Outsourcing Search Services on Private Spatial Data

Man Lung Yiu[†] Gabriel Ghinita[‡] Christian S. Jensen[†] Panos Kalnis[‡]

[†]*Department of Computer Science, Aalborg University*
{mly, csj}@cs.aau.dk

[‡]*Department of Computer Science, National University of Singapore*
{ghinitag, kalnis}@comp.nus.edu.sg

***Abstract*— Social networking and content sharing service providers, e.g., Facebook and Google Maps, enable their users to upload and share a variety of user-generated content, including location data such as points of interest. Users wish to share location data through an (untrusted) service provider such that trusted friends can perform spatial queries on the data. We solve the problem by transforming the location data before uploading them. We contribute spatial transformations that redistribute locations in space and a transformation that employs cryptographic techniques. The data owner selects transformation keys and shares them with the trusted friends. Without the keys, it is infeasible for an attacker to reconstruct the exact original data points from the transformed points. These transformations achieve different tradeoffs between query efficiency and data security. In addition, we describe an attack model for studying the security properties of the transformations. Empirical studies suggest that the proposed methods are secure and efficient.**

I. Introduction

Consider this scenario: On a trip to Paris, Alice takes photos with her GPS-equipped camera phone (e.g., Nokia N95) that automatically geo-tags each photo. She uploads the photos to a service, such as Flickr or Facebook, in order to share them with friends. In this scenario, Alice is the private data owner (PDO), whereas her geo-tagged photos are user-generated content (UGC). Alice *outsources* the management of her UGC to a service provider (SP). She wishes that only her friends, called *trusted query users*, can access the UGC. Anybody else, including the SP, is a potentially malicious *attacker* who may take advantage of the UGC for commercial purposes, e.g., unsolicited advertisements, or criminal purposes, e.g., robbery.

This paper focuses on the outsourcing of spatial datasets. It presents methods that protect *location data* (i.e., a set P of points) from attackers, while allowing trusted query users to issue spatial queries that are executed efficiently by the SP. Non-spatial content (e.g., photo) is assumed to be encrypted with conventional encryption; it can be decrypted by trusted users. The PDO maps P to P' using a transformation with a set of secret parameters that constitute the *key*. The PDO uploads P' to the SP and sends invitations (with the key) to trusted users through a secure channel. The SP does not know the key, so it cannot derive P. To issue a range query q, a trusted user $\mathcal{U}$ maps q to q' by using the key and submits q' to the SP. The SP executes q' against P' and returns the result $R' \subseteq P'$ to $\mathcal{U}$, who uses the key to derive the actual result $R \subseteq P$. The transformed dataset P' must satisfy two important criteria: (*i*) it should be infeasible to reconstruct the precise points of P from P' without the key, and (*ii*) it should support efficient and accurate processing of range queries from trusted users.

A brute-force solution is to apply conventional encryption (e.g., AES) to the dataset P and then store the encrypted file at the server. At query time, the client downloads the entire file, decrypts it, and then searches for the requested results. This solution is secure, but inefficient for typical queries requiring a small fraction of the data.

Most related work [1], [2], [3] in database outsourcing assumes that a tamper-proof device (or trusted software) exists in front of the SP. In our case, it is not feasible for every PDO to install her device at the SP. If query processing requires on-the-fly transformations, these must be done by the user; furthermore, the methods must have low communication cost.

In summary, our first contribution is an attack model for assessing the security of transformations. Our second contribution is the proposal of spatial transformations (i.e., HSD, ERB) and a cryptographic method (i.e., CRT). They cover different tradeoffs between data security and query efficiency.

The rest of the paper is organized as follows. Section II reviews related work. Section III defines formally the problem. Section IV presents an attack model and our spatial transformation methods. Section V presents the cryptographic method. An empirical evaluation is presented in Section VI. Finally, Section VII concludes the paper.

II. Related Work

Hacigümüs et al. [3] addressed the confidentiality of outsourced data by storing at the SP the encrypted tuples, with auxiliary *bucketing* information to facilitate indexing. This technique returns a superset of the actual query results, which calls for expensive filtering. Agrawal et al. [1] pointed out that bucketing is vulnerable to *estimation exposure*, i.e., inferring approximate values of the encrypted data. Motivated by these limitations, an order-preserving encryption scheme (OPES) for 1D numeric values was proposed that transforms the input data distribution (e.g., Zipfian) into a user-specified target distribution (e.g., Gaussian) [1]. This scheme assumes that an attacker cannot stage a known plaintext attack, unlike our attack model in Section IV-A. Damiani et al. [2] propose to build a B$^+$-tree on 1D data and then apply conventional encryption (e.g., AES) on each node. The decryption key is required in order to process queries and a trusted front-end (with tamper-resistant device) at the SP is assumed.

