Exploring Privacy Preservation in Outsourced K-Nearest Neighbors with Multiple Data Owners

Frank Li[†]   Richard Shin[†]   Vern Paxson[†*]

[†]University of California, Berkeley   [*]International Computer Science Institute

{frankli, ricshin, vern}@cs.berkeley.edu

ABSTRACT

The k-nearest neighbors (k-NN) algorithm is a popular and effective classification algorithm. Due to its large storage and computational requirements, it is suitable for cloud outsourcing. However, k-NN is often run on sensitive data such as medical records, user images, or personal information. It is important to protect the privacy of data in an outsourced k-NN system.

Prior works have all assumed the data owners (who submit data to the outsourced k-NN system) are a single trusted party. However, we observe that in many practical scenarios, there may be multiple mutually distrusting data owners. In this work, we present the first framing and exploration of privacy preservation in an outsourced k-NN system with multiple data owners. We consider the various threat models introduced by this modification. We discover that under a particularly practical threat model that covers numerous scenarios, there exists a set of adaptive attacks that breach the data privacy of any exact k-NN system. The vulnerability is a result of the mathematical properties of k-NN and its output. Thus, we propose a privacy-preserving alternative system supporting kernel density estimation using a Gaussian kernel, a classification algorithm from the same family as k-NN. In many applications, this similar algorithm serves as a good substitute for k-NN. We additionally investigate solutions for other threat models, often through extensions on prior single data owner systems.

Keywords

Privacy-Preserving; Nearest Neighbors; k-NN; Kernel Density Estimation; Outsourced Cloud Computation

1. INTRODUCTION

The k-nearest neighbors (k-NN) classification algorithm has been widely and effectively used for machine learning applications. Wu *et al.* categorized it as one of the top 10 most influential data mining algorithms [24]. K-NN identifies the k points nearest to a query point in a given data set,

and classifies the query based on the classifications of the neighboring points. The intuition is that nearby points are of similar classes. K-NN classification benefits from running over large data sets and its computation can be expensive. These characteristics make it suitable to outsource the classification computation to the cloud. However, k-NN data is often sensitive in nature. For example, k-NN can be applied to medical patient records, census information, and facial images. It is critical to protect the privacy of such data when outsourcing computation.

The outsourced k-NN model focuses on providing classification, rather than training. Hence, it is assumed that the algorithm parameters have been adequately chosen through initial investigation. Prior work on preserving privacy in such a model uses computation over encrypted data [23, 31, 8, 25]. These works have all assumed a single trusted data owner who encrypts her data before sending it to a cloud party. Then queriers can submit encrypted queries to the system for k-NN classification. In this setting, we would like to keep data private from the cloud party and queriers, and keep queries private from the cloud party and the data owner. In some existing works, the queriers share the secret (often symmetric) data encryption keys. In others, the querier interacts with the data owner to derive the encryption for a query without revealing it.

From these existing models, we make a simple observation with non-trivial consequences: the data owner may not be completely trusted. In all previous models, trust in the data owner is natural since only the owner's data privacy is at risk. However, this assumption does not hold in some practical scenarios:

- Multiple mutually-distrusting parties wish to aggregate their data for k-NN outsourcing, without revealing data to each other. Since k-NN can perform significantly better on more data, all parties can benefit from improved accuracy through sharing. As an example, hospitals might contribute medical data for a k-NN disease classification study or a service available to doctors. However, they do not want to release their patient data in the clear to each other or a cloud service provider. In some cases, these data owners may act adversarially if that allows them to learn the other owners' data.

- The k-NN outsourced system allows anyone to be a data owner and/or querier. This is a generalization of the previous scenario, and a privacy-preserving solution allows data owner individuals to contribute or participate in a system directly without trusting any

other parties with plaintext data. Some potential applications include sensor information, personal images, and location data.

These scenarios involve data aggregated from multiple owners, hence we term this the *multi-data owner outsourced* model. In this paper, we provide the first framing and exploration of privacy preservation under this practical model. Since these multi-data owner systems have not yet been used in practice (perhaps because no prior works address privacy preservation), we enumerate and investigate variants of a privacy threat model. However, we focus on one variant that we argue is particularly practical and covers a number of interesting cases. Under this model, we discover a set of adaptive privacy-breaching attacks based purely on the nature of how k-NN works, regardless of any system protocol and encryption designs.

To counter these privacy attacks, we propose using kernel density estimation with a Gaussian kernel instead of k-NN. This is an alternative algorithm from the same class as k-NN (which we can consider as kernel density estimation with a uniform kernel). While this algorithm and k-NN are not equivalent, we demonstrate that in many applications the Gaussian kernel should provide similar accuracy. We construct a privacy-preserving scheme to support such an algorithm using partially homomorphic encryption and garbled circuits, although at a computational and network cost linear to the size of the data set. Do note that k-NN itself is computationally linear. While this system may not yet be practical for large data sets, it is both proof of the existence of a theoretically secure alternative as well as an first step to guide future improvements.

In summary, our contributions are:

- We provide the first framework for the k-NN multi-data owner outsourced model.

- We explore privacy preservation of k-NN under various threat models. However, we focus on one threat model we argue is both practical and covers several realistic scenarios. We discover a set of privacy-breaching attacks under such a model.

- We propose using kernel density estimation with a Gaussian kernel in place of k-NN. We describe a privacy-preserving construction of such a system, using partially homomorphic encryption and garbled circuits.

2. BACKGROUND

2.1 Nearest Neighbors

K-nearest neighbors (k-NN) is a simple yet powerful non-parametric classification algorithm that operates on the intuition that neighboring data points are similar and likely share the same classification. It runs on a training set of data points with known classifications: $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ is the i-th data point and the label $y_i \in \{1, \cdots, C\}$ indicates which of the C classes $\mathbf{x}_i$ belongs to. For a query point $\mathbf{x}$, k-NN determines its classification label y as the most common label amongst the k closest points to $\mathbf{x}$ in $\mathcal{D}$. Closeness is defined under some distance metric (such as Manhattan or Euclidean distance for real-valued vectors). Since all of k-NN's computation is at classification time, instead of training time, k-NN is an example of a lazy learning method.

In the case of 1-NN ($k = 1$), as the number of training data points approaches infinity, the classification error of k-NN becomes bounded above by twice the Bayes error (which is the theoretical lower bound on classification error for any ideal classification algorithm). In general, k-NN benefits greatly from execution over large data sets. Given a single data owner will have a limited amount of data, aggregation of data from multiple parties is desirable for increased accuracy in k-NN applications. Also, since k-NN classification is computationally expensive, it is practical to outsource computation. These characteristics motivate our interest in multi-data owner outsourced k-NN classification.

2.2 Kernel density estimation and regression

K-NN is an example of an *instance-based* learning method as it involves making a prediction directly from existing instances in the training data (as opposed to using a model of the training data, for example). K-NN uses only the k nearest instances to a query and allows each to equally influence the query classification. However, there are other possible rules to decide how much influence each instance should have on the final prediction, known as *kernels*.[1]

Formally, a kernel $K(x)$ is a function which satisfies

$$(1) \int_{-\infty}^{\infty} K(x)dx = 1 \qquad (2) \ K(x) = K(-x).$$

In other words, it integrates to 1 and is symmetric across 0.

