Secure kNN Computation on Encrypted Databases

W. K. Wong, David W. Cheung, Ben Kao, and Nikos Mamoulis
Department of Computer Science, The University of Hong Kong
Pokfulam Road, Hong Kong
wkwong2@cs.hku.hk, dcheung@cs.hku.hk, kao@cs.hku.hk, nikos@cs.hku.hk

ABSTRACT

Service providers like Google and Amazon are moving into the SaaS (Software as a Service) business. They turn their huge infrastructure into a cloud-computing environment and aggressively recruit businesses to run applications on their platforms. To enforce security and privacy on such a service model, we need to protect the data running on the platform. Unfortunately, traditional encryption methods that aim at providing "unbreakable" protection are often not adequate because they do not support the execution of applications such as database queries on the encrypted data. In this paper we discuss the general problem of secure computation on an encrypted database and propose a SCONEDB (Secure Computation ON an Encrypted DataBase) model, which captures the execution and security requirements. As a case study, we focus on the problem of k-nearest neighbor (kNN) computation on an encrypted database. We develop a new asymmetric scalar-product-preserving encryption (ASPE) that preserves a special type of scalar product. We use APSE to construct two secure schemes that support kNN computation on encrypted data; each of these schemes is shown to resist practical attacks of a different background knowledge level, at a different overhead cost. Extensive performance studies are carried out to evaluate the overhead and the efficiency of the schemes.

Categories and Subject Descriptors

H.2.7 [**Database Administration**]: Security, integrity, and protection

General Terms

Algorithms, Security

Keywords

Security, kNN, Encryption

SIGMOD'09, June 29–July 2, 2009, Providence, Rhode Island, USA.
Copyright 2009 ACM 978-1-60558-551-2/09/06 ...$5.00.

1. INTRODUCTION

Emerging computing paradigms such as database service outsourcing and utility computing (a.k.a. cloud computing) offer attractive financial and technological advantages. These are drawing interests of enterprises in migrating their computing operations, including DBMS's, to service providers. Nevertheless, many vocal consultants, including Gartner [7], have issued warnings on the security threats in the cloud computing model. Private information, which includes both customer data and business information, should not be revealed to unauthorized parties. In this paper we address a very important problem of security in services outsourcing: the elements of an encryption scheme and the execution protocol for encrypted query processing. More specifically, we study how sensitive data and queries should be transformed in an *encrypted database environment* and how a service provider processes encrypted queries on an encrypted database without the plain data revealed. We call our model of secure query processing SCONEDB (for Secure Computation ON an Encrypted DataBase).

The conventional way to deal with security threats is to apply encryption on the plain data and to allow only authorized parties to perform decryption. Unauthorized parties, including the service provider, should not be able to recover the plain data even if they can access the encrypted database. Some previous works [2, 10, 11] have studied this encryption problem in the outsourced database (ODB) model. However, these studies are restricted to simple SQL operations, e.g., exact match of attribute value in point query [12]; comparisons between numeric values in range query [2]. In practice, users often interact with a database via applications in which queries are not easily expressible in SQL.

Moreover, most of the previous methods were specially engineered to work against one specific attack model. However, the problem should be studied with respect to various security requirements, considering different attacker capabilities. In this paper we focus on k-nearest neighbor (kNN) queries and show how various encryption schemes are designed to support secure kNN query processing under different attacker capabilities. The kNN query is an important database analysis operation, used as a standalone query (e.g., in similarity search applications on top of multimedia databases) or as a core module of common data mining tasks (e.g., classification and clustering).

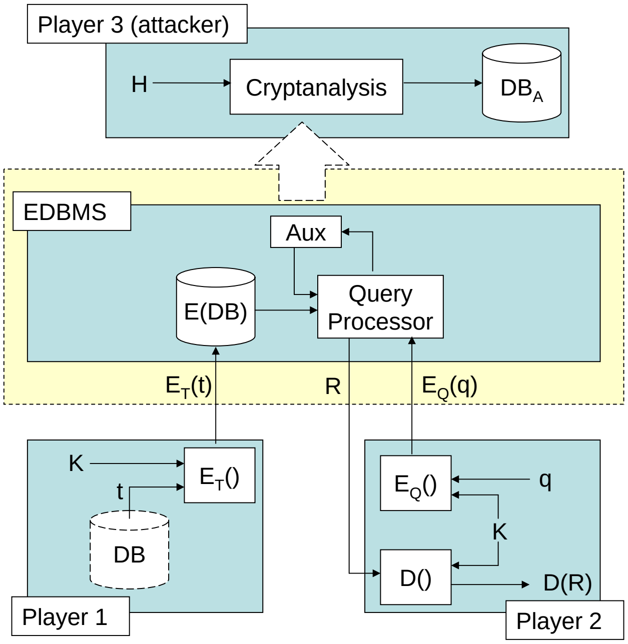

Figure 1: The SCONEDB Model

1.1 The SCONEDB Model

Figure 1 shows our SCONEDB model for secure encrypted database computation. In our model, player 1 is the owner of a database DB on which player 2 wants to execute certain queries. To take advantage of the computational resources of a Service Provider (SP), instead of processing queries locally, player 1 exports DB to an *Encrypted Database Management System* (EDBMS) at which queries submitted by player 2 are processed. (In some applications, player 1 may not even store DB and rely on the EDBMS as the sole repository of the data.) For security, all the tuples in DB are encrypted by player 1 before they are exported to the EDBMS. The EDBMS maintains and processes the encrypted database, $E(DB)$. Likewise, queries submitted by player 2 are also encrypted. EDBMS executes an encrypted query on the encrypted database and returns to player 2 an encrypted result R (e.g., R is a set of encrypted tuples which are the answer of a kNN query). Player 2 applies a decryption function D on R to obtain the plain results.