Confidentiality has also been addressed in the context of privacy-preserving publication of sensitive datasets. The k-

anonymity principle [4] generalizes each tuple such that it is indistinguishable from at least $k-1$ other tuples. However, the generalized tuples prevent exact query answers, rendering the data useful only for statistic purposes. In contrast, the users in our setting require exact query results.

In privacy-preserving data mining, a mining task (e.g., classification, clustering) is outsourced to the service provider (SP) for extracting patterns from the *perturbed data*, without revealing the original data. Data perturbation techniques [5] could be used for introducing noise into the data. Unfortunately, these methods do not guarantee accurate region queries over the perturbed data so they are inapplicable to our problem.

In the context of location privacy, users ask for nearby points of interest, without revealing their exact locations to the service. Following the spatial k-anonymity principle, a trusted location anonymizer [6], [7] is employed to maintain the locations of all users. At query time, it generalizes the query user's location into a region before submitting it to the service. Other work on private nearest-neighbor queries [8], [9], [10] adopt alternative privacy definitions other than spatial k-anonymity. All these location privacy solutions are orthogonal to our problem, as they do not protect the points of interest.

A related issue in data outsourcing is the *authentication* of query results, which must be *sound* (the SP does not alter the data), and *complete* (the SP returns all valid results). The *Merkle Hash Tree* (MH-tree) [11] is a main memory binary tree for indexing 1D values, and it has been used [12] for the authentication of range queries. Authentication is orthogonal to our confidentiality problem, and existing authentication schemes can be used on top of our transformations.

III. Problem Setting

This section defines formally the problem of confidential spatial data outsourcing. We focus on 2D point datasets (i.e., common spatial UGC) and the (typical) range query. Given a rectangle $W = [x_l, x_h] \times [y_l, y_h]$ and a set P of points, the *range query* retrieves each $p \in P$ that intersects with W.

We assume the following architecture. In a pre-processing phase, the PDO chooses a secret *transformation key* and converts the original point set P into the transformed point set P'. Next, the PDO builds an index $T(P')$ over P'. The transformed dataset is sent to the SP. The PDO trusts all her friends (i.e., *querying users*) and sends them the transformation key over a secure channel (e.g., SSL). To issue a query q, the user $\mathcal{U}$ encodes q to q' using the key, and sends q' to the SP. The SP evaluates the query over P' and returns the encoded results to $\mathcal{U}$. Finally, $\mathcal{U}$ decodes the results using the key, thus obtaining the original points.

Our objective is to develop transformation techniques that satisfy the following criteria: (i) hardness for an attacker (including the SP) to recover precisely the original dataset P from the transformed dataset P', (ii) low transformation overhead, (iii) efficient support for range query processing at the SP.

IV. Spatial Transformations

We first formulate the notion of *spatial transformation*. For the sake of discussion, we assume that the spatial domain is fixed to the unit square $[0,1]^2$, for both the original point set P and the transformed point set P'. Given a point $p = (x, y)$ in P, we compute its transformed point $p' = (x', y')$ in P', by applying the transformation functions $F_X(\cdot)$ and $F_Y(\cdot)$ to the x and y coordinates, respectively. These functions are associated with a hidden *transformation key*, which will be detailed for each specific transformation technique. The *key length* Υ indicates the number of parameters used in the key.

Section IV-A presents an intuitive attack model (for spatial transformations), by capturing the knowledge available to the attacker. In Section IV-B, we propose a spatial transformation, called HSD. Section IV-C discusses a transformation method, called ERB, that injects errors into the data such that they are reversible with the help of the key.

A. Attack Model

Our attack model is analogous to the *known-plaintext attack* in the cryptography literature, which was not considered in [1]. Even though the attacker has some prior knowledge of the dataset, it is worth preventing the attacker from learning any meaningful information (e.g., distribution, dense regions, outliers) about the remaining data points.

Let P and P' be the original and transformed dataset, respectively. Assume the attacker knows only the following:

- A set $S \subset P$ of m points, $S = \{s_1, s_2, ..., s_m\}$.
- The set $S' \subset P'$ of transformed points, $S' = \{s'_1, s'_2, ..., s'_m\}$, where s'_i is the transformed point of s_i.