Given n samples $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ of a random variable $\mathbf{x}$, one can estimate the probability density function $p(\mathbf{x})$ with a kernel K in the following way, called kernel density estimation:

$$p(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} K(\|\mathbf{x} - \mathbf{x}_i\|)$$

where $\|\cdot\|$ is a norm. Given this estimate, classification can be determined as the most probable class:

$$\mathcal{D}_C = \{x_i \mid (x_i, y_i) \in \mathcal{D}, y_i = C\}$$

$$p(y = C|\mathbf{x}, \mathcal{D}) \propto \sum_{\mathbf{x}_i \in \mathcal{D}_C} K(\|\mathbf{x} - \mathbf{x}_i\|)$$

$$\arg\max_C p(y = C|\mathbf{x}, \mathcal{D}) = \arg\max_C \sum_{\mathbf{x}_i \in \mathcal{D}_C} K(\|\mathbf{x} - \mathbf{x}_i\|)$$

See Appendix A of the full paper version for a more detailed exposition.

Therefore, to classify a particular point $\mathbf{x}$, we can sum the kernel values of the points which belong to each class and determine which class has the largest sum. The following uniform kernel equates this derivation with k-NN:

$$d_{k,\mathbf{x},\mathcal{D}} := \text{distance of } k^{\text{th}} \text{ nearest point from } \mathbf{x} \text{ in } \mathcal{D}$$

$$K(\|\mathbf{x} - \mathbf{x}_i\|; k, \mathcal{D}) = \begin{cases} \frac{1}{2d_{k,\mathbf{x},\mathcal{D}}} & \text{if } \|\mathbf{x} - \mathbf{x}_i\| \leq d_{k,\mathbf{x},\mathcal{D}} \\ 0 & \text{otherwise} \end{cases}$$

$d_{k,\mathbf{x},\mathcal{D}}$ is the *width* of this kernel, as $K(t) > 0$ for $\|t - \mathbf{x}\| \leq d_{k,\mathbf{x},\mathcal{D}}$ and $K(t) = 0$ for $\|t - \mathbf{x}\| > d_{k,\mathbf{x},\mathcal{D}}$. Thus, k-NN classification is kernel density estimation with a uniform finite-width kernel.

---

[1]To disambiguate between the many distinct meanings of "kernel" in machine learning, statistics, and computer science, the kind of kernels in this paper are also called "smoothing kernels".

One can substitute in a different kernel to obtain a classifier which behaves similarly. One example is the Gaussian kernel:

$$K(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

where σ is the standard deviation parameter. Note that this kernel has infinite width, meaning all points have some influence on the classification.

2.3 Paillier Cryptosystem

The Paillier cryptosystem [15] is a public-key semantically secure cryptographic scheme with partially homomorphic properties. These homomorphic properties allow certain mathematical operations to be conducted over encrypted data. Let the public key pk be (N, g), where N is a product of two large primes and g is a generator in $\mathbb{Z}_{N^2}^*$. The secret decryption key is sk. Also let E_{pk} be the Paillier encryption function, and E_{sk}^{-1} be the decryption function. Given $a, b \in \mathbb{Z}_N$, the Paillier cryptosystem has the following properties:

$$E_{sk}^{-1}(E_{pk}(a) \cdot E_{pk}(b) \mod N^2) = a + b \mod N \quad \text{(Add)}$$

$$E_{sk}^{-1}(E_{pk}(a)^b \mod N^2) = a \cdot b \mod N \quad \text{(Mult)}$$

In other words, multiplying the ciphertext for two values results in the ciphertext for the sum of those values, and computing the c-th power of a value's ciphertext will result in the ciphertext for the value multiplied by c.

The Paillier ciphertext is twice the size of a plaintext; if N is 1024 bits, the ciphertext is 2048 bits. A micro-benchmark of Paillier in [17] shows practical performance runtimes: encryption of a 32-bit integer takes 9.7 ms, decryption takes 0.7 ms, and the homomorphic addition operation takes 0.005 ms.

2.4 Yao's Garbled Circuits

Yao's garbled circuit protocol [28, 13] allows a two-party evaluation of a function $f(i_1, i_2)$ run over inputs from both parties, without revealing the input values when assuming a semi-honest adversary model. If Alice and Bob have private input values i_A and i_B, respectively, the protocol is run between them (the input owners) and outputs $f(i_A, i_B)$ without revealing the inputs to any party.

In the protocol, the first party is called the generator and the second party is the evaluator. The generator takes a Boolean circuit for computing f, and generates a "garbled" version GF (intuitively, it is cryptographically obfuscating the circuit logic). Any input i for function f has a mapping to a garbled input for GF, which we will denote as $GI(i)$. The generator gives the garbled circuit GF to the evaluator, as well as the generator's garbled inputs $GI(i_1)$. Since the generator created the garbled circuit, only the generator knows the valid garbled inputs. The evalutor then engages in a 1-out-of-2 oblivious transfer protocol [19, 9] with the generator to obliviously obtain the garbled input values for her own private inputs i_2. The evaluator can now evaluate $GF(GI(i_1), GI(i_2))$ to obtain a garbled output, which maps back to the output of $f(i_1, i_2)$.

There exists efficient implementations of garbled circuits [14, 6, 2, 22], although naturally the garbled circuit's size and evaluation runtime increases with the complexity of the evaluated function's circuit.
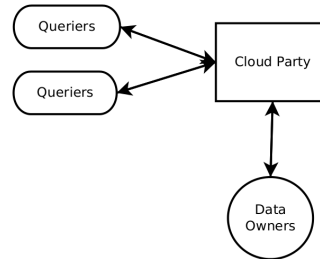

Figure 1: The general model of existing outsourced k-NN system. The data owner is trusted and outsources encrypted data to the cloud party. Queriers request k-NN computation from the cloud party, and are sometimes trusted depending on the prior work. The cloud party executes k-NN classification and is semi-honest.

Prior Work	DO	Q	Cloud
Wong et al [23]	Trusted	Trusted	Semi-honest
Zhu et al [31]	Trusted	Semi-honest	Semi-honest
Elmehdwi et al [8]	Trusted	Semi-honest	Semi-honest
Xiao et al [25]	Trusted	Trusted	Semi-honest

Table 1: A summary of the trust models from prior private outsource k-NN works. DO is the data owner, Q is the querier, and Cloud represents any cloud parties. Semi-honest parties are assumed to follow protocols but may attempt to discover query or data values.

3. RELATED WORK

Prior works have focused on two general approaches to achieving privacy in k-NN: distributed k-NN and outsourced k-NN computed on encrypted data.

In distributed k-NN, multiple parties each maintain their own data set. Distributed computation involves interactions between these parties to jointly compute k-NN without revealing other data values [5, 30, 26, 12, 18, 21, 27, 29]. A general framework for how these systems operate is they iteratively reveal the next closest neighbor until k neighbors have been determined. While allowing multiple parties to include their data sets, these works do not provide a solution for outsourced k-NN because they requires data owners to store and compute on their data. Furthermore, they must remain online for all queries.

The other line of prior privacy-preserving k-NN work relies on computing k-NN over encrypted data. These systems are designed for outsourcing, as depicted in Figure 1. Table 1 summarizes the trust model each work assumes. An important observation is that all prior works assume a trusted data owner.

Wong et al. [23] provides secure k-NN outsourcing by developing an asymmetric scalar-product-preserving encryption (ASPE) scheme. ASPE transforms data tuples and queries with secret matrices that are inverses of each other. Multiplying encrypted tuples and queries cancel the transformations to output scalar products, used for distance comparisons. Hence, a data owner and queriers can upload encrypted data tuples and queries to a cloud party, who can compute k-NN using scalar products. It is worth noting this

encryption scheme is deterministic, so identical data tuples have the same ciphertext and likewise for queries. The secret matrices are symmetric keys for the encryption scheme, and must be shared with both the data owner and queriers. This approach assumes a trusted data owner and queriers, and a semi-honest cloud party who follows the protocols but attempts to learn query or data values.

