A SECURITY ANALYSIS

Security proof. The security analysis of our query rewriter is based on the composition lemma in [24] (Sec. 7.3.1). The lemma states that given a secure protocol $\pi^{g|f}$ that can securely compute g based on a plaintext query f and a secure protocol π^{f} for operation f, the operation g can be securely computed by executing protocol $\pi^{g|f}$ but substitutes every plaintext query f with an execution of π^{f}. We prove the security of query rewriter for radius-known and radius-unknown queries separately.

- **Security of Radius-Known Queries.** Federated range query, range counting and distance join belong to *radius-known queries*. For *federated range query* and *distance join*, the secure protocol $\pi^{g|f}$ is the secure set union (Def. 8) based on the query results of one or multiple plaintext range queries over the federation. The protocol π^{f} corresponds to the plaintext range query (Def. 5). The protocol π^{f} is secure, since the plaintext range query only occurs in each silo, which will not leak any information to other silos. From the composition lemma, federated range query and distance join are secure against semi-honest adversaries. Federated range counting is also secure based on a similar proof (*i.e.* replacing $\pi^{g|f}$ with the secure summation in Def. 6 and π^{f} with the plaintext range counting in Def. 5).
- **Security of Radius-Unknown Queries.** The federated kNN query and kNN join belong to *radius-unknown queries*. For federated kNN query (Alg. 1), let protocol π^{f} be the secure comparison between k and the local counts. The security of π^{f} can be proved in the same way as before. Denote protocol $\pi^{g|f}$ as Alg. 1 based on the plaintext comparison query. We then show protocol $\pi^{g|f}$ is also secure. Assuming semi-honest adversaries, each silo can simulate its view of the execution of the protocol. Specifically, knowing the range $[l, u]$ used at the beginning of a round, each silo can compute *thres* used in that round. If *thres* is the same as the final radius, it concludes that the protocol must have ended in this round. Otherwise, it simply updates the range to that side of *thres* which contains the final radius. Along with the knowledge of the initial range, this shows that each silo can simulate the execution of the protocol, which means the protocol $\pi^{g|f}$ is secure. Thus, our Alg. 1 is secure against semi-honest adversaries based on the composition lemma. The security of federated kNN join can be proved similarly.

Table 4: A case study on hacking federated range counting in Hu-Fu.

Colluded silos	Attack success rate	Running time
1	0	No result for 2 months
2	0	No result for 2 months
3	0	No result for 2 months

Case Study. We have also conducted a case study to show how Hu-Fu defenses against the semi-honest adversaries when processing federated range counting.

- **Experimental Setup.** The case study is based on the BJ dataset with 6 silos (*i.e.* our default setting) and we vary the number m of colluded adversaries (*i.e.* silos) from 1 to 3. Then, when processing federated range counting, the attack will occur at the only secure operator (*i.e.* secure summation [19]). To attack the secure summation, we use the method introduced in [19]. In general, a secure operator successfully defenses against the attack, if adversaries cannot get any feasible result after a long time (*e.g.*, two months in our experiment). Furthermore, the other experiment parameters are the same as those in Sec. 7.2.
- **Experimental Result.** As shown in Table 4, even when 50% of the silos collude, they cannot infer any feasible result even after two months' cracking. Note that it is commonly assumed in existing work [10, 12] that the portion of colluded adversaries is less than 50%. Based on the computational security [20], this result proves Hu-Fu is hard to attack.

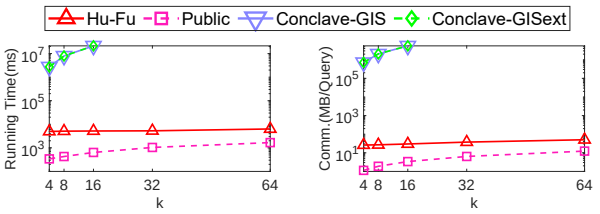


Figure 12: Performance of federated kNN join when varying k.

