (2005) https://doi.org/10.1145/1061318.1061320

[36] Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data - SIGMOD '84, p. 47. ACM Press, New York, NY, USA (1984). https://doi.org/10.1145/602259.602266

[37] Song, H., Wu, Z., Guo, B., Wu, Z., Wang, M.: An efficient approach towards index structures for skyline queries. Journal of King Saud University Computer and Information Sciences **37**(6) (2025) https://doi.org/10.1007/s44443-025-00183-3

[38] Bai, M., Jiang, S., Zhang, X., Wang, X.: An efficient skyline query algorithm in the distributed environment. Journal of Computational Science **58**, 101524 (2022)

[39] 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 (2024). IEEE

[40] 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**, 1682 (2023)

[41] 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 (1999). Springer

[42] 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 (2014). IEEE

[43] 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)

[44] Sharma, R., Villányi, B.: A spatiotemporal Order-Revealing query processing approach for Industrial Internet of Things. Journal of King Saud University - Computer and Information Sciences **34**(10), 8985–8995 (2022) https://doi.org/10.1016/j.jksuci.2022.08.024

[45] Tian, M., Zhang, Y., Zhu, Y., Wang, W., Wu, Q., Xiang, Y.: BPPIR: Blockchain-assisted privacy-preserving similarity image retrieval over multiple clouds. Journal of King Saud University - Computer and Information Sciences **35**(1), 324–334 (2023) https://doi.org/10.1016/j.jksuci.2022.12.003

[46] Wu, J., Yuan, L., Xie, T., Dai, H.: A sharding blockchain protocol for enhanced scalability and performance optimization through account transaction reconfiguration. Journal of King Saud University - Computer and Information Sciences **36**(8), 102184 (2024) https://doi.org/10.1016/j.jksuci.2024.102184

[47] RUBIN, D.B.: Inference and missing data. Biometrika **63**(3), 581–592 (1976) https://doi.org/10.1093/biomet/63.3.581

[48] Little, R., Rubin, D.: Statistical Analysis with Missing Data, 3rd edn. Wiley, Hoboken, NJ, USA (2019). https://doi.org/10.1002/9781119482260

[49] Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.B.: Missing value estimation methods for DNA microarrays. Bioinformatics **17**(6), 520–525 (2001) https://doi.org/10.1093/bioinformatics/17.6.520

[50] Buuren, S.v., Groothuis-Oudshoorn, K.: mice: Multivariate Imputation by Chained Equations inR. Journal of Statistical Software **45**(3) (2011) https://doi.org/10.18637/jss.v045.i03

[51] Stekhoven, D.J., Bühlmann, P.: MissForest—non-parametric missing value imputation for mixed-type data. Bioinformatics **28**(1), 112–118 (2011) https://doi.org/10.1093/bioinformatics/btr597

[52] Jäger, S., Allhorn, A., Bießmann, F.: A Benchmark for Data Imputation Methods. Frontiers in Big Data **4** (2021) https://doi.org/10.3389/fdata.2021.693674

[53] Ali, A., Abu-Elkheir, M., Atwan, A., Elmogy, M.: Missing values imputation using Fuzzy K-Top Matching Value. Journal of King Saud University - Computer and Information Sciences **35**(1), 426–437 (2023) https://doi.org/10.1016/j.jksuci.2022.12.011

[54] Gondara, L., Wang, K.: MIDA: Multiple Imputation Using Denoising Autoencoders, pp. 260–272. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93040-4_21

[55] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research **11**, 3371–3408 (2010)

[56] Yoon, J., Jordon, J., Schaar, M.: GAIN: Missing data imputation using generative adversarial nets. In: Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 5689–5698 (2018)

[57] Tobler, W.R.: A Computer Movie Simulating Urban Growth in the Detroit

Region. Economic Geography **46**, 234 (1970) https://doi.org/10.2307/143141

[58] Damgård, I., Jurik, M., Nielsen, J.B.: A generalization of Paillier's public-key system with applications to electronic voting. International Journal of Information Security **9**(6), 371–385 (2010) https://doi.org/10.1007/s10207-010-0119-9

[59] Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. In: Advances in Neural Information Processing Systems, vol. 33, pp. 7537–7547 (2020)

[60] Hinton, G.E., Salakhutdinov, R.R.: Reducing the Dimensionality of Data with Neural Networks. Science **313**(5786), 504–507 (2006) https://doi.org/10.1126/science.1127647

[61] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations (ICLR) (2015)

[62] Leutenegger, S.T., Lopez, M.A., Edgington, J.: STR: a simple and efficient algorithm for R-tree packing. In: Proceedings 13th International Conference on Data Engineering, pp. 497–506. IEEE Comput. Soc. Press, Los Alamitos, CA, USA (1997). https://doi.org/10.1109/icde.1997.582015

[63] Goldreich, O.: Foundations of Cryptography. Cambridge University Press, Cambridge, UK (2004). https://doi.org/10.1017/cbo9780511721656

[64] Yao, A.C.-C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (Sfcs 1986), pp. 162–167 (1986). IEEE

[65] Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 79–88. ACM, New York, NY, USA (2006). https://doi.org/10.1145/1180405.1180417

[66] Islam, M.S., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In: Network and Distributed System Security Symposium (NDSS) (2012)

[67] Stefanov, E., Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X., Devadas, S.: Path ORAM: an extremely simple oblivious RAM protocol. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security - CCS '13, pp. 299–310. ACM Press, New York, NY, USA (2013). https://doi.org/10.1145/2508859.2516660

[68] Damgård, I., Geisler, M., Krøigaard, M.: Efficient and Secure Comparison for On-Line Auctions, pp. 416–430. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73458-1_30

[69] Kelley Pace, R., Barry, R.: Sparse spatial autoregressions. Statistics & Probability Letters **33**(3), 291–297 (1997) https://doi.org/10.1016/s0167-7152(96)00140-x

[70] Inside Airbnb: Inside Airbnb: New York City listings. https://insideairbnb.com/get-the-data/. Snapshot of 14 September 2026, accessed 20 October 2026 (2026)

[71] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., *et al.*: PyTorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035 (2019)

[72] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

[73] Wood, G.: Ethereum: A Secure Decentralised Generalised Transaction Ledger. Ethereum Project Yellow Paper (2014)