In the outsourced k-NN system from [31], data encryption is again conducted by a trusted data owner, using a symmetric scheme with a secret matrix transformation as a key. However, queriers do not share this key. Instead, they interact with the data owner to derive a query encryption without revealing the query. Note this requires a data owner to always remain online for all queries. Also, data tuple encryption (being a matrix transformation) is deterministic, but query encryption is not due to randomness introduced during the query encryption protocol. The encryption scheme is designed similiarly to ASPE and preserves distance, so a cloud party can execute k-NN using distances computed from encrypted data tuples and queries. In this system's trust model, the data owner is trusted while the queriers and the cloud party are semi-honest.

The system in [8] is designed to protect data and query privacy with two cloud parties. One cloud party is the data host, who stores all uploaded (encrypted) data tuples. The other cloud party is called the key party, since it generates the keys for a public-key encryption scheme. Data tuples and queries are encrypted with the key party's public key. In this system, Paillier encryption is used for its partially homomorphic properties. For each k-NN execution, the system computes encrypted distances through interactions between the key party and the data host. The key party orders the distances and provides the data host with the indices of the k nearest neighbor data tuples to return to a querier. This work assumes a trusted data owner, and semi-honest queriers and cloud parties.

Instead of finding exact k-NN, [25] allows a cloud party to approximate it using encrypted data structures uploaded by the data owner that represent Voronoi boundaries. The Voronoi boundary surrounding a data point is the boundary equidistant from that data point to neighboring points. The region enclosed in a boundary is the region within which queries will return the same 1-nearest neighbor. Because the Voronoi boundaries change whenever a new data point is added, this approach prevents continuous data uploading without redoing the entire outsourcing process. The encryption scheme is symmetric, and both the data owner and queriers share the secret key. This work's trust model is identical to [23]'s, where the cloud party is semi-honest and the data owner and queriers are fully trusted.

Another contribution from [25] is a reduction-based impossibility proof for privacy preservation in outsourced exact k-NN under a single cloud party model, where the cloud party has access to an encryption key. Note that [23] and [31] assume the cloud party does not have the encryption key, hence avoiding this impossibility argument. The proof is a reduction to order-preserving encryption (OPE), and leverages prior work that shows OPE is impossible under certain cryptographic assumptions [4]. Fully homomorphic encryption does actually allow OPE but it is still impractical [10]. Let B be an algorithm, without access to a decryption oracle, that finds the nearest neighbor of an encrypted query $E(q)$ in the encrypted data $E(D)$. The impossibility proof shows that B can be used to construct an OPE function $\mathcal{E}()$, hence B cannot exist. However, their argument does not apply to a system model with multiple cloud parties. Their proof, which relies on B's lack of access to a decryption oracle, can be circumvented by providing access to the decryption function at another cloud party, such as in [8]. Furthermore, OPE has been realized in an interactive two-party setting [16]. As later discussed in Section 5 and 8, we consider it reasonable that a cloud party has encryption capabilities in a multi-data owner model. Thus further exploration of private k-NN is needed for scenarios not within the scope of this impossibility result.

One important observation about these prior works is that they all exhibit linear complexity in computation (and network bandwidth in the case of [8]). Intuitively, this is because the protocols are distance based. Since the distances from a query to all data tuples varies for each query, all distances must be recomputed per query. While there are techniques [3, 11] for reducing the computational complexity of k-NN, these techniques may not be privacy preserving. For example, these algorithms necessarily compute on only a portion of the data tuples near the query. Observing similar data tuple subsets used can leak the similarity of subsequent queries. However, future work may yield more efficient privacy preserving k-NN constructions.

4. SYSTEM AND THREAT MODELS FOR MULTIPLE DATA OWNERS

All existing private outsourced k-NN systems assume a single trusted data owner entity. In this paper, we are the first to consider multiple mutually distrusting data owner entities. This is a simple and practical extension of existing models, yet has important implications. We explore the various threat models that can arise in such a scenario, but we focus the most attention on one threat model we find particularly realistic. The remaining threat models can be more easily dealt with, for example by extending existing single data owner systems, and will be discussed in detail in Section 8. In this section, we first discuss our model of a multi-data owner system. We then provide a framework for modeling threats in the system.

4.1 K-NN System Model

The privacy-preserving outsourced k-NN model has three types of parties: the data owners who outsource their data, the cloud party(s) who host the k-NN computation, and the queriers requesting k-NN classification. For emphasis, the key difference between our system model and prior models is the existence of multiple data owner entities. An immediate consequence of this modification is seen on the structure of the cloud parties. As discussed in Section 3, results from [25] indicate that privacy-preserving outsourcing of exact k-NN cannot be achieved by a single cloud party without fully homomorphic encryption, which is impractical. Instead, at least two cloud parties should exist, one which stores encrypted data, and one with access to the associated decryption function that acts as a decryption oracle. Hence our system model includes an additional cloud party we call the Cryptographic Service Provider (CSP) who generates a key pair for a public-key encryption scheme, and distributes the public key to data owners and queriers to use for encryption. The cloud party storing the encrypted data, termed the data
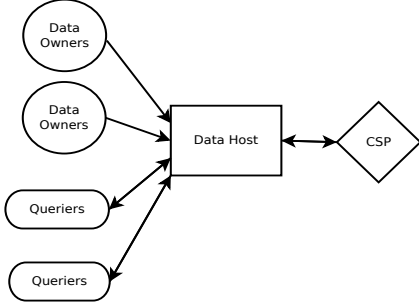
Figure 2: The model of an outsourced k-NN system containing four parties. The cloud parties are the data host and the crypto-service provider (CSP).

host, is able to compute on encrypted data via interaction with the CSP. As depicted in Figure 2, our system model involves multiple data owners and queriers, the data host, and the CSP. Note that encrypted queries and data submissions must be sent to the data host, not the CSP, since the CSP can simply decrypt any received values.

4.2 Threat Model Abstractions

We consider any party as potentially adversarial. Like prior works, we will consider semi-honest adversaries that aim to learn data or query values, possibly through collusion, while following protocols. We do not consider attacks where parties attempt to disrupt or taint the system outputs. Additionally, we must assume that the CSP and the data host do not directly or indirectly collude, since the CSP maintains the secret key to decrypt all data tuples on the data host. This is a reasonable assumption, for example, if the two cloud parties are hosted by different competing companies incentivized to protect their customers' data privacy.

To describe our threat models, we will take a slightly unorthodox approach. Instead of providing a model for each party's malicious actions, we model the malicious behavior of a party based on roles it possesses. The logic behind this approach is that different parties in our system model can pose the same threats because they possess the same roles. Enumerating and investigating all combinations of roles *and* parties is redundant. Hence, just using roles provides a cleaner abstraction from which to analyze threats. Below we describe the four possible roles that arise in our system model.

- Data Owner Role: A party with this role may submit encrypted data into the system. We focus on misbehavior to compromise data privacy, and do not deal with spam or random submissions aimed at tainting system results. Note that the data owner role cannot compromise data privacy by itself since it does not allow observation of system outputs.

- Querier Role: This role allows submission of encrypted queries to receive k-NN classification results.

  We note that the querying ability can allow the discovery of the Voronoi cell surrounding a data point.

The Voronoi cell is the boundary surrounding a point p that is equidistant between p and its neighbors. In the 1-NN case, queries within p's Voronoi cell will return p's classification. A query outside of the cell will return the classification of a neighboring point. Hence, changes in 1-NN outputs can signal a crossing over of a boundary. However, we deem this inherent leakage minimal since discovering the Voronoi cell would require numerous queries to reveal each boundary edge. The Voronoi cell also simply bounds the value of the data point, and does not directly reveal it. Furthermore, neighboring cells of the same class will appear merged in such an attack, since the output signal will not change between cells. K-NN with a large k parameter makes analysis even less accurate.