B EXPERIMENT ON REAL DATASET

Fig. 12 illustrates the results of federated kNN join on the BJ dataset when varying the value of k. Note that partial results of Conclave-GIS and Conclave-GISext (when $k > 16$) are not provided, since they take longer than 6 hours to process this query. As is shown, Hu-Fu is at least 553× faster and takes 27, 404× lower communication cost than Conclave-GIS. As the parameter k of federated kNN join increases from 4 to 64, the running time and communication cost of Hu-Fu only increase by 28% and 48% respectively, because the parameter k has relatively marginal impact on the federated kNN query and a federated kNN join is decomposed into multiple federated kNN queries for all the compared solutions.

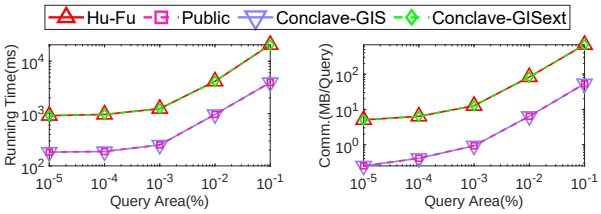


Figure 13: Performance of federated distance join when varying query area.

Fig. 13 presents the performance of federated distance join on the BJ dataset when varying the query area. The result of federated distance join when varying the query area shows a similar pattern with that of federated range query. This is because a federated distance join is decomposed into multiple federated range queries for all the compared solutions. The increase of both running time and communication cost is caused by the increase of the number of retrieved spatial objects.

The SMCQL-GIS and SMCQL-GISext are excluded from both Fig. 12 and Fig. 13 because they only support two silos, and the tests in this section are conducted on six silos.

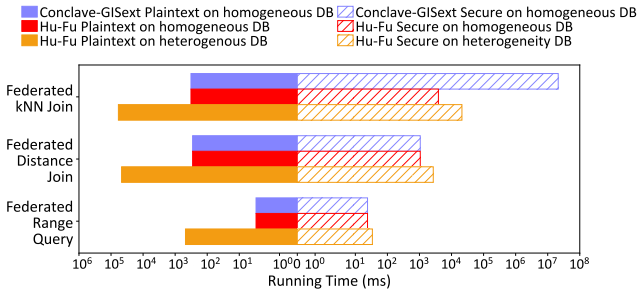


Figure 14: Running time breakdown.

C EXPERIMENT ON HETEROGENEOUS SILOS

Fig. 14 shows the results of federated range query, distance join and kNN join in the experiment of heterogeneous silos. We can first observe that our Hu-Fu can support all six spatial database systems, including PostGIS [45], MySQL [61], SpatiaLite [51], Simba [64], GeoMesa [27], and SpatialHadoop [16]. In addition, for all federated spatial queries in Fig. 14, we observe the running time of Hu-Fu becomes longer, compared with the results of homogeneous silos (*i.e.*, PostGIS in this test). This is because the time cost of Hu-Fu's plaintext primitives (*i.e.* plaintext range query) increases due to the slowest spatial database system. We can also observe that the experimental patterns of federated distance join and federated kNN join are similar to those of federated range query and federated kNN query. This is because a federated distance/kNN join is decomposed into a series of federated range/kNN query. Besides, we also recommend that all silos can use more efficient spatial database systems to further speed up the federated spatial queries.

D EXPERIMENT ON HU-FU WITH AN HONEST BROKER

In the following, we conduct an experiment to demonstrate the performance of Hu-Fu with an honest broker (denoted by "Hu-Fu-HB").