Our goal is to protect the original data points. The *general attack* is defined as a transformation-independent method that *estimates* an *approximate* original location of some transformed point in $P' - S'$, based on the known S and S' (but not the key). We define the *feature vector* of a point $p' \in P' - S'$ over S' as:

$$\mathcal{V}(p', S') = \langle dist(p', s'_1), dist(p', s'_2), ..., dist(p', s'_m) \rangle$$

where $dist(\cdot)$ denotes the Euclidean distance. Similarly, for a given location c in the original domain space, its feature vector over S is defined as $\mathcal{V}(c, S)$. We then define the dissimilarity between c and p' as:

$$\Phi(c, p') = L_1\left(\frac{\mathcal{V}(p',S')}{|\mathcal{V}(p',S')|}, \frac{\mathcal{V}(c,S)}{|\mathcal{V}(c,S)|}\right)$$

where L_1 is the Manhattan distance. Based on this concept, the attacker *estimates* the original location of p' as p^*, which is defined as the location c having the smallest $\Phi(c, p')$ value. Since there is no closed form of p^*, we assume that the attacker applies the Monte Carlo method to obtain an accurate approximation of p^*. We quantify the attacker's estimation error using the Euclidean distance $dist(p, p^*)$ between the original point p corresponding to p' and the point p^* determined by the attack.

B. HSD Spatial Transformation

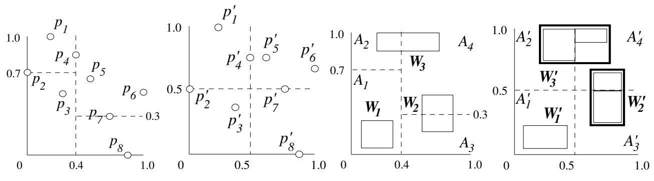
The *hierarchical space-division* (HSD) is a spatial transformation that offers security by equalizing the distribution of points in the transformed space. The intuition is that by approximating a uniform distribution, the attacker cannot obtain any insight into the original data through inspection of the data distribution in the transformed space. HSD achieves this by employing a kD-tree partitioning of the data points.

Data Transformation. Given an integer E, we first construct a kD-tree on the dataset P, but with only the top E tree levels. The resulting tree has $2^E - 1$ nodes, where each node stores its splitting X-value (or Y-value) for the original space. These $2^E - 1$ splitting values constitute the parameters in the transformation key. Figure 1a shows an HSD example with $E = 2$ levels for the original point set P. In the transformed space (in Figure 1b), another kD-tree is built to capture uniform distribution. To transform an original point $p = (x, y)$ into $p' = (x', y')$, it suffices to find the leaf rectangle $A = [Ax_l, Ax_h] \times [Ay_l, Ay_h]$ that contains p in the original tree, and then identify the rectangle $A' = [A'x_l, A'x_h] \times [A'y_l, A'y_h]$ of the corresponding leaf node of the tree in the transformed space. We thus compute p' as:

$$x' = F_X(x) = A'x_l + (A'x_h - A'x_l) \cdot \frac{x - Ax_l}{Ax_h - Ax_l}$$

The value of y' is computed similarly.

Figure 1b depicts the (transformed) locations of points in P'. For instance, the leaf rectangle $A' = (0, 0.5] \times (0, 0.5]$ with the transformed point p'_3 corresponds to the leaf rectangle $A = (0, 0.4] \times (0, 0.7]$ with the original point p_3 (in Figure 1a). Observe that P' is more evenly distributed than P.



(a) original P (b) trans. P' (c) original W_i (d) trans. W'_i

Fig. 1. HSD Transformation Example, $E = 2$

To decode a transformed point $p' = (x', y')$, we also apply the transformation procedure described above. The only difference is that we compute the rectangle A' that contains p' in the transformed space, and then search the tree in order to derive the corresponding rectangle A in the original space.

Query Transformation. To transform a range query W, we follow all branches of the tree intersecting W, thus obtaining multiple leaf rectangles A_i in the original space and corresponding rectangles A'_i in the transformed space. Figure 1c depicts three example queries, W_1, W_2, and W_3, in the original space. The transformed queries are shown in Figure 1d. Leaf rectangles in the original and transformed spaces are labeled A_i and A'_i, respectively. For instance, query W_3 is transformed into two regions in A'_2 and A'_4. Their minimum bounding rectangle (MBR) is taken as the transformed range query W'_3. Some transformed queries may contain extra space, leading to the retrieval of false hits, which can be safely discarded by the user after decoding them.

C. ERB Transformation

Perturbation techniques [5] inject noise into datasets before outsourcing them. However, this process prevents perfect reconstruction of individual original points from the outsourced data points. In the following, we present an error-injection technique that is reversible by the data owner (and users), but computationally hard to compromise for an attacker.