Under our SCONEDB model, players 1 and 2 have to agree on a security protocol. In particular, they choose a common *encryption scheme*. In SCONEDB, an encryption scheme consists of the following components:

- A *secret key* K. A key K is required as a parameter to the encryption and decryption processes (note that a key may contain a number of components, e.g., RSA requires a pair of numbers as the key). In our model, the key is kept private to players 1 and 2.
- A *database encryption function* $E_T()$. The encrypted database $E(DB)$ is obtained by encrypting each tuple t in DB by $E_T(t, K)$.
- A *query encryption function* $E_Q()$. Each query q is encrypted by $E_Q(q, K)$ before it is submitted to the EDBMS.
- A *result decryption function* $D()$. Each tuple $\hat{t}$ in the encrypted result R is decrypted by $D(\hat{t}, K)$.
- A set of auxiliary operators Aux. An auxiliary operator A_e in Aux operates on the encrypted database to obtain information for the purpose of answering queries. Note that A_e operates without knowing the secret key K and hence has no access to the plain tuples. For example, for kNN queries, A_e may return the Euclidean distance between an encrypted database tuple p and an encrypted query tuple q.

The main goal in the SCONEDB model is to design an encryption scheme in which Aux can operate on $E(DB)$ to support query processing.

1.2 Attack models

In our SCONEDB model, we assume that the EDBMS, which is possibly located at a third party (e.g., a service provider in the *cloud*), is not secure. Therefore, we assume that an attacker (player 3) sees the environment of the EDBMS. In particular, the attacker has accesses to the encrypted database, the encrypted queries, and the encrypted results. Also, we assume that the attacker knows what encryption scheme is being used. That is, he knows all the components of the scheme except the key. These include the encryption and decryption procedures ($E_T()$, $E_Q()$ and $D()$) and the set of auxiliary operators Aux. We assume that the attacker's objective is to recover a plain database $DB_A \subseteq DB$. We assume the attacker is capable to execute PTIME cryptanalysis algorithms with respect to the size of the encrypted database. Our objective is to deny the attacker from obtaining DB_A. Apart from $E(DB)$, the attacker may possess additional knowledge about the original data. To better evaluate the strength of an encryption scheme, we classify attackers into different levels based on the knowledge H they possess.

- Level 1: the attacker observes only the encrypted database $E(DB)$, i.e., $H = \langle E(DB) \rangle$. This corresponds to the ciphertext-only attack (COA) in cryptography [5]. In practice, there are applications accessed by secluded users, for which others can hardly observe any information other than the encrypted data.
- Level 2: Apart from $E(DB)$, the attacker knows a set of plain tuples P in DB but he does not know the corresponding encrypted values of those tuples in $E(DB)$, i.e., $H = \langle E(DB), P \rangle$ where $P \subset DB$. This corresponds to the known-sample attack in database literature [15]. For example, if the attacker observes the encrypted database of a bank and some of his sources are customers of the bank, he then knows the values of several tuples in the plain database.
- Level 3: Apart from $E(DB)$, the attacker observes a set of tuples P in DB and he knows the corresponding encrypted values of those tuples, i.e., $H = \langle E(DB), P, I \rangle$, where $P \subset DB$ and $I(t) = E_T(t, K)$ for all $t \in P$. This corresponds to the known-plaintext attack (KPA) in cryptography [5] or known input-output attack in database literature [15]. For example, if the attacker opens a new account at the bank and observes only one new encrypted tuple afterwards, he can associate the new account's information (unencrypted) with the encrypted value of the new tuple.

A higher-level attack is more powerful than a lower-level one. If an encryption scheme resists a higher level attack, it resists a lower level one as well. Among the 3 attack levels defined, we remark that level-2 attacks capture practical scenarios. This is because in some applications, it is not difficult to observe a small number of plain database tuples (e.g., by artificially inserting "spy" tuples in DB).

We assume the attacker cannot observe the plain queries in all cases. In particular, we do not allow the attacker to disguise as player 2 and submit queries to the database. Note that level-3 attacks are rare in practice, since it is not easy for someone who does not hold the encryption key to associate known plain tuples to their encrypted values.

1.3 kNN on SCONEDB model

In this paper we focus on kNN queries and illustrate how an encryption scheme (which includes the above five components) can be developed to securely support kNN applications under the SCONEDB model. A kNN query searches for k points in a database that are the nearest to a given query point q. Note that each database tuple can be modeled as a multi-dimensional point, if we consider some of its attributes as dimensions and their values as their coordinates. One approach to securely support kNN is to use distance-preserving transformation (DPT) to encrypt data points [20] so that the distance between any two encrypted points in $E(DB)$ is the same as that between the corresponding original points in DB. Given this property, kNN can be computed on the encrypted database. Unfortunately, such transformation is shown to be not secure in practice. If an attacker can access the DPT-encrypted database $E(DB)$ and knows a few points in the plain database DB, he can recover DB entirely [15].

A similar problem on kNN computation at an untrusted platform is studied for location based services (LBS) [8, 17, 9, 13], where users submit queries to an untrusted server which holds the data. The focus of such applications is on protecting the privacy of users (query content), since the database is assumed to be owned by the server [9]. While some studies in LBS also address the privacy of records in the database, k-anonymity is adopted as the standard to protect the database [8, 17]. We remark that k-anonymity has a different security goal compared to our model; k-anonymity aims at preventing an attacker to identify an individual from the database, but the content in the database may be exposed. In addition, most of these models require the existence of a trusted intermediate party (location anonymizer), which handles the data and query transformation. This party, except from being a single point of attack, compromises performance as every query and result has to pass through it. In this paper we seek for alternative encryption schemes that protect data security and at the same time they return accurate kNN results to users.