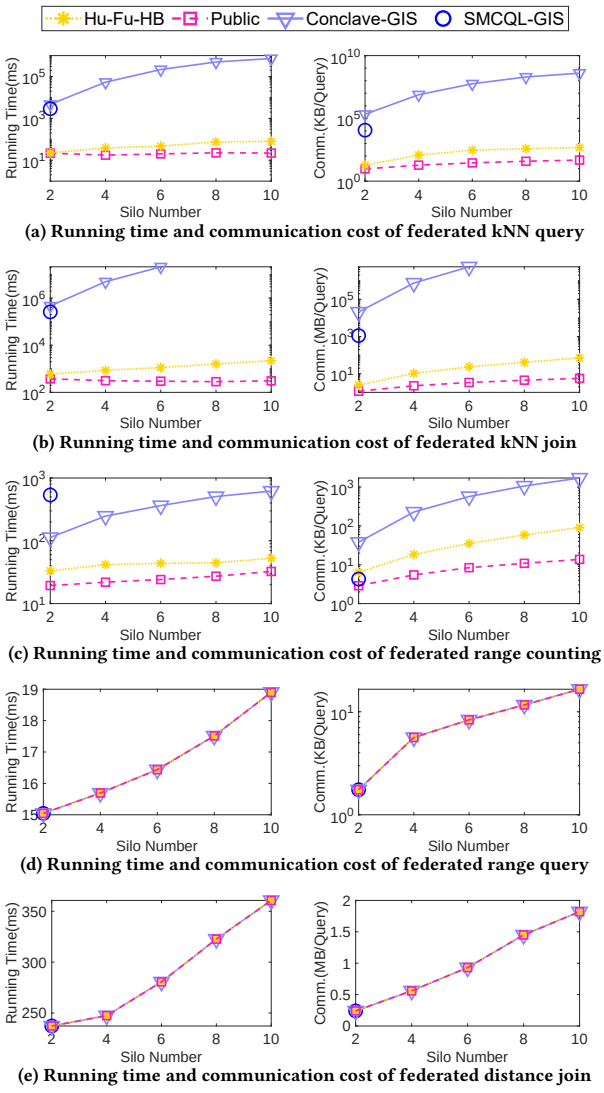
Experimental Setup. This experiment has tested all federated spatial queries on the real dataset BJ. We compare Hu-Fu-HB with SMCQL-GIS/Conclave-GIS in query processing efficiency measured in running time and communication cost. Besides, the other experiment settings are the same as those in Sec. 7.2.

Experimental Result. Fig. 15 and Fig. 16 present the experimental results of Hu-Fu-HB and SMCQL-GIS/Conclave-GIS in terms of running time and communication cost. First, We can observe that Hu-Fu-HB performs the same as SMCQL-GIS and Conclave-GIS on federated range query and distance join. This is because all the compared algorithms (including ours) are simplified to Public due to the honest broker who collects the partial results (*i.e.* sensitive data) of plaintext range query in each silo and is assumed to never reveal them to anyone else. Second, as for federated range counting, kNN query and kNN join, Hu-Fu-HB still outperforms SMCQL-GIS and Conclave-GIS with up to 4 orders of magnitudes faster in running time and 5 orders of magnitudes lower in communication cost. This is because our query writer is more effective to process queries on large-scale spatial data.

Summary. The experiment result illustrates that compared with SMCQL-GIS/Conclave-GIS, Hu-Fu-HB has similar efficiency in federated range query and distance join. Meanwhile, Hu-Fu-HB achieves better efficiency in federated range counting, kNN query and kNN join.

E EXPERIMENT ON GEOGRAPHICALLY PARTITIONED DATASET

To explore the performance of Hu-Fu in the dataset where each silo corresponds to a single country, we have conducted an experiment on a new synthetic dataset (denoted by "*OSM country*").