A *secure hash function* (e.g., SHA-512) converts an arbitrary-length plaintext message into a fixed-length digest message (e.g., 512 bits). It is computationally infeasible for the attacker to recover the original message from the digest message. By dividing the digest message with its domain size, we obtain a pseudo-random real number (in $[0, 1]$) derived from the original message. Our *error-based transformation* (ERB) is built on top of such a secure hash function.

Data Transformation. The data owner specifies a transformation key consisting of three parameters: an error threshold $\epsilon \in [0, 1)$ and two (cryptographic) key values $\mathcal{K}_X$ and $\mathcal{K}_Y$. We assume that each point has an (unique) identifier id. Given an original data point $p = \langle id, x, y\rangle$, its transformed point $p' = \langle id, x', y'\rangle$ is computed as follows:

$$x' = (1 - \epsilon) \cdot x + \epsilon \cdot \mathtt{SHA}(\mathcal{K}_X \circ id)$$

where SHA returns a real number in [0,1] and $\circ$ denotes concatenation. The value y' is computed similarly.

Knowing the threshold ϵ and the key values $\mathcal{K}_X$ and $\mathcal{K}_Y$, it is trivial to use the id and transformed location (x', y') of a point p' to reconstruct its original location (x, y).

Query Transformation. Note that any SHA value falls between 0 and 1. To guarantee that range queries are evaluated correctly, an original query $W = [x_l, x_h] \times [y_l, y_h]$ is converted into a transformed query, $W' = [x'_l, x'_h] \times [y'_l, y'_h]$, as follows:

$$x'_l = (1 - \epsilon) \cdot x_l \quad ; \quad x'_h = (1 - \epsilon) \cdot x_h + \epsilon$$

The values of y'_l and y'_h are derived similarly.

Upon receiving the result of the transformed query W' (from the server), the client decodes each transformed result point p' back into p and then checks whether p is an actual result. Observe that there may exist false positives (but not false negatives) among the points returned from the server.

V. Cryptographic Transformation

We proceed to present our *Cryptographic Transformation* (CRT) technique. It employs conventional encryption (e.g., AES) to achieve provable data confidentiality. Spatial information is completely obscured in the transformed data so the general attack is inapplicable to CRT.

CRT is similar to [2] with the following important differences: (i) our approach uses an R*-tree, instead of a B$^+$-tree and (ii) we do not assume the existence of a tamper resistance device at the SP. Thus, a query is evaluated through a distributed, multiple-round protocol between the user and the SP. Figure 2 exemplifies the functionality of CRT. Data points (e.g., a, b, c) are stored in an encrypted index (only the relevant part is shown). To find the result for query q, the encrypted root (node A) is sent to user $\mathcal{U}$, who decrypts A and determines that the MBR of node B intersects q. Then $\mathcal{U}$ retrieves node B from the SP and computes the query result (i.e., point b). Every index node that intersects q must be sent to the user.

The protocol operates in this level-by-level manner and the number of communication rounds equals the tree height.

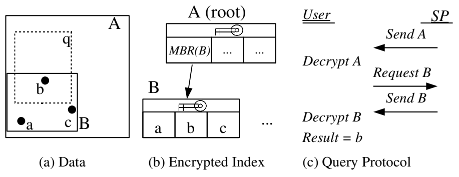


Fig. 2. Query Processing in CRT

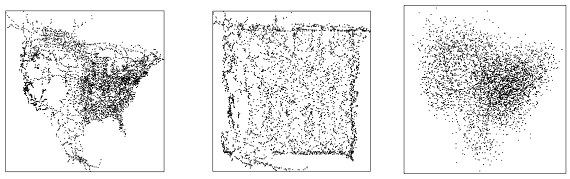
VI. Experimental Study

In addition to our proposed solutions (HSD, ERB, CRT), we introduce two benchmark methods in our comparison. BULK is a secure solution that stores the whole encrypted dataset at the server; when a query is issued, the client retrieves the entire dataset. In contrast, OPT is an insecure solution that stores the original dataset at the server and has optimal communication cost (i.e., no false hits).

We evaluate the above methods using four real spatial datasets [13]: Oldenburg (OL), San Joaquin County (TG), San Francisco (SF), and North America (NA). The domain of each dataset is normalized to the unit square $[0,1]^2$. The key size Υ (default: 64) denotes the number of values in the spatial transformation key (for HSD). ERB requires an error threshold ϵ value (default: 0.15), whereas CRT uses a node capacity parameter (default: 50). Regarding the query distribution, we randomly generate square region queries with a default side length of 5% of the dataspace extent. We assess the query performance in terms of *communication cost* (in KBytes), taken as the average over 100 experimental instances.