In the rest of the paper, we study various encryption schemes and analyze their vulnerability to different levels of attacks. In Section 2, we show that if the distance between any two points in the plain database DB can be determined from the points' encrypted values in $E(DB)$ (a property we call *distance recoverability*), then the encryption scheme is vulnerable to level-2 attacks. This observation leads us to develop an *asymmetric scalar-product-preserving encryption* (ASPE) that is not distance-recoverable (Section 3). ASPE can be used to construct a scheme to support kNN computation that resists level-2 attacks. In Section 4, we describe how we can extend this scheme to resist level-3 attacks, however, with additional overhead. Section 5 empirically evaluates the proposed schemes. In Section 6, we briefly discuss how our encryption scheme effectively transforms the kNN problem into a top-k problem, and how that leads to efficient solutions to the problem of secure kNN computation. Section 7 reviews related work. Finally, Section 8 further discusses our SCONEDB model and concludes the paper.

2. DISTANCE-RECOVERABLE ENCRYPTION

In kNN computation, distances between database points to a query point are computed for finding the nearer neighbors to the query point. To solve the secure kNN problem, it is natural to consider adopting an encryption scheme that allows the system to compute $d(p_1, p_2)$ on $E(DB)$ for database points p_1 and p_2 in DB. kNN can then be computed efficiently w.r.t. such a scheme. However, we show in this section that no encryption scheme is secure against level-2 attacks if it allows distance computation as suggested above. We start with the definition of *distance recoverability.*

Definition 1. *(Distance-recoverable encryption (DRE)) Given an encryption function E and a key K, let $E(p, K)$ be the encrypted value of a point p in DB. E is distance-recoverable if and only if there exists a computational procedure f such that $\forall p_1, p_2, K, f(E(p_1, K), E(p_2, K)) = d(p_1, p_2)$.*

A DPT [20] is an example of DRE. This is because, by definition, a DPT preserves distances in the transformed space. Hence, if E is a DPT, we have $d(E(p_1, K), E(p_2, K)) = d(p_1, p_2)$. So, f is simply the Euclidean distance. A DPT transforms the space by rotations and translations. For a point p in DB represented as a column vector, the encrypted value $E(p, K)$ of p w.r.t. a DPT E can be expressed as $Np + t$, where N is a $d \times d$ orthogonal matrix and t is a d-dimensional column vector. Distance between points is preserved, i.e., $d(p_1, p_2) = d(E(p_1, K), E(p_2, K))$. So, DPT supports efficient kNN computations. Here, N and t together form the encryption key K. Regrading level-1 attacks, the attacker cannot recover DB since he does not know N or t [20]. So, DPT is a scheme that resists level-1 attacks. Note that in our model, we assume that the attacker knows E. Therefore, if E is a DRE, we assume that the attacker knows f as well. For example, if E is a DPT, the attacker knows that f is the Euclidean distance function $d()$. However, we will show that DRE, and hence DPT, is not secure under level-2 or level-3 attacks. We first show how to attack DRE at level-3.

Theorem 1. *Assume a DRE E is used to encrypt DB to get $E(DB)$. A level-3 attacker with $H = \langle E(DB), P, I \rangle$ can recover DB if P contains at least $d + 1$ points x_i $(1 \leq i \leq d + 1)$ such that the set of vectors $\{x_j - x_1 | 2 \leq j \leq d + 1\}$ are linearly independent.*

Proof. Since the encryption is a DRE, the distance between any two points p and q, $d(p, q)$, can be computed by the attacker using $f(E(p, K), E(q, K))$. Suppose the attacker wants to find the original value of an encrypted point $y' \in E(DB)$. Let the set of known points in P be $\{x_1, x_2, ..., x_{d+1}\}$ and y be the original value of y' before encryption. He can set up $d + 1$ equations: $d(x_i, y) = f(I(x_i), y')$ for

$i = 1$ to $d+1$. Note that the RHS of the equations are known numeric values to the attacker. Each equation thus represents a d-dimensional hypersphere. The solution of y lies on the intersection of the hyperspheres. Since y exists in the database, a solution must exist. We can show that if the set of vectors $\{x_j - x_1 | 2 \le j \le d+1\}$ are linearly independent, the $d+1$ hypersheres intersect at exactly one point (see Appendix A). So, y can be uniquely determined. Hence the attacker can recover the entire database. □

The level-3 attack shown above is independent of the implementation of DRE. So, no DRE (e.g., DPT) can survive this level-3 attack. Furthermore, we can show that DRE has poor resistance to level-2 attacks by showing that the attacker can "upgrade" his level-2 knowledge to level-3 using *signature linking attack*.

Let us explain signature linking attack. At level-2, $H = \langle E(DB), P\rangle$, the attacker constructs the signature of P by the pairwise distances between every two points in P. Suppose the points in P are ordered and $P = \{x_1, x_2, ..., x_{|P|}\}$. The signature of P, $sig(P)$, is a vector of size ${}_{|P|}C_2$ of the form $(d(x_1,x_2)$, $d(x_1,x_3)$, ..., $d(x_1,x_{|P|})$, $d(x_2,x_3)$, ..., $d(x_{|P|-1},x_{|P|}))$. The attacker tries to find an ordered set of encrypted points Q in $E(DB)$, such that $|Q| = |P|$ and Q gives the same signature as P. Let $Q = \{x'_1, x'_2, ..., x'_{|P|}\}$. $sig(Q)$ is $(f(x'_1,x'_2)$, $f(x'_1,x'_3)$, ..., $f(x'_1,x'_{|P|})$, $f(x'_2,x'_3)$, ..., $f(x'_{|P|-1},x'_{|P|}))$. If there is only one set Q with a matching signature, the attacker can conclude that x'_i is the encrypted value of x_i, i.e., he can construct I with $I(x_i) = x'_i$ for all $x_i \in P$. With this I, $H = \langle E(DB), P, I\rangle$, and the attacker can now carry out a level-3 attack.