(a) Running time and communication cost of federated kNN query

(b) Running time and communication cost of federated kNN join

(c) Running time and communication cost of federated range counting

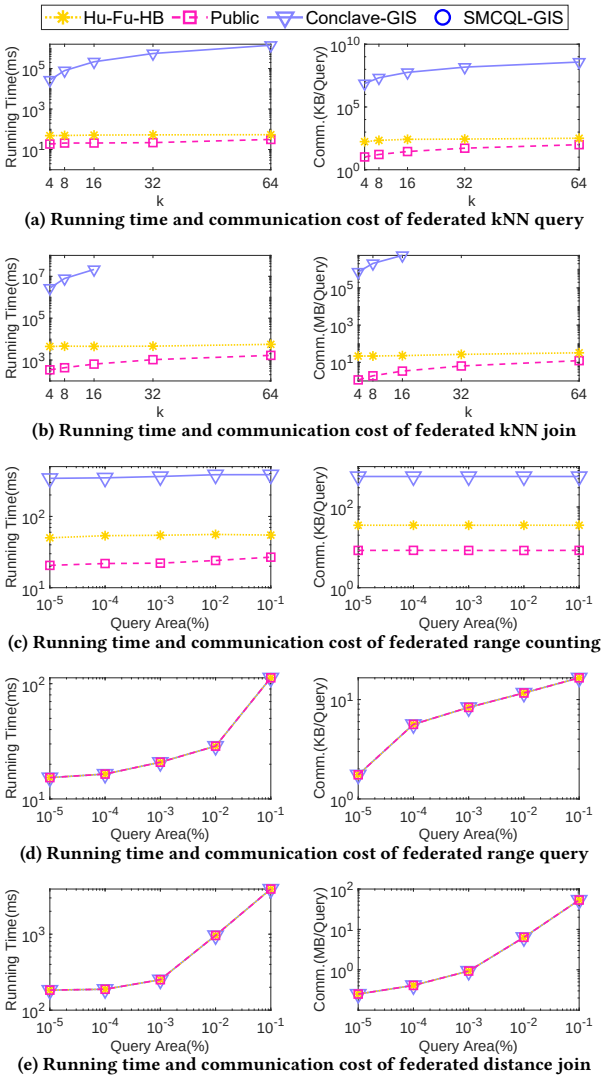
(d) Running time and communication cost of federated range query

(e) Running time and communication cost of federated distance join

Figure 15: Hu-Fu with an honest broker varying the silo number.

Experimental Setup. The *OSM country* dataset consists of spatial points from six countries in OpenStreetMap [40]. These countries include China, India, Japan, Malaysia, North Korea and South Korea, and each of them corresponds to a single data silo. As shown in Table 5, the volume of data in each silo follows the native data proportion of the corresponding country in OpenStreetMap. For example, the numbers of data points in China, India and Japan are

much larger than those of Malaysia, North Korea and South Korea. Here, we vary the total number of all data points from 10^4 to 10^9. Besides, the other experiment parameters are the same as those in Sec. 7.3.



(a) Running time and communication cost of federated kNN query

(b) Running time and communication cost of federated kNN join

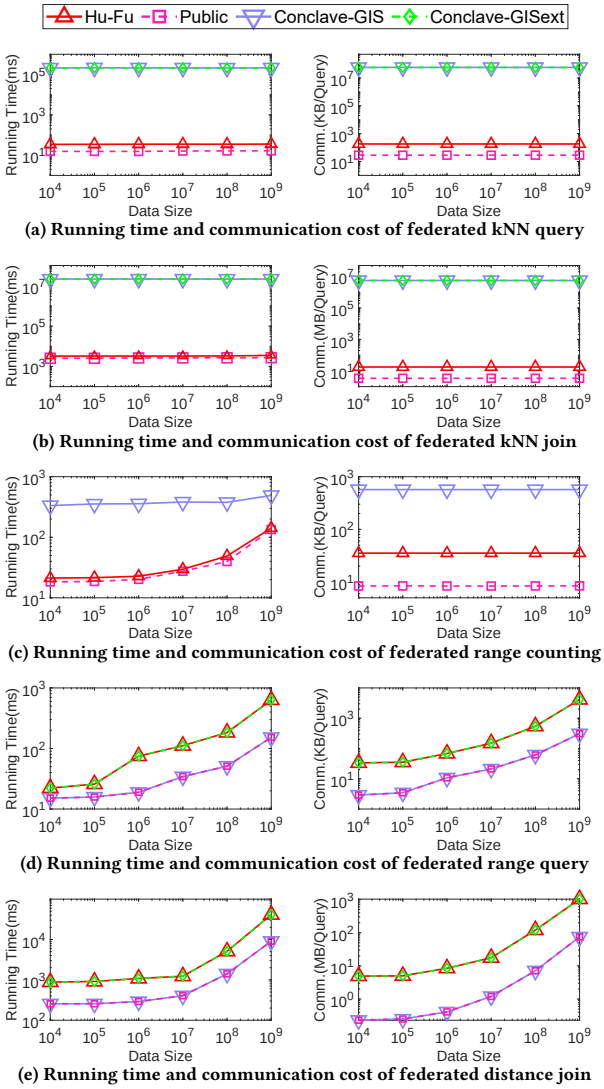
(c) Running time and communication cost of federated range counting

(d) Running time and communication cost of federated range query

(e) Running time and communication cost of federated distance join

Figure 16: Hu-Fu with an honest broker varying the query-specific parameter.