- Data Host Role: The data host role can only be possessed by a cloud party. It allows storage and observation of incoming encrypted data tuples and queries, and the data host computation that is conducted. A holder of the role also observes any interactions with other parties.

- CSP Role: Only a cloud party may possess this role. A CSP possesses both an encryption and decryption key, and can decrypt any data it observes. It interacts with a data host role to serve as a decryption oracle, and can observe any interaction with other parties, as well as the CSP's own computation.

In Section 5, we focus on the primary threat model in this paper, where any single party can possess both the data owner and querier role. In Section 8, the remaining threat models are explored. These are threats from one of the cloud parties possessing either the data owner role or the querier role, but not both. Also, we consider the case where the cloud parties possess neither roles.

5. ATTACKS IN THE DATA OWNER-QUERIER ROLES THREAT MODEL

In this section, we consider the threat model where an adversarial party possesses both the data owner and querier roles (termed the DO-Q threat model). Since we consider all parties as potentially adversarial, any scenario where both roles may belong to a single party falls under this threat model. We argue this is a realistic threat model, and discover a set of conceptually simple attacks on any system regardless of system design or encryption scheme. The attacks work based purely on the mathematical nature of the k-NN output, rather than implementation specifics. Hence, we conclude that multi-data owner outsourced k-NN cannot be secured under such a threat model.

5.1 DO-Q Threat Model

The DO-Q threat model covers any situation where a single party can possess both the data owner and querier roles. We consider this a practical threat model because it can arise in numerous scenarios. Hence, we will focus on it in this section as well as Section 6. The following outlines the possible scenarios where a single party may possess both roles:

- It is reasonable to expect that data owners, contributing their sensitive data, are allowed access to querying.

If not, there is less incentive for data owners to share data. For examples, hospitals might want to pool their data to create a disease classification system. The doctors at any participating hospital should be able to query the system when diagnosing their patients.

- The data owners may not be explicitly given querying permissions, say if the data owners and queriers are separate institutions. However, miscreants in each party may collude together, providing a joint alliance with both roles.

- If the data host can encrypt data tuples and queries, it can act as its own data owner and querier. It can insert its own encrypted data tuples into the system's data set, and compute k-NN using its own encrypted queries. A data host with access to the encryption keys is not unreasonable. In [8], the data host needs to encrypt values to carry out operations on encrypted data tuples. In addition, the queries are encrypted under the same key as data tuples to allow homomorphic operations.

- If the data host lacked the data owner and/or querier roles (e.g., if an encryption key was kept secret from it), it may still collude with a data owner and/or querier to obtain the roles. This collusion can also occur between the CSP with a data owner and querier.

- If the system is public to data submissions, queries, or both, then any party can supplement their current roles with the public roles. For example, a fully public system allows anyone to obtain both the data owner and querier role.

5.2 Distance-Learning Attacks in the DO-Q Model

We now present a set of attacks that reveal distances between a query and encrypted data tuples, allowing triangulation of the plaintext data values. We begin by presenting attacks under the simpler 1-NN, and gradually progress towards k-NN. We assume that k-NN does not output the neighboring tuples directly, which provides arbitrary privacy breaches through querying, but rather just the query's predicted classification.

We note that prior work [7] has developed algorithms for cloning the Voronoi diagram of 1-NN using only queries if the query responses contain the exact location of the nearest neighbor or the distance and label of the nearest neighbor. If only the nearest neighbor label is returned, then the algorithm provides an approximate cloning. Our attacks differ in that they are structurally very different, we look at k-NN beyond 1-NN, our attacks reveal the exact data value of a target tuple rather than the Voronoi diagram, and our attacks leverage data insertion as well as querying. Also we focus on a system model where query responses do not contain the distance (which we considered an already broken construction). The algorithms in [7] require at least distance in the query response to conduct exact cloning.

5.2.1 Attack using Plaintext Distance

We begin by considering a broken construction of outsourced k-NN. The k-NN system must calculate the distances from all data tuples to the query point. If these distances are ever visible in plaintext to a party with just the querier role, then it is simple to discover the true value of any encrypted data tuple.

Knowing a query q, the adversary can observe the distance l from q to the nearest neighbor. This forms a hypersphere of radius l around q. If the data is d dimensions, $d+1$ different query points with the same nearest neighbor constructs $d+1$ hyperspheres, which will intersect at one unique point, the nearest neighbor's plaintext value. Hence, we can uncover data even though it is encrypted in the data set. Note that in this case, the adversary needs both a querier role as well as the cloud party role that observes plaintext distances. This is different from the threat model we are considering, but we discuss it as it provides insight for the following attacks.

5.2.2 Attack on 1-NN

Now consider the 1-NN scenario where the system does not foolishly reveal the distance in plaintext. An adversary with the data owner and querier roles can still breach the privacy of the system. The querier role allows the adversary to submit a query q, and observe the classification C of its nearest neighbor p. The attacker, using its data owner role, can then insert an encrypted "guess" data tuple $E(g)$, with any classification $C' \neq C$. If the new nearest neighbor classification returns C', we know p is farther away from q than g. If not, then p is closer. Using this changing 1-NN output signal, the adversary can conduct binary search using additional guess tuples to discover the distance from q to p. Hence, the distance is revealed and triangulation can be conducted as in the insecure previous case. This takes $O((d+1)\log D)$ guesses in total to compromise a data tuple, where d is the data dimensionality and D is the distance from q to the initial guess insertion.

Note the above attack appears to require tuple deletions or updates, without which guess tuples that are closer than p will disallow continued binary search. Deletions or updates will certainly improve attack performance, and is reasonable to allow in many scenarios (e.g., location data which is constantly changing). However, it is not required. First, the attacker could conduct a linear search, rather than binary, starting with a guess at distance D and linearly decreasing the guess distance until equal to the distance between q and p. Alternatively, a too-near guess still narrows down the range in which p's value can be located, and the adversary can restart a search with a new query in that range.

5.2.3 Attack on k-NN

Consider now attacks on k-NN, instead of 1-NN. In this scenario, the k-NN system returns the classes of all k nearest neighbors. Let p of class C be the nearest neighbor to query q. In fact, k-NN can be reduced to 1-NN by inserting $k-1$ data tuples of class $C' \neq C$ adjacent to q. Then, p is the only unknown data tuple whose classification is included in the k-NN output set. The attacker with data owner and querier roles can test guess data insertions of class C' until C is no longer included in the k-NN output, indicating the guess tuple is closer than p. This signal can be used as before to conduct a distance search.

5.2.4 Attack on Majority-Rule k-NN

The final scenario for a k-NN system is one that operates on majority rule. Rather than outputting all classifications from the k nearest neighbors, only the most common classifi-

cation is returned. Again, data owner and querier privileges allow a distance-learning attack by reducing to 1-NN. Let p of class C be the nearest neighbor. The attack can insert $k-1$ data tuples adjacent to a query q, split evenly over all classes such that the output of k-NN solely depends on the classification of p. For example, if we have binary classification (0 and 1) and 3-NN, the attacker can insert a 0-class tuple and a 1-class tuple adjacent to q. The original nearest neighbor (p) now solely determines the output of 3-NN, and the adversary may again use a change in k-NN output as a signal for a distance search.