The success of *signature linking attack* rests on two issues: (1) Is it easy to find Q? (2) Is it likely that another set, say Q', which is not the transformed set of the points in P, happens to give the same signature? (We call this a signature collision.) For the first issue, we remark that although the search space is huge[1], it can be pruned very effectively. For example, given two encrypted points x'_1 and x'_2 such that $f(x'_1,x'_2) \notin sig(P)$, we know that Q cannot contain both x'_1 and x'_2. For the second issue, we can show that the probability of signature collision is generally very small (see Appendix B). Also, if multiple Qs are found with the same signature as P, the attacker can increase the size of P to lower the likelihood of collision and repeat the attack. In Section 5, we will report experiments for evaluating the feasibility of signature linking attack in terms of the number of points in P required and its computation cost. The results confirms that the attacker can easily recover the database at an affordable cost.

3. ASYMMETRIC SCALAR-PRODUCT -PRESERVING ENCRYPTION

From our discussion, we observe that the weakness of DRE comes from the fact that the attacker is able to recover distance information from the encrypted database. More specifically, given any two points p_1 and p_2 in DB, their distance $d(p1, p2)$ can be determined from their encrypted values $E_T(p_1, K)$ and $E_T(p_2, K)$. These distances allow the attacker to compute signatures and thus to apply the signature linking attack. To resist level-2 attacks, we need an encryption function that does not reveal distance information. For kNN search, we observe that exact distance computation is not necessary. Rather, we only need a distance comparison operation. Given two points p_1, p_2 in DB, we must decide which of the two points is nearer to a query point q. Note that,

$$\begin{aligned} d(p_1,q) &\ge d(p_2,q) \\ \sqrt{||p_1||^2 - 2p_1\cdot q + ||q||^2} &\ge \sqrt{||p_2||^2 - 2p_2\cdot q + ||q||^2} \\ ||p_1||^2 - ||p_2||^2 + 2(p_2 - p_1)\cdot q &\ge 0 \qquad (1) \end{aligned}$$

where $||p||$ represents the Euclidean norm of p, and $\cdot$ represents scalar product. $||p||^2$ can be represented by $p \cdot p$. So, the inequality is decomposed to a number of scalar product computations. This suggests a scalar-product-preserving encryption E_{spp}, i.e., $\forall p_1, p_2 \in DB, p_1 \cdot p_2 = E_{spp}(p_1, K) \cdot E_{spp}(p_2, K)$, for kNN computation. Unfortunately, a scalar-product-preserving encryption is also distance-recoverable and hence is not secure against level-2 attacks.

Theorem 2. *Scalar-product-preserving encryption is distance-recoverable.*

Proof. Let p'_1 (resp. p'_2) be the encrypted point of p_1 (resp. p_2) in DB. We define the function f by

$$f(p'_1, p'_2) = \sqrt{p'_1\cdot p'_1 - 2(p'_1\cdot p'_2) + p'_2\cdot p'_2} \qquad (2)$$

Since the encryption preserves scalar product, we have RHS $= \sqrt{p_1\cdot p_1 - 2(p_1\cdot p_2) + p_2\cdot p_2} = d(p_1,p_2)$. □

There are three types of scalar products: (type-1) scalar product of a database point with itself (e.g., $||p||^2$); (type-2) scalar product of a database point with the query point; (type-3) scalar product of two different database points p_1 and p_2. We observe that Eq. 1 consists of type-1 and type-2 products but not type-3 products, which are essential in Eq. 2. If an encryption preserves only type-1 and type-2 products but not type-3 products, then we can compare the distances $d(p_1,q)$ and $d(p_2,q)$ by Eq. 1, but the attacker cannot recover distances using Eq. 2. Furthermore, for each point p in the database, its corresponding type-1 scalar product, $||p||^2$, is fixed. Hence, if player 1 pre-computes all type-1 scalar products and make them available (e.g., by storing them in the database) for kNN query processing, then the encryption needs not preserve type-1 products either. In summary, we need an encryption that preserves type-2 but not type-1 or type-3 scalar products. With the pre-computed type-1 scalar products, we can verify the inequality of Eq. 1 on the encrypted database to implement the distance comparison operation. Since the encryption does not preserve type-1 or type-3 products, it is not distance-recoverable by design. The encryption is thus resilient to level-2 attacks.

Definition 2. *(Asymmetric scalar-product-preserving encryption (ASPE))*

Let E be an encryption function and $E(p, K)$ be the encrypted value of a point p given a key K. E is an ASPE if and only if E preserves type-2 scalar products but not the other two types, i.e.,

(i) $p_i \cdot q = E(p_i, K) \cdot E(q, K)$ for any p_i in DB and any query point q and

(ii) $p_i \cdot p_j \ne E(p_i, K) \cdot E(p_j, K)$ for any p_i and p_j in DB.

[1] If there are n points in the database, there are ${}_nP_{|P|}$ candidate Q sets to examine.

In Definition 2, we require that the encrypted value of a query q not equal to that of any point p_j in DB, even when $q = p_j$. This suggests that query points and database points should be encrypted differently. That is the encryption functions $E_T()$ and $E_Q()$ in the encryption scheme should be different.

The scalar product of p and q (represented by column vectors) can be represented as $p^T I q$, where p^T is the transpose of p and I is a $d \times d$ identity matrix. I can be decomposed to MM^{-1} for any invertible matrix M, i.e., $p^T q = (p^T M)(M^{-1} q)$. If we set $p' = E_T(p, K) = M^T p$ (resp. $q' = E_Q(q, K) = M^{-1} q$), it is not possible for one to determine the value of p (resp. q), from p' (resp. q') without knowing M. Also, $p'^T q' = p^T M M^{-1} q = p^T q$, i.e., type-2 scalar product is preserved. Suppose p_1' and p_2' are the encrypted points of p_1 and p_2 in DB respectively, then $p_1'^T p_2' = p_1^T M M^T p_2$, which is not equal to $p_1^T p_2$ in general. Type-1 and type-3 scalar products are therefore not preserved. So, we can implement ASPE by using M and M^{-1} as the transformations for database points and queries, respectively.

