Indexing Dynamic Encrypted Database in Cloud for Efficient Secure k-Nearest Neighbor Query

Xingxin LI, Youwen ZHU, Rui XU, Jian WANG, Yushu ZHANG

Frontiers of Computer Science, DOI: 10.1007/s11704-022-2401-1

Problems & Ideas

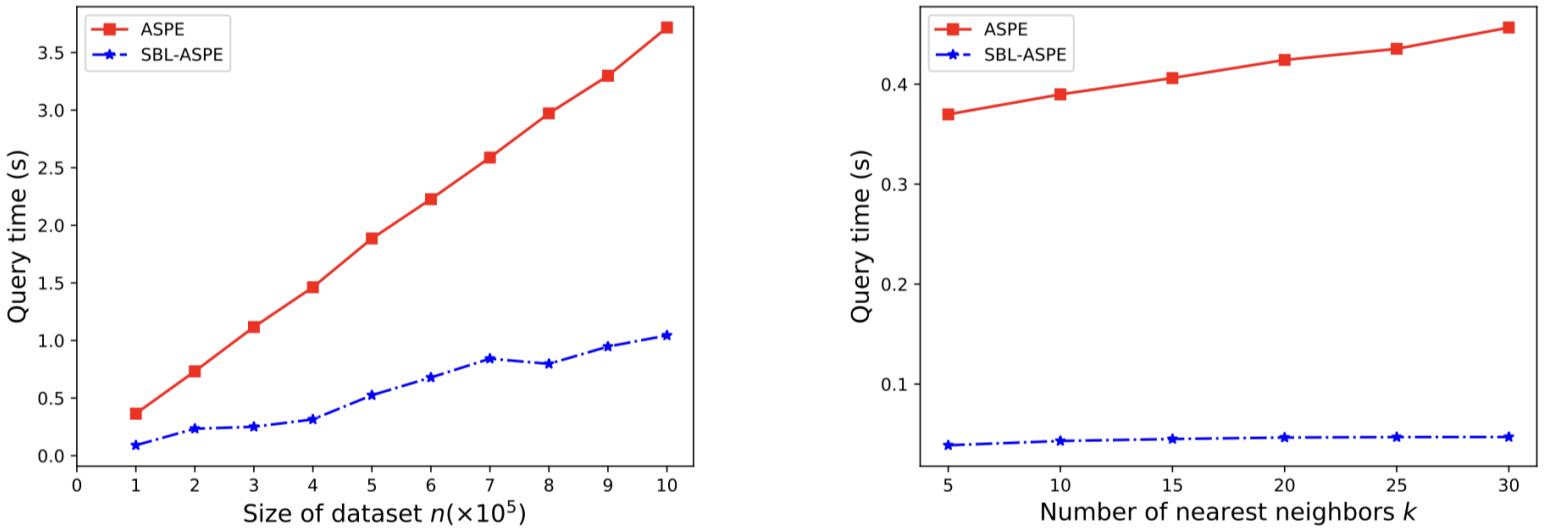
- Problems of secure k-Nearest Neighbor (kNN)Query:
  - Most secure kNN query schemes have linear search complexity regarding to dataset sizes.
  - Existing index strategy for secure kNN query cannot support dynamic update over encrypted datasets.
- Ideas: A new encrypt-then-index strategy that transforms encrypted databases and then builds index on the transformed data.

- **KeyGen**$(\cdot) \rightarrow ky$: The DO randomly generates a $(d+1) \times (d+1)$ invertible matrix M as the secret key.
- **DBEnc**$(ky, D) \rightarrow D'$: For the database D, the DO encrypts each data record $p_i \in D$ with the key ky as $p' = (p_i, -0.5\|p_i\|^2)M^{-1}$, where $\|p_i\|^2$ is the squared Euclidean norm and outputs the encrypted database $D' = \{p'_1, \ldots, p'_n\}$.We also use **DBEnc**$(ky, p) \rightarrow p'$ to denote the encryption of a single data point p.
- **TransED**$(D') \rightarrow D^*$: The CS computes the squared Euclidean norm $\|p'_i\|^2$ of each encrypted data point p'_i and finds the maximum among them, denoted as $\alpha = \max_{1 \le i \le n} \|p'_i\|^2$. The CS then transforms each encrypted data point $p'_i \in D'$ to $p^*_i = (p'_i, \sqrt{\alpha - \|p'_i\|^2})$, and outputs a transformed database $D^* = \{p^*_1, p^*_2, \cdots, p^*_n\}$.
- **Index**$(D^*) \rightarrow \Gamma$: The CS builds an R-tree index Γ on the transformed encrypted database D^* by using the algorithm Rtree.**Build**(D^*). Note that any effective spatial data structures beyond the R-tree that can improve the distance comparison can apply here. In our constructions, we use an R-tree index as the spatial data structure for its simplicity and popularity.
- **QueryEnc**$(ky, q) \rightarrow q'$: The QU takes the key ky and a query point q and outputs the encryption of q as $q' = rM^{-1}(q, 1)^T$, where r is a random positive number selected by the QU and $(\cdot)^T$ denotes the transpose of a vector.
- **TransEQ**$(q') \rightarrow q^*$: The CS transforms the encrypted query point q' to a vector $q^* = (q', 0)$.
- k-**NNCompt**$(\Gamma, q^*, k) \rightarrow (\hat{p}^*_1, \ldots, \hat{p}^*_k)$: The CS traverses the R-tree Γ and finds out the k-NN results of q^* from D^* by invoking the k-NN query algorithm Rtree.k**NNQuery**(Γ, q^*, k). The results are denoted by $(\hat{p}^*_1, \ldots, \hat{p}^*_k)$.

Figure shows the main algorithms of our kNN query scheme which simultaneously achieves sub-linear complexity for kNN queries and allows data owners to dynamically update their databases. The correctness can be guaranteed because $dis(p_i, q) \le dis(p_j, q) \Leftrightarrow dis(p^*_i, q^*) \le dis(p^*_j, q^*)$.

Main Contributions

- Contributions:
  - A new encrypt-then-index strategy enables sub-linear complexity for k-NN queries and allows data owners to dynamically update their databases;
  - We apply our strategy to improve the efficiency and flexibility of multiple widely-used schemes without sacrificing security, such as ASPE, ZHT, and MRSE;



The query time of our scheme SBL-ASPE and original work ASPE. Left: the query time when k is fixed to 1; Right: the query time when the dataset size n is fixed to 10^5.