Experimental Result. The results of this experiment are shown in Fig. 17. First, we can observe that in the *OSM country* dataset, Hu-Fu also shows a significant improvement in both running time and communication cost over Conclave-GIS and Conclave-GISext on federated kNN query, kNN join and range counting. For example, Hu-Fu is at least 3 orders of magnitude faster than Conclave-GIS and Conclave-GISext in federated kNN query and kNN join, and costs 5 orders of magnitude lower network communication. Second, Hu-Fu achieves the same efficiency as Conclave-GISext on federated range query and distance join. Overall, the ranking of the compared solutions in terms of either running time or communication cost is consistent with the ranking in Sec. 7.3. Note that we exclude Conclave-GISext from the results of federated range counting as shown in Fig. 17c, because this query does not need the secure set union to protect data ownership in this query.



(a) Running time and communication cost of federated kNN query

(b) Running time and communication cost of federated kNN join

(c) Running time and communication cost of federated range counting

(d) Running time and communication cost of federated range query

(e) Running time and communication cost of federated distance join

Figure 17: Scalability test on synthetic dataset partitioned by countries.

Summary. The experiment on *OSM country* dataset has a similar result with that in Sec. 7.3. Specifically, Hu-Fu outperforms SMCQL-GISext/Conclave-GISext on federated kNN query, kNN join and range counting. Meanwhile, Hu-Fu performs similarly as SMCQL-GISext/Conclave-GISext on federated range query and distance join.

F EXPERIMENT ON THE IMPROVEMENT BY THE INDEX IN EACH DATA SILO

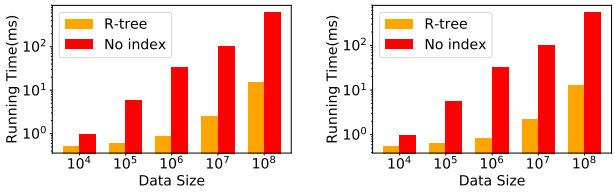
To demonstrate that the federated spatial queries have already been accelerated by local indexes in Hu-Fu, we have conducted a new experiment on the efficiency improvement caused by these local indexes (*i.e.* R-trees in our default setting).

Table 5: Percentage of data of each country in *OSM country*.

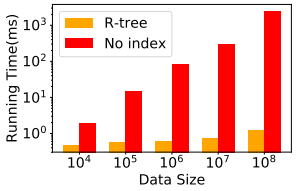
Country	Percentage of data
China	21.7%
India	29.8%
Japan	39.4%
Malaysia	4.2%
North Korea	1.2%
South Korea	3.7%

Experimental Setup. In this experiment, we test the running time of the plaintext spatial queries in Hu-Fu (*i.e.* plaintext range query, plaintext range counting and plaintext kNN query) on PostgreSQL with and without an R-tree. And the plaintext spatial queries are processed on the OSM dataset. Besides, we vary the data size from 10^4 to 10^8 and use the default setting of query area (0.001%) and k (16). The other experiment settings are the same as those in Sec. 7.3.

Experimental Result. As shown in Fig. 18, local indexes can improve the efficiency of plaintext spatial queries, especially when the data size is large. Specifically, the local index (R-tree) reduces the running time by up to 40×, 44× and 2042× when processing plaintext range query, range counting, and kNN query, respectively. Moreover, the plaintext spatial queries without local indexes cost even longer time than corresponding federated spatial queries in Hu-Fu. For instance, the running time of processing one federated kNN query by Hu-Fu takes 52 ms when the data size is 10^8, which is even faster than that of the plaintext kNN query without the local index (2465 ms). Thus, the experimental result proves that we have used local indexes in Hu-Fu to speed up the processing of the plaintext operators.



(a) Plaintext range query **(b) Plaintext range counting**



(c) Plaintext kNN query

Figure 18: Running time of plaintext spatial queries with/without R-tree.