3.1 A Secure Scheme Against Level-2 Attacks

We have described a special encryption function ASPE that preserves type-2 scalar products; together with the pre-computed type-1 scalar products, we can perform distance comparisons to find the neighbors of a query point. However, if the type-1 product $||p||$ is revealed to the attacker, he knows that p lies on a hypersphere that is centered at the origin with a radius $||p||$. Although the exact location of p is unknown, the information revealed partially compromises security. In this section, we show how we hide this information by "encrypting" $||p||$ and how the EDBMS computes kNN on such encrypted data.

Our idea is to treat a pre-computed type-1 scalar product $||p||^2$ as the (d+1)-st dimension of the point p. More specifically, given a (d-dimensional) database point p, we create a (d+1)-dimensional point $\hat{p}$. The first d dimensions of $\hat{p}$ are those of p, and the (d+1)-st dimension of $\hat{p}$ is set to $-0.5||p||^2$. (We multiply $||p||^2$ with this factor to facilitate distance comparisons, as shown later by Theorem 3.) The *extended* database points are then transformed (encrypted) using ASPE.

Similarly, we need to extend a query q to a (d+1)-dimensional point $\hat{q}$ before applying ASPE. The simplest way is to set the (d+1)-st dimension of $\hat{q}$ to 1. The weakness of this simple method is that the unencrypted query points $\hat{q}$'s all lie on a d-dimensional hyperplane with the unit vector in the (d+1)-st dimension being the normal of the hyperplane. Since APSE is a linear transformation, the encrypted query points all lie on a d-dimensional hyperplane in the transformed space as well. The attacker can determine the normal of that hyperplane in the transformed space. By considering the normal in the original space and the normal in the transformed space, the attacker obtains some level-3-like information, which is undesirable.

To avoid this problem, we introduce a random factor. For each query q, we generate a random number $r > 0$ and scale $\hat{q}$ by r, i.e., $\hat{q} = r(q^T, 1)^T$. We will show in Theorem 3 that this scaling does not affect the correctness of the distance comparison operation.

We summarize in Scheme 1 the procedures of the encryption scheme using ASPE.[2]

- **Key**: a $(d+1) \times (d+1)$ invertible matrix M.
- **Tuple encryption function** E_T: Consider a database point p. (1) Create a (d+1)-dimensional point $\hat{p} = (p^T, -0.5||p||^2)^T$. (2) The encrypted point $p' = M^T \hat{p}$.
- **Query encryption function** E_Q: Consider a query point q. (1) Generate a random number $r > 0$. Create a (d+1)-dimensional point $\hat{q} = r(q^T, 1)^T$. (3) The encrypted query point $q' = M^{-1} \hat{q}$.
- **Distance comparison operator** A_e: Let p_1' and p_2' be the encrypted points of p_1 and p_2 respectively. To determine whether p_1 is nearer to a query point q than p_2 is, the system checks whether $(p_1' - p_2') \cdot q' > 0$, where q' is the encrypted point of q.
- **Decryption function** D: Consider an encrypted point p'. The original point $p = \pi_d M^{T^{-1}} p'$ where π_d is a $d \times (d+1)$ matrix which projects on the first d dimensions and $\pi_d = (I_d, 0)$ where I_d is the $d \times d$ identity matrix.

Scheme 1. ASPE

Theorem 3. *Suppose p_1', p_2' and q' are the encrypted points of the database points p_1, p_2 and the query point q, respectively, Scheme 1 correctly determines whether p_1 is closer to q than p_2 is by evaluating $(p_1' - p_2') \cdot q' > 0$.*

Proof. Note that

$$\begin{aligned} (p_1' - p_2') \cdot q' &= (p_1' - p_2')^T q' \\ &= (M^T \hat{p}_1 - M^T \hat{p}_2)^T M^{-1} \hat{q} \\ &= (\hat{p}_1 - \hat{p}_2)^T \hat{q}. \end{aligned}$$

The scalar product of these two (d+1)-dimensional points can be represented as

$$\begin{aligned} & (p_1 - p_2)^T (rq) + (-0.5||p_1||^2 + 0.5||p_2||^2) r \\ = \; & 0.5r(||p_2||^2 - |p_1||^2 + 2(p_1 - p_2)^T q) \\ = \; & 0.5r(d(p_2, q) - d(p_1, q)) \end{aligned}$$

So, the condition is equivalent to
$0.5r(d(p_2, q) - d(p_1, q)) > 0 \Leftrightarrow d(p_2, q) > d(p_1, q)$. □

3.2 Cost and security analysis

In this section, we analyze the cost of Scheme 1 and study whether the scheme can resist level-2 and level-3 attacks. First, the cost:

- Encryption and decryption: To encrypt and decrypt, we perform two kinds of operations: (1) multiplication of an $O(d) \times O(d)$ matrix and an $O(d)$-dimensional point, which takes $O(d^2)$ time, and (2) computation of the Euclidean norm of an $O(d)$-dimensional point, which takes $O(d)$ time. Computing $E_T()$ requires both

[2] There is a special case that if an encrypted point is the origin of the transformed space, the corresponding unencrypted point is the origin of the original space. In order to avoid this special inference, we can perform a translation before applying Scheme 1. In that case, the origin is translated to a random point O'. This translation does not affect the correctness of the scheme.

operations and hence it takes $O(d^2)$ time. Computing $E_Q()$ or $D()$ requires (1) only and that takes $O(d^2)$ time.

- kNN computation using linear scan: Given an encrypted query q', the EDBMS is required to compute only one scalar product for each encrypted database point p'_i. This takes $O(nd)$ time where n is the size of database. Then, every distance comparison can be made by two scalar products: $p'_1 \cdot q'$ and $p'_2 \cdot q'$. This takes $O(1)$. In a linear scan, we can use a heap to store the kNN results. So, it takes at most $O(\lg k)$ distance comparisons for a point. In summary, a kNN query can be computed in $O(nd\lg(k))$ time. Note that the time complexity is the same as linear scan on unencrypted data except that we have $(d+1)$-dimensional points instead of d-dimensional points. (We will discuss how more sophisticated kNN algorithms can be derived in Section 6.)