5.3 Fundamental Vulnerability

The vulnerability exposed by these attacks is fundamental to exact k-NN. For any given query, k-NN outputs a function evaluated on the subset of tuples within a distance d, where d is the distance of the k-th nearest neighbor. Representing k-NN as kernel density estimation using a uniform kernel, d is the kernel's width. The vulnerability above fundamentally relies on the fact that the kernel's width changes depending on which data points are near the query. An adversary can learn distances through guess insertions and an observation of k-NN output change. Hence, in any scenario where a party possesses both the data owner and querier roles, exact k-NN cannot be secure, regardless of implementation details. An alternate or approximate solution must be used. In Section 6, we propose a privacy-preserving scheme using a similar algorithm from the same family as k-NN.

6. PRIVACY-PRESERVING KERNEL DENSITY ESTIMATION

In Section 5, we demonstrated that data privacy can be breached in any multi-data owner exact k-NN system under the DO-Q threat model. Given the practicality of this scenario, particularly since the data host typically will have those roles, we seek to provide a secure alternative to exact k-NN. In this section, we propose a privacy-preserving system using an algorithm from the same family as k-NN.

In particular, k-NN is a specific case of kernel density estimation (as discussed in Section 2.2). Intuitively, the kernel measures the amount of influence a neighbor should have on the query's final classification. For a given query, kernel density estimation computes the sum of the kernel values for all data points of each class. The classification is the class with the highest sum. K-NN uses a kernel that is uniform for the k nearest neighbors, and zero otherwise. We propose substituting this kernel with another common kernel, the Gaussian kernel:

$$K(\|\mathbf{q} - \mathbf{x}_i\|) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}\|\mathbf{q}-\mathbf{x}_i\|^2}$$

In this equation, $\mathbf{q}$ is the query tuple, $\mathbf{x}_i$ is the i-th data tuple, σ is a parameter (chosen to maximize accuracy using standard hyperparameter optimization techniques, such as cross-validation with grid search), and $\|\cdot\|$ is the L2 norm. As the distance between q and x_i increases, the kernel value (and influence) of x_i decreases. K-NN allows the k nearest neighbors to all provide equal influence on the query point's classification. When using a Gaussian kernel, all points have influence, but points farther away will influence less than points nearby. While this approach is *not* equivalent to exact k-NN due to the non-uniformity of neighbor influences, it is similar and from the same family of algorithms. Later,

we show experimentally that it can serve as an appropriate substitute in order to provide privacy preservation.

6.1 Why the Gaussian Kernel?

The Gaussian kernel for kernel density estimation is specifically chosen to provide defense against the adaptive k-NN distance-learning attacks under the DO-Q threat model (Section 5). Here, we briefly provide the intuition for why this is true, with the formal proof in Section 6.3.

In k-NN, the width of the kernel depends on the distance of the kth nearest point from the query, so that only the k nearest neighbors influence the classification. An adversary with the data owner role can change this set of neighbors by inserting data tuples within the kernel width, and manipulate whether a particular point influences the classification. Distances can be learned by using changes in this subset as a signal. When using the Gaussian kernel, which has an infinite kernel width, any data the adversary inserts will not change the influence of other points, and the outcome will be affected by exactly an amount the adversary already can compute. For example, for a given query q, an adversary inserting a point y knows the influence for y's class will increase by exactly $K(\|\mathbf{q} - \mathbf{y}\|)$. Hence, the adversary learns nothing new from possessing the data owner role and querying.

It is important to realize that this security arises from the Gaussian kernel's unvarying and infinite width. The choice of an alternative kernel is non-trivial and must be carefully selected. For example, another viable substitute is the logistic kernel $K(u) = \frac{1}{e^u + 2 + e^{-u}}$, since it too has an unvarying and infinite width. Our decision to use the Gaussian kernel was based on the ease of developing a scheme using partially homomorphic cryptographic primitives.

6.2 A Privacy-Preserving Design

In this section, we will step-by-step describe the construction of a privacy-preserving classification system using kernel density estimation with a Gaussian kernel. As we will demonstrate, our protocols provide classification without leaking data or queries. Our system will follow the same system model as described in Section 4.1.

6.2.1 Setup

Recall that the system model provides outsourcing data storage and computation to a data host and a cryptographic service provider. The cryptographic service provider generates a public/private key pair for the Paillier cryptosystem and distributes the public key to data owners and queriers, who use it to encrypt data they submit to the cloud, as well as the data host. Data owners submit their data in encrypted form to the data host. Note that while we use the Paillier cryptosystem, other similarly additive homomorphic cryptosystems may be appropriate as well.

6.2.2 Computing Squared Distances

Since a kernel is a function of distance, our system must be able to compute distance using encrypted data without revealing it. Algorithm 1 describes $SquaredDist(E(a), E(b))$, which allows the data host to compute the encrypted squared distance between a and b given only their ciphertexts. The protocol does not leak the distance to either cloud parties, to prevent a distance-learning attack. Only

Algorithm 1 $SquaredDist(E(a), E(b))$: Output $E(\|a - b\|^2)$

1. DH: **for** $1 \leq i \leq m$ **do:**
   (a) $x_i \leftarrow E(a_i) \cdot E(b_i)^{-1}$. Thus, $x_i = E(a_i - b_i)$.
   (b) Choose random $\mu_i \in \mathbb{Z}_N$.
   (c) $y_i \leftarrow x_i \cdot E(\mu_i)$, such that $y_i = E(a_i - b_i + \mu_i)$.
   (d) Send y_i to CSP.
2. CSP: **for** $1 \leq i \leq m$ **do:**
   (a) $w_i \leftarrow E((E^{-1}(y_i))^2)$.
   (b) Send w_i to DH.
3. DH: **for** $1 \leq i \leq m$ **do:**
   (a) $z_i \leftarrow w_i \cdot x_i^{-2\mu_i} \cdot E(-\mu_i^2)$.
4. DH: Encrypted squared distance $E(\|a - b\|^2) = \Pi_{i=1}^m z_i$.

squared distances are required as they are used in the Gaussian kernel calculation.

Assume our tuples are m dimensions, and a_i is the i-th feature of tuple a. E and E^{-1} are the Paillier encryption and decryption functions, respectively, using the previously-chosen public/private key pair. Note all Paillier operations are done modulo the Paillier parameter N, but for simplicity we elide the modulus. Also, we abbreviate the data host as DH, and the cryptographic service provider as CSP.

Step 1 computes the encryption of $d_i = (a_i - b_i)$ and additively masks each with a random secret μ_i, such that $E^{-1}(y_i) = d_i + \mu_i \bmod N$. This prevents the CSP from learning d_i in step 2. The CSP decrypts y_i, squares it, and sends it back to the DH the encryption of the square. Note $w_i = E((E^{-1}(y_i))^2) = E(d_i^2 + 2\mu_i d_i + \mu_i^2)$. The DH can encrypt μ_i^2 and compute $E(2\mu_i d_i)$ as $x_i^{2\mu_i}$, allowing it to recover $E(d_i^2)$ in step 3. Finally, the DH can compute $E(d^2) = E(\Sigma_{i=1}^m d_i^2) = \Pi_{i=1}^m E(d_i^2)$, as in step 4. This provides the DH with encrypted squared distances without revealing any information to either cloud parties.

6.2.3 Computing Kernel Values

Now that we can compute squared distances, we must compute the Gaussian kernel values in a secure fashion. Algorithm 2 does so such that the CSP obtains a masked kernel value for each data tuple.