The general attack is applicable to HSD and ERB, but not CRT. In the attack scenario, the attacker knows a random subset $S \subset P$ of original points and its transformed set $S' \subset P'$. By default, we set $|S| = 10$, and the covering radius r (of S) to be 0.25. We measure security in terms of the *attacker's estimation error*, as mentioned in Section IV-A.

Visualization of Spatial Transformation. We visualize the proposed transformations using the real dataset NA. Figure 3a shows the original point set. Figure 3b depicts the data transformed by HSD, with $\Upsilon = 64$, i.e., $E = 6$. The transformed data distribution is completely different compared to the original one. Figure 3c illustrates the data transformed by the ERB, at $\epsilon = 0.2$. The distortion achieved is weaker than that of HSD. Nevertheless, particular characteristics of the original dataset are blurred, e.g., the 'gap regions' in the West, the exact densities of dense regions, and outliers.



(a) original points (b) HSD (c) ERB

Fig. 3. Visualization of North America Points

Effect of Data Distribution. Figure 4 shows the communication cost of the methods and the attacker error for different datasets. Clearly, the cost of our methods (HSD, ERB, CRT) is much lower than that of the baseline solution BULK. ERB is rather expensive because the amount of query expansion heavily depends on the error threshold ϵ. CRT incurs the overhead of transferring the intermediate nodes; nevertheless, this overhead becomes reasonable for larger datasets (e.g., SF, NA), where CRT incurs a communication cost similar to HSD.

Figure 4 shows the attacker estimation error for all datasets; the higher this value, the better. The theoretical lower bound of the maximum possible estimation error is 0.707 (i.e., the estimated location is the center of the space). HSD achieves considerable estimation error, which is one-fourth of 0.707. ERB incurs an error value close to ϵ.

Data	**Communication cost (KBytes)**					**Attacker error**	
	BULK	OPT	HSD	ERB	CRT	HSD	ERB
OL	119.23	0.28	0.38	6.12	2.91	0.168	0.151
TG	356.69	1.02	2.22	19.84	4.95	0.204	0.139
SF	3417.10	8.73	16.59	182.52	14.83	0.170	0.152
NA	3433.84	11.62	16.38	215.66	18.45	0.135	0.135

Fig. 4. Communication cost and attacker error, default setting

VII. Conclusion

Social networking and content sharing services allow subscribers to share private spatial data (e.g., geo-tagged photos) with trusted query users through an untrusted service provider. The paper presents methods to encode a dataset such that only trusted users can access the content, while the service provider blindly evaluates queries, without seeing the actual data. We developed spatial transformations (HSD and ERB) and studied their security using a general attack model. We also proposed a cryptographic method (CRT), which is computationally secure, but it incurs multi-round latency for queries.

References

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order-Preserving Encryption for Numeric Data," in *SIGMOD*, 2004.
[2] E. Damiani, S. D. C. Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, "Balancing Confidentiality and Efficiency in Untrusted Relational DBMSs," in *CCS*, 2003.
[3] H. Hacigümüs, B. R. Iyer, C. Li, and S. Mehrotra, "Executing SQL over Encrypted Data in the Database-Service-Provider Model," in *SIGMOD*, 2002.
[4] P. Samarati, "Protecting Respondents' Identities in Microdata Release," *IEEE TKDE*, vol. 13, no. 6, pp. 1010–1027, 2001.
[5] R. Agrawal and R. Srikant, "Privacy-Preserving Data Mining," in *SIGMOD*, 2000.
[6] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing Location-Based Identity Inference in Anonymous Spatial Queries," *IEEE TKDE*, vol. 19, no. 12, pp. 1719–1733, 2007.
[7] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The New Casper: Query Processing for Location Services without Compromising Privacy," in *VLDB*, 2006.
[8] A. Khoshgozaran and C. Shahabi, "Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy," in *SSTD*, 2007.
[9] M. L. Yiu, C. S. Jensen, X. Huang, and H. Lu, "SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services," in *ICDE*, 2008.
[10] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. L. Tan, "Private Queries in Location Based Services: Anonymizers are not Necessary," in *SIGMOD*, 2008.
[11] R. C. Merkle, "A Certified Digital Signature," in *CRYPTO*, 1989.
[12] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, "Spatial Outsourcing for Location-based Services," in *ICDE*, 2008.
[13] T. Brinkhoff, "A Framework for Generating Network-based Moving Objects," *GeoInformatica*, vol. 6, no. 2, pp. 153–180, 2002.