Regrading attacks, we first show that Scheme 1 is not secure against level-3 attacks.

Theorem 4. *Assume that Scheme 1 is attacked by a level-3 attacker whose knowledge $H = \langle E(DB),\ P,\ I\rangle$. If there are $d+1$ points x_i ($1 \le i \le d+1$) in P such that the vectors $(x_i, -0.5||x_i||^2)$ are linearly independent, then the attacker can recover DB from $E(DB)$.*

Proof. To recover DB from $E(DB)$, one needs to recover the encryption key, that is the $(d+1) \times (d+1)$ matrix M. Since we know $P = \{x_1, x_2, ..., x_{d+1}\}$ and the corresponding encrypted values $I(x_i)$, we can set up the following equations to solve M: $M\hat{x_i} = I(x_i)$ where $\hat{x_i} = (x_i, -0.5||x_i||^2)^T$ for $i = 1$ to $d+1$. Let A, B be $(d+1) \times (d+1)$ matrices such that $A = (\hat{x_1}, \hat{x_2}, ..., \hat{x}_{d+1})$ and $B = (I(x_1), I(x_2), ..., I(x_{d+1}))$. We have $MA = B$. Note that A is invertible since the $\hat{x_i}$'s are linearly independent. We can compute $M = BA^{-1}$. Hence, we are able to recover any encrypted point in $E(DB)$. □

We remark that level-3 attacks are rare, since it is generally difficult for the attacker to obtain the necessary knowledge. Although Scheme 1 is not secure against level-3 attacks, we can show that it is resilient to common level-2 attacks. Under level-2 attacks, $H = \langle E(DB), P\rangle$, any point in $E(DB)$ can be the encrypted point of a given point in P. The attacker must acquire more knowledge in order to attack. For example, he can link the known points in P to a set of encrypted points and use the signature linking attack (see Section 2). We will show in Theorem 5 that Scheme 1 does not reveal the distances between database points and thus it prevents the distance-based signature linking attack.

Theorem 5. *E_T in Scheme 1 is not distance-recoverable.*

Proof. A detailed proof is presented in Appendix C. Here we give an idea of the proof. If an encryption E is distance-recoverable (i.e., E is a DRE), then by our definition, there exists a computational procedure f such that for all points p_1 and p_2 and any encryption key K_1 such that $a_1 = E(p_1, K_1)$ and $a_2 = E(p_2, K_1)$, we have $f(a_1, a_2) = d(p_1, p_2)$. That is, given the encrypted values a_1 and a_2, the distance $d(p_1, p_2)$ can be computed by f regardless of the encryption key. In our proof, we show that it is possible to construct two points x and y and an encryption key K_2 such that

1. $E(x, K_2) = a_1 = E(p_1, K_1)$, and
2. $E(y, K_2) = a_2 = E(p_2, K_1)$, but
3. $d(x, y) \neq d(p_1, p_2)$.

Now, if E is a DRE, we have

$$\begin{aligned} f(a_1, a_2) &= f(E(p_1, K_1), E(p_2, K_1)) = d(p_1, p_2) \text{ and} \\ f(a_1, a_2) &= f(E(x, K_2), E(y, K_2)) = d(x, y) \end{aligned}$$

A contradiction since $d(p1, p_2) \neq d(x, y)$. □

There are other attacking techniques that are applicable at level-2. Here we consider three examples, namely, PCA, Duplicate Analysis, and Brute-force attack.

Principle component analysis (PCA) has been proposed in [15] to match the correlations in the set of known data points and the correlations in the encrypted data. The matching result enriches the knowledge of the attacker. In our scheme, the values on each dimension of $E(DB)$ is a linear combination of the values on all dimensions in the plain database. So, ASPE does not preserve the correlations among the original dimensions in the transformed space. Hence, PCA is not applicable. Duplicate analysis [2] exploits the observation that the domains of some attributes may be small, e.g., day of the month. Observations on the duplicates in the encrypted database may help an attacker to find the domain of an attribute. Duplicate analysis targets at value-based encryption, i.e., each value in each dimension is encrypted individually. Since Scheme 1 is a tuple-based encryption, duplicate analysis on individual attribute is not applicable.

As another option, the attacker may use a *brute-force exhaustive attack*. We have shown in Theorem 4 that $d+1$ points are enough to solve the keys in a level-3 attack. Given a level-2 attack, without the knowledge of I, the attacker has to try every possible I to recover the database. He can divide P into two sets: a training set P_t and a testing set P_v where $|P_t| = d+1$ and $P_v = P - P_t$. He randomly picks a set of $d+1$ encrypted points Q and sets up a hypothesis that Q contains the corresponding encrypted points of the points in P_t. Then, the attacker can set up equations to solve for M and use P_v to verify the hypothesis: if the recovered database contains P_v, the hypothesis may be correct; otherwise, the hypothesis cannot be true. The confidence in the verification of hypothesis increases with the number of points in P. However, this brute-force exhaustive attack is exponentially expensive for the attacker. There are ${}_nP_{d+1} = O(n^{d+1})$ possible candidates of Q. For each candidate, the attacker performs a 'decryption' of database using the recovered key which takes $O(n)$ decryptions. For example, if $n = 10K$, $d = 2$ and an attacker is capable to perform $1M$ decryptions in a second, it takes more than 310 years to try all hypotheses. Our scheme is therefore very resilient against this level-2 brute-force attack.

4. DEALING WITH STRONGER ATTACKS

In this section we discuss how the ASPE-based encryption scheme can be enhanced to resist level-3 attacks.