Algorithm 2 $KernelValue(q)$: Compute Gaussian kernel values for data tuples t in data set D given query q

1. DH: **for** $1 \leq i \leq |D|$ **do:**
   (a) $s_i \leftarrow SquareDist(t_i, q) = E(\|t_i - q\|^2)$
   (b) Choose random $\mu_i \in \mathbb{Z}_N$
   (c) $e_i \leftarrow s_i \cdot E(\mu_i)$. Thus, $e_i = E(\|t_i - q\|^2 + \mu_i)$.
   (d) Send e_i to CSP.
2. CSP: **for** $1 \leq i \leq |D|$ **do:**
   (a) $g_i \leftarrow \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2} E^{-1}(e_i)}$

In step 1, the DH additively masks each computed encrypted squared distance by adding a different random mask. These values are sent to the CSP who in step 2 decrypts and computes the kernel values using the masked squared distances. Note that since each distance is masked with a random value, the CSP does not learn any distances, even if the CSP knows some data values in the data set. The CSP will not be able to determine which masked values correspond to its known values. Again, no cloud party learns any distances, and they observed only encrypted or masked

values. The query is only used in this algorithm, without ever being decrypted. Hence query privacy is achieved.

6.2.4 Computing Classification

The final step is to conduct kernel density estimation for classification by determining which class is associated with the largest sum of kernel values. Our scheme represents classes not as a single value, but rather as a vector. If there are c classes, we require a set of c orthonormal unit-length bases, where each basis is associated with a particular class. When uploading a data tuple, a data owner uploads the encrypted basis associated with the tuple's class.

We can compute classification as described in Algorithm 3. This algorithm is run after Algorithm 2, such that the CSP knows the masked Gaussian kernel values for all data tuples. Let c_i represent the classification vector for the i-th data tuple.

Algorithm 3 $Classify(q)$: Output a classification prediction for query q where there are c classes

0. DH: Run $KernelValue(q)$
1. DH: **for** $1 \leq i \leq |D|$ **do:**
   (a) Generate a random invertible matrix B_i of size $c \times c$.
   (b) $v_i = B_i \times E(c_i)$.
   (d) Send v_i to CSP.
2. CSP: **for** $1 \leq i \leq |D|$ **do:**
   (a) $w_i = v_i^{g_i}$, where g_i are kernel density values.
   (b) Send w_i to DH.
3. DH: **for** $1 \leq i \leq |D|$ **do:**
   (a) $w_i' \leftarrow B_i^{-1} \cdot w_i^{e^{\frac{1}{2\sigma^2} \mu_i}}$, where μ_i is the random distance masking for the i-th tuple (see Algorithm 2 step 1(c))
4. DH: $A \leftarrow \Sigma_{i=1}^{|D|} w_i'$. Note that this is a $c \times 1$ vector.
5. DH: **for** $1 \leq i \leq c$:
   (a) Choose new $\mu_i \in \mathbb{Z}_N$ randomly.
   (b) $A_c \leftarrow A_c + \mu_i$, where A_c is the c-th component of A.
6. DH: Let $f(in_1, ..., in_c, \mu_1, ..., \mu_c)$
   $$= \arg\max_k (in_k - \mu_k \bmod N).$$
   (a) Generate the garbled circuit GF_f of f and the garbled inputs $GI(\mu_i) \, \forall i \in \{1, ..., c\}$.
   (b) Send GF_f, $GI(\mu_i)$ and $A_i \, \forall i \in \{1, ..., c\}$ to CSP.
7. CSP:
   (a) $A_i' \leftarrow E^{-1}(A_i) \, \forall i \in \{1, ..., c\}$.
   (b) Conduct oblivious transfers with DH to obtain $GI(A_i') \, \forall i \in \{1, ..., c\}$
   (c) $gout \leftarrow GF_f(GI(A_1'), ..., GI(A_c'), GI(\mu_1), ..., GI(\mu_c))$.
   (d) Send $gout$ to DH.
8. DH: Map $gout$ to its non-garbled value out and return to the querier. out is the number representing the predicted class.

In step 1, the encrypted classification vectors are masked through matrix multiplication with a random invertible matrix B_i, which can be efficiently constructed [20]. Note this computation can be conducted on encrypted data because the DH knows the values of B_i, and the matrix multiplication involves only multiplying by plaintext constants and adding ciphertexts. The masked classification vectors are sent to the CSP in step 2, who scales them by the kernel values (from Algorithm 2). In step 3, the DH is returned these values. The DH undoes the transformation by multiplying by B_i^{-1}. The distance masks in the Gaussian kernel

from Algorithm 2 are removed by multiplying by $e^{\frac{1}{2\sigma^2}\mu_i}$, where μ_i is the mask for the i-th tuple. Now, w_i' is the encrypted basis for the i-th tuple's class, scaled by the kernel value. In step 4, the sum of these scaled encrypted bases will sum the kernel values for each class in the direction of the classes' bases, forming the vector A. Note that A is still a vector of encrypted values though.

At this point, we need to determine the class with the highest kernel values summation. Having the CSP decrypt A and return the kernel value sums for the classes may appear straightforward, but in fact leads to an attack on future inserted data tuples if an adversary can view the kernel value sums and can continuously query q. When the next data tuple t is inserted, the adversary will observe an increase in the kernel values summation for t's class by $K(\|t - q\|^2)$. Knowing q and $K(\|t - q\|^2)$, the adversary can compute $\|t - q\|$. Assuming the data has m features, the attacker can determine t using a set of $(m + 1)$ queries both before and after insertion to compute $(m + 1)$ distances. Note that our threat model allows for one of the cloud parties to possess the querying role, so whichever party can observe the kernel value sums can conduct this attack. Thus, the protocol must determine the classification without revealing the kernel value sums to any party.

We accomplish this using a garbled circuit for the function $f(in_1, ..., in_c, \mu_1, ..., \mu_c) = \underset{k}{\arg\max}(in_k - \mu_k \mod N)$. If there are c classes, this function accepts c masked inputs in and c additive masks μ, unmasks each masked input with its associated mask, and returns the index corresponding to the largest unmasked value. If the masked inputs are masked kernel value summations for each class, f returns the class index with the kernel values summation. In step 5, the DH additively masks the (encrypted) kernel values summation for each class. Then in step 6, the DH generates the garbled circuit GF_f for f and garbles its own inputs, which are the masks for each class's kernel values summation. Note that f is not that complex of a function (for example, compared to a decryption function) and its garbled circuit can be practical implemented using existing systems [14]. The DH sends GF_f, its garbled inputs, and the masked (still encrypted) A to the CSP. The CSP decrypts the masked A vector in step 7(a). By conducting oblivious transfer in step 7(b) with the DH to obtain the garbled input associated with each A_i, the CSP does not reveal the masked kernel value sums. With all of the inputs now to GF_f, the CSP evaluates the garbled circuit and returns the garbled output to the DH. Because the DH created the circuit, it can map the garbled output back to its non-garbled value, which is the class index returned by f with the largest kernel values summation. This classification is exactly the classification output of kernel density estimation using a Gaussian kernel.

6.3 Security Analysis

First we must show that use of the Gaussian kernel by our scheme is resistant to the adaptive distance-learning attacks detailed in Section 5. While Section 6.1 provide an intuitive argument for this, here we present a formal proof.

THEOREM 6.1. *Under the Gaussian kernel in kernel density estimation, the adversary learns nothing from the distance-learning attacks of Section 5. More specifically, the adversary does not learn the distance from a query output by adding to the data set (or removing her own data tuple).*

PROOF. Let $D = \{d_0, ..., d_n\}$ be the current data set, q be the adversary's desired query point, and d_A be any data tuple the adversary could insert. Also define $GK_j(D, q)$ to be the kernel value sums for the j-th class given the query q and data set D, and $C(d_i)$ to be the i-th data tuple's classification. Recall that if the system allows for c classes, the query output is $\underset{j\in\{1,...,c\}}{\arg\max}(GK_j(D, q))$.