4.1 Random Asymmetric Splitting

A weakness of Scheme 1 is that given an enough number of points in P, a level-3 attacker can set up enough number of equations to solve for the unknowns in the transformation matrix M. To make the scheme harder to crack, one method is to introduce randomness into the scheme to make it very difficult to set up the equations. To achieve that, given a database point p, we split the value of p at each dimension randomly. More specifically, we randomly generate two d-dimensional points p_a and p_b (called two *random shares*) such that for each dimension i, we have $p[i] = p_a[i] + p_b[i]$. For simplicity, we write $p = p_a + p_b$. Note that for any query q, we have $p \cdot q = p_a \cdot q + p_b \cdot q$, and thus $p \cdot q$ can be computed by two scalar products. As a 2-D example, if $p = (3, 7)$, we can create $p_a = (10, 2)$ and $p_b = (-7, 5)$. We call this method p-splitting.

We can use two ASPEs with *different* keys to encrypt and to compute the two scalar products. In other words, we generate two transformation matrices, M_1 and M_2, and encrypt the two random shares of p as $p'_a = M_1^T p_a$, and $p'_b = M_2^T p_b$. If q is also encrypted by the same ASPEs to obtain two encrypted query points: $q'_a = M_1^{-1} q$ and $q'_b = M_2^{-1} q$, we can compute $sp_a = p'_a \cdot q'_a$ and $sp_b = p'_b \cdot q'_b$ separately and recover $p \cdot q$ by $sp_a + sp_b$. Hence, this splitting preserves type-2 scalar products and allows the distance comparison operation to be correctly evaluated. Note that in the above description, unlike p, q is *not* split.

This splitting technique alone, however, does not improve security. Consider a query point q and let q'_a and q'_b be the encrypted points of q using the two ASPEs M_1 and M_2. We know that $q'_a = M_1^{-1} M_2 q'_b$. Note that $M_1^{-1} M_2$ is an unknown $d \times d$ matrix. If an attacker observes d (encrypted) queries, he can construct a square matrix Q'_a whose columns are formed by the observed q'_a vectors. Likewise, the observed q'_b vectors can be used to obtain a square matrix Q'_b. We have $Q'_a = M_1^{-1} M_2 Q'_b$ and thus $M_1^{-1} M_2 = Q'_a Q_b'^{-1}$.

Now, a level-3 attacker can set up the necessary equations to solve for M_1 and M_2 in the following way. Since $p'_a = M_1^T p_a$, we have $(Q'_a Q_b'^{-1})^T p'_a$

$$
\begin{aligned}
&= (Q'_a Q_b'^{-1})^T M_1^T p_a \\
&= (M_1^{-1} M_2)^T M_1^T p_a \\
&= M_2^T (M_1^T)^{-1} M_1^T p_a = M_2^T p_a
\end{aligned}
$$

Hence, $(Q'_a Q_b'^{-1})^T p'_a + p'_b = M_2^T p_a + M_2^T p_b = M_2^T p$. For a level-3 attacker, if p is a database point in his knowledge base P, the attacker knows the encrypted values p'_a and p'_b. So, $t = (Q'_a Q_b'^{-1})^T p'_a + p'_b$ can be computed by the attacker. He can establish the equation: $M_2^T p = t$ and solve for M_2 as shown in the discussion of Theorem 4. The attacker can also solve for M_1 similarly, and thus crack the encryption scheme.

Instead of splitting p, we can also consider splitting q. Similar to the previous procedure, we can split q into two random shares q_a and q_b so that $q_a + q_b = q$. The two vectors q_a and q_b are encrypted using two ASPEs with different keys to get $q'_a = M_1^{-1} q_a$ and $q'_b = M_2^{-1} q_b$. Also, p (not split) is encrypted by the two ASPEs and we get $p'_a = M_1^T p$ and $p'_b = M_2^T p$. We call this q-splitting. Now, player 1 and player 2 can secretly agree on what to split in the encryption scheme; they can either agree to applying splitting only on database points, or only on query points. In order for the attacker to correctly set up equations to solve for the transformation matrices, the attacker has to know which *configuration* (i.e., p-splitting or q-splitting) players 1 and 2 have agreed on. Or, he has to attack each configuration separately. That doubles his cost.

We can further extend this splitting technique by applying splitting on each dimension independently. Again, consider a 2D example where $p = (x_1, x_2)$ and $q = (y_1, y_2)$. We may create $p_a = (x_1, x_{2a})$, $p_b = (x_1, x_{2b})$, $q_a = (y_{1a}, y_2)$, and $q_b = (y_{1b}, y_2)$ such that $x_{2a} + x_{2b} = x_2$ and $y_{1a} + y_{1b} = y_1$. In this example, we split p on the 2nd dimension and q on the 1st dimension. The scalar product of $p \cdot q$ is preserved: $p \cdot q = p_a \cdot q_a + p_b \cdot q_b$.

In general, for each dimension i, players 1 and 2 secretly agree on which of $p[i]$ and $q[i]$ to split. They thus share a *configuration bit vector*, which is a d-bit vector such that each entry in the bit vector indicates whether p-splitting or q-splitting is used for the corresponding dimension. Basically, the attacker has to try all configurations in order to solve for the transformation matrices. Since there are 2^d possible configurations, the enhanced scheme is 2^d more costly to attack compared with Scheme 1.

4.2 Adding Artificial Dimensions

We can improve the security of our encryption scheme by making d larger. This can be achieved by adding *artificial* dimensions to the data points. We extend a d-dimensional data point p (q) to a d' dimensional ($d \leq d'$) point $\dot{p}$ ($\dot{q}$) by padding artificial attributes such that the scalar product over the added attribute values is 0.