Before tuple insertion, for each class j in $\{1, ..., c\}$:

$$GK_j(D, q) = \Sigma_{i=1}^{|D|} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(\|d_i - q\|^2)} \mathbb{1}_{C(d_i)=j}$$

Here, $\mathbb{1}_{C(d_i)=j}$ is an indicator variable for whether the i-th data tuple d_i is of class j. After inserting tuple d_A, for each class j in $\{1, ..., c\}$:

$$GK_j(D \cup \{d_A\}, q) = \Sigma_{i=1}^{|D|} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(\|d_i - q\|^2)} \mathbb{1}_{C(d_i)=j}$$
$$+ \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(\|d_A - q\|^2)} \mathbb{1}_{C(d_A)=j}$$
$$= GK_j(D, q) + \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(\|d_A - q\|^2)} \mathbb{1}_{C(d_A)=j}$$

The adversary, through using its data owner role, can cause a change of $\delta = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(\|d_A - q\|^2)} \mathbb{1}_{C(d_A)=j}$. However, the Gaussian kernel values have not changed for any other data tuples. The change in a query output is directly and completely a result of δ, which the adversary knows independent of querying since it knows d_A, q, and σ (which can be public). Given the query output is based only on hidden Gaussian kernel value sums, the output after insertion does not leak any more information on an individual tuple's kernel value beyond what the query output leaks inherently (without insertion). Because the distance is only used in the Gaussian kernel, and the adversary learns nothing about other tuples' Gaussian kernel values through insertion, it does not learn the distance from insertion.

If the adversary deleted one of her tuples, the outcome is exactly the same except δ is negative. With a semi-honest adversary, we can assume the adversary only deletes her own tuples. $\square$

The next analysis is to show our scheme leaks no further data beyond its output. More formally, our scheme leaks no additional information compared to an ideal scheme where the data is submitted to a trusted third party, who computes the same classification output. In our initial discussion of the protocol design in Section 6.2, we justified the security along this front for each step of the construction. See Appendix B of the full paper version for a more formal proof.

A final analysis that could be conducted is to formally characterize what the kernel density estimation output itself reveals about the underlying data. While we proved that the Gaussian kernel output does not leak data tuple values through the distance-learning attacks of Section 5, we have not yet developed a framework to analyze other leakage channels other than the preliminary analysis above. Such a problem is challenging and we leave it to future work. However, even in the face of potential leakage, data perturbation provides some security, although in exchange for output accuracy.

6.4 Performance Considerations

Our kernel density solution requires $O(N)$ computation and communication overhead, where N is the number of data tuples in the data set. On large data sets, we acknowledge this cost might be impractical. Future work remains to find privacy-preserving constructions with optimizations. Our goals with the presented construction are to both demonstrate that there exists a theoretically viable privacy-preserving alternative to k-NN, and lay the groundworks for future improvements. Note that existing private k-NN schemes also have linear performance costs, as discussed in Section 3.

One solution for improving performance is parallelism, since our scheme operates on each data tuple independent of other tuples. Ideally, multiple machines would run in parallel at both cloud parties, providing parallelism not only in computation but also network communication. Each machine at the data host would compute a subcomponent of the kernel value sums, and one master node can aggregate these subcomponents and execute Yao's garbled circuit protocol to produce the classification. Our protocol was intentionally designed so that the function circuit that is garbled is of limited complexity, without need for complex sub-routines such as decryption. Thus, Yao's protocol can be implemented efficiently [14].

K-NN suffers this linear-growth complexity as well, however approximation algorithms significantly improve performance by reducing the number of points considered during neighbor search. For example, k-d trees [3] divide the search space into smaller regions, while locality hashing [11] hashes nearby points to the same bin. Hence, search for a query's neighbors involves computation over the data points in the same region or bin, rather than the entire data set. Unfortunately, it is not obvious how to execute approximate k-NN in a privacy-preserving fashion. Similarly, there must be future work on similar optimizations and approximations for kernel density estimation.

7. COMPARING K-NN AND GAUSSIAN KERNEL DENSITY CLASSIFICATION

In this section, we examine the differences between classification with k-NN and with Gaussian kernel density estimation. We empirically evaluated the performance of using both algorithms, to show they can give similar results on real data. Appendix C of the full paper version expands on this evaluation, discussing a number of additional considerations when using Gaussian kernel density estimation. We construct hypothetical data sets where we can expect them to behave differently, and discuss the underlying causes for

Data Set	k-NN	KDE	Agreement
Cancer 1	97.85%	97.14%	99.28%
Cancer 2	96.49%	96.49%	98.24%
Diabetes	81.82%	74.68%	87.66%
MNIST	97%	96%	99%

Table 2: Empirical results for k-NN versus kernel density estimation (KDE) with a Gaussian kernel. Using several data sets, we evaluate each algorithm's classification accuracy as well as the degree of agreement between the two algorithms' outputs.

the discrepancy. We also discuss some practical issues specific to using a Gaussian kernel and how to solve them.

We summarize the results in Table 2. The medical data comes from the UCI Machine Learning Repository [1], a collection of real data sets used for empirical evaluations of machine learning algorithms. "Cancer 1" and "Cancer 2" contain measurements from breast cancer patients conducted at the University of Wisconsin. "Cancer 1" contains 9 features for each patient; "Cancer 2" came several years after "Cancer 1" and contains 30 features which are more fine-grained. "Diabetes" contains measurements from Pima Indian diabetes patients (8 features per patient). We also use the MNIST handwritten digit recognition data set, which consist of 28×28 pixel grayscale images each containing a single digit between 0 and 9.

Following our recommendations described in Appendix C, we scaled each feature to $[0, 1]$. For the UCI data sets, we used cross-validation to select k and σ; for MNIST, we set them manually to 5 and 0.25. Except for MNIST which already came divided into training and test sets, we randomly took 20% of each data set for testing. Table 2 contains the prediction accuracy and classification agreement for both algorithms on these test sets when using the remaining 80% as training data.

Except for the Pima Indian diabetes data set, Gaussian kernel density classification exhibits an accuracy no more than 1% lower than k-NN's, and the two algorithms agree on over 99% of classifications. In the diabetes case, both algorithms perform poorly because the data is not well separated, with regions scattered with both classes. We argue such a data set is not well suited for k-NN in the first place. Prior investigation must decide whether a particular algorithm is suitable for use.

8. ALTERNATIVE THREAT MODELS

Sections 5 and 6 focused on the DO-Q threat model since it accounts for a number of realistic scenarios, including various collusion relationships. Without any party (directly or through collusion) possessing both the data owner and querier roles, there are several remaining threat situations. In this section, we explore these threat models. The one pervasive assumption remains: the data host and the cryptographic service provider (CSP) do not collude. As mentioned in Section 4.2, a data owner or querier role itself does not allow privacy compromise of the system. Hence, the remaining alternate threat models are outlined below:

- Each party possesses only its original role (e.g., a data host only possesses a data host role).

- The data host additionally obtains a data owner role.

- The data host additionally obtains a querier role.

- The CSP additionally obtains a data owner role.

- The CSP additionally obtains a querier role.

Note that a given system may be under more than one of the above threats. For example, the CSP and data host may both obtain an additional role. However, since we assume no collusion directly or indirectly between the two cloud parties, we can analyze each threat independently.

These threat models, while each covering fewer scenarios, are still realistic. There are settings where the cloud parties

do not collude with certain other parties. For example, the cloud parties may have reputations to maintain, and may avoid collusion or misbehavior for fear of legal, financial, and reputational backlash from discovery through security audits, internal and external monitoring, whistleblowers, or investigations. Note that in all cases, our kernel density estimation solution using the Gaussian kernel can still be used as an appropriate alternative, since the DO-Q threat model allows for a strictly more powerful (in terms of roles) adversary. Our discussion that follows will look at other solutions though, under each threat model.

8.1 Original Roles Threat Model

In this threat model, each party only possesses or utilizes the role it was originally designated. This is the simplest of threat models because data owners do not collude with any other parties. Since data owners themselves do not receive output from the k-NN system, they cannot compromise privacy. Hence, we can treat them as trusted, allowing us to revert to single data owner designs [8, 23], except using multiple data owners. These existing systems already provide privacy against cloud and querier adversaries. This scenario could realistically arise if data owners and querier are separate institutions. For example, hospitals could contribute data to a research study conducted at a university, and only the researchers at that university can query the system.

8.2 Data Host Threat Models

We consider now the threat models where the data host obtains the role of a data owner or a querier, but not both. This can be through collusion, or a design giving the data host the ability to insert data (e.g., encrypt data tuples) or initiate queries.

An extension of the system in [8] can protect against a data host with the data owner role. This system uses computation over Paillier ciphertext, similar in flavor to our kernel density estimation approach. Their system consists also of a CSP which they call the key party, and a data host. Their scheme is privacy-preserving in the single data owner model. The data host is allowed to encrypt, and hence possesses the role of a data owner, but all data that flows through the data host is encrypted, including that received from the key party. Given the data host does not know query values, the data host cannot learn anything that it did not already possess prior knowledge about (e.g., data submitted using the data owner role). Hence, we argue this scheme can be simply extended to multiple data owners in this threat model. Do note that this scheme is not secure under the DO-Q threat model since it still computes exact k-NN.

The data host with the querier role appears more problematic due to information leakage. Intuitively, the ability to observe plaintext query outputs can leak the classification of data tuples. In particular, the data host can observe what tuples are used in the output construction, and the query output associates classifications to those tuples. It is challenging to defend against this breach because in outsourced exact k-NN, some cloud party typically must determine the subset of encrypted data tuples associated with the system output. If that party colludes with a querier (or possesses the querying role), the group will be able to associate those tuples with query-returned classifications. In this situation, our Gaussian kernel density algorithm can again provide privacy, since the query output is dependent on the classification over all encrypted tuples. Hence, the query output cannot be associated with any particular set of data tuples.

8.3 Key Party Threat Models

Finally, we consider a key party either colluding with or possessing the role of a data owner or a querier, but not both. Again we will consider extending the Paillier-based scheme in [8]. A key party with either roles should not be able to compromise this system even with multiple data-owners. K-NN computation is done with relations to distances from a query point. Without knowing the query point, the key party who knows values in the data set (through the data owner role) will not be able to associate observed distance-based values with known data. Furthermore, the values it observes should either be masked (as in our scheme), or encrypted (as in [8]). A key party with querying ability will only observe masked or encrypted values, hence it will not be able to learn anything from the computation.

9. CONCLUSION

In this paper, we have presented the first exploration of privacy preservation in an outsourced k-NN system with multiple data owners. Under certain threat conditions, we can extend existing single data owner solutions to secure the multi-data owner scenario. However, under a particularly practical threat model, exact k-NN cannot be secured due to a set of adaptive distance-learning attacks. This highlights the need for investigation into the inherent leakage from outputs of machine learning algorithms. As an alternative solution, we propose use of a Gaussian kernel in kernel density estimation, which is the family of algorithms k-NN belongs to. We present a privacy preserving system that supports this similar algorithm, as well as evidence of its similarity to k-NN. Admittedly, the scheme may not be practical for large data sets, given that the computational and communication complexities scale linearly. Improving performance is a remaining challenge, and we hope this first step lays the groundwork for future optimized privacy-preserving solutions.

10. ACKNOWLEDGEMENTS

11. REFERENCES

[1] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[2] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. In *Proceedings of the IEEE Symposium on Security and Privacy*, SP'13.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM, Sept. 1975.*

[4] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Proceedings of the Annual Conference on Advances in Cryptology*, CRYPTO'11.

[5] M. Burkhart and X. Dimitropoulos. Fast privacy-preserving top-k queries using secret sharing. In *Proceedings of the International Conference on Computer Communications and Networks*, ICCCN'10.

[6] D. Demmler, T. Schneider, and M. Zohner. ABY - a framework for efficient mixed-protocol secure two-party computation. In *Proceedings of the Network and Distributed System Security Symposium*, NDSS'15.

[7] M. Dickerson, D. Eppstein, and M. Goodrich. Cloning voronoi diagrams via retroactive data structures. In *European Symposium on Algorithms*, Lecture Notes in Computer Science. 2010.

[8] Y. Elmehdwi, B. K. Samanthula, and W. Jiang. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *Proceedings of the IEEE International Conference on Data Engineering*, ICDE'14.

[9] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM, June 1985.*

[10] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the ACM Symposium on Theory of Computing*, STOC'09.

[11] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the International Conference on Very Large Data Bases*, VLDB'99.

[12] M. Kantarcioglu and C. Clifton. Privately computing a distributed k-nn classifier. In *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD'04.

[13] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for two-party computation. In *Journal of Cryptography*, April 2009.

[14] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay: a secure two-party computation system. In *Proceedings of the USENIX Security Symposium*, USENIX'04.

[15] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99.

[16] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In *Proceedings of the IEEE Symposium on Security and Privacy*, SP'13.

[17] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the ACM Symposium on Operating Systems Principles*, SOSP'11.

[18] Y. Qi and M. J. Atallah. Efficient privacy-preserving k-nearest neighbor search. In *Proceedings of the International Conference on Distributed Computing Systems*, ICDCS'08.

[19] M. Rabin. How to exchange secrets by oblivious transfer. Technical report, Boston, MA, USA, 1981.

[20] D. Randall. Efficient generation of random nonsingular matrices. Technical report, Berkeley, CA, USA, 1991.

[21] M. Shaneck, Y. Kim, and V. Kumar. Privacy preserving nearest neighbor search. In *Proceedings of the IEEE International Conference on Data Mining Workshops*, ICDM Workshops '06.

[22] E. Songhori, S. Hussain, A.-R. Sadeghi, T. Schneider, and F. Koushanfar. TinyGarble: Highly compressed and scalable sequential garbled circuits. In *Proceedings of the IEEE Symposium on Security and Privacy*, SP'15.

[23] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis. Secure knn computation on encrypted databases. In *Proceedings of the International Conference on Management of Data*, SIGMOD'09.

[24] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst., Dec. 2007.*

[25] X. Xiao, F. Li, and B. Yao. Secure nearest neighbor revisited. In *Proceedings of the IEEE International Conference on Data Engineering*, ICDE'13.

[26] L. Xiong, S. Chitti, and L. Liu. K nearest neighbor classification across multiple private databases. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, CIKM'06.

[27] L. Xiong, S. Chitti, and L. Liu. Preserving data privacy in outsourcing data aggregation services. *ACM Trans. Internet Technol., Aug. 2007.*

[28] A. Yao. How to generate and exchange secrets. In *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*, FOCS'86.

[29] J. Zhan and S. Matwin. A crypto-based approach to privacy-preserving collaborative data mining. In *Proceedings of the IEEE International Conference on Data Mining Workshops*, ICDM Workshops'13.

[30] F. Zhang, G. Zhao, and T. Xing. Privacy-preserving distributed k-nearest neighbor mining on horizontally partitioned multi-party data. In *Advanced Data Mining and Applications*, Lecture Notes in Computer Science. 2009.

[31] Y. Zhu, R. Xu, and T. Takagi. Secure k-nn computation on encrypted cloud data without sharing key with query users. In *Proceedings of the International Workshop on Security in Cloud Computing*, Cloud Computing'13.