More specifically, The values of dimensions d+1 to d' of $\dot{p}$ and $\dot{q}$ are generated randomly. We partition the range $[d+1, d']$ into two groups: G_p, G_q. G_p (resp. G_q) represents the indexes of the dimensions on which we perform p-splitting (resp. q-splitting). Without loss of generality, let us assume $G_p = [d+1, s]$ and $G_q = [s+1, d']$ for some integer s. We will discuss how to generate the values for dimensions d+1 to s. (The values of the other dimensions can be similarly generated except that we swap the roles of $\dot{p}$ and $\dot{q}$.) First, we create a set of $s - d$ pre-generated random values: w_{d+1}, w_{d+2}, ..., w_s, where $w_s \neq 0$ if $s - d \geq 2$. These random numbers become parts of the encryption key. We set $\dot{p}[i] = w_i$ for each i in [d+1,s]. Although we use the same set of values (w_i's) for every $\dot{p}$, each $\dot{p}$ is split into two shares $\dot{p}_a$ and $\dot{p}_b$ randomly. Hence, the generated $\dot{p}_a$'s and $\dot{p}_b$'s appear as random points to the attacker.

For each $\dot{q}$, the values of dimensions d+1 to s-1 are all generated randomly. (Unlike p, different q could use a different set of random numbers.) Let t_c be the number assigned to $\dot{q}[c]$ ($d+1 \leq c \leq s-1$). The s-th dimension of $\dot{q}$ is set to $t_s = \frac{-\sum_{i=d+1}^{s-1} w_i t_i}{w_s}$. (If $s = d+2$, we set t_s to 0 if $w_s \neq 0$ or any value if $w_s = 0$.) It is easy to see that the scalar product of $\dot{p}$ and $\dot{q}$ over the attributes d+1 to s is 0.

We repeat the above value assignment procedure to attributes s+1 to d' except that the roles of $\dot{p}$ and $\dot{q}$ are switched. Since the scalar product over the artificial attributes is by design equal to 0, we have $\dot{p} \cdot \dot{q} = p \cdot q$.

4.3 Analysis on the Refined Scheme

The introduction of the splitting technique and artificial attributes significantly increases the cost of the attacker. We can integrate the two new techniques into Scheme 1 by first applying them before ASPE is applied. The enhanced encryption scheme is shown in Scheme 2.

As we have mentioned before, the number of possible ways of splitting is $2^{d'}$, which is exponentially large. In order to make the system secure, d' must be sufficiently large. RSA keys are required to be at least 1024-bit. The general

- **Key**: two $d' \times d'$ invertible matrices M_1, M_2; A bit string S of d' bits: S_i denotes the i-th bit in S; $d' - (d+1)$ random numbers: w_{d+2}, w_{d+3}, ..., $w_{d'}$.

- **Tuple encryption function** E_T: Consider a database point p. (1) Create a d'-dimensional point $\hat{p}$ where the first d dimensions are copied from p. $\hat{p}[d+1]$ is set to $-0.5||p||^2$. For $i = d+2$ to d', if $S_i = 1$, set $\hat{p}[i]$ to w_i; otherwise, set $\hat{p}[i]$ to a random number. For the last dimension with which $S_i = 0$, $\hat{p}[i]$ is given a value so that the scalar product over the artificial attributes is 0 (see Section 4.2). (2) Create two shares of $\hat{p}$: $\hat{p}_a$ and $\hat{p}_b$. For $i = 1$ to d', if $S_i = 1$, we randomly split the value of $\hat{p}[i]$ into $\hat{p}_a[i]$ and $\hat{p}_b[i]$. If $S_i = 0$, $\hat{p}_a[i]$ and $\hat{p}_b[i]$ are both set to $\hat{p}[i]$. (3) The encrypted value of p is the pair ($p'_a = M_1^T \hat{p}_a$, $p'_b = M_2^T \hat{p}_b$).

- **Query encryption function** E_Q: Consider a query point q. (1) Generate a random number $r > 0$. Create a d'-dimensional point $\hat{q}$ where the first d dimensions are given by rq. The $(d+1)$-st dimension is set to r. For $i = d+2$ to d', if $S_i = 0$, $\hat{q}[i]$ is set to w_i; Otherwise, $\hat{q}[i]$ is set to a random number. For the last dimension with which $S_i = 1$, $\hat{q}[i]$ is given a value so that the scalar product over the artificial attributes is 0. (2) Create two shares of $\hat{q}$: $\hat{q}_a$ and $\hat{q}_b$. For $i = 1$ to d', if $S_i = 0$, we randomly split the value of $\hat{q}[i]$ into $\hat{q}_a[i]$ and $\hat{q}_b[i]$. If $S_i = 1$, $\hat{q}_a[i]$ and $\hat{q}_b[i]$ are both set to $\hat{q}[i]$. (3) The encrypted value of q is the pair ($q'_a = M_1^{-1}\hat{q}_a$, $q'_b = M_2^{-1}\hat{q}_b$).

- **Distance comparison operator** A_e: Let (p'_{1a}, p'_{1b}) and (p'_{2a}, p'_{2b}) be the encrypted values of p_1 and p_2 respectively. To determine whether p_1 is nearer to a query point q than p_2, the system checks whether $(p'_{1a} - p'_{2a}) \cdot q'_a + (p'_{1b} - p'_{2b}) \cdot q'_b > 0$ where (q'_a, q'_b) is the encrypted value of q.

- **Decryption function** D: Consider an encrypted value (p'_a, p'_b). First, recover the two shares and extract the first d dimensions in it: $p_a = \pi_d M_1^{T^{-1}} p'_a$ and $p_b = \pi_d M_2^{T^{-1}} p'_b$ where π_d is a $d \times (d+1)$ matrix which is the projection on the first d dimensions. Then, $p[i]$ is equal to $p_a[i]$ if $S_i = 0$; or $p_a[i] + p_b[i]$ if $S_i = 1$.

Scheme 2. Enhanced ASPE

consensus is that 1024-bit RSA keys are roughly equivalent in strength to that of 80-bit symmetric keys. We can set $d' \geq 80$ so that the search space is sufficiently large. In addition, a large d' implies larger transformation matrices, making the encryption scheme even harder to crack. Therefore, it is generally very difficult for the attacker to successfully attack Scheme 2.

THEOREM 6. *Scheme 2 is resilient to a level-3 attack if the attacker cannot derive the splitting configuration, i.e., the bit string S of the key in Scheme 2.*

PROOF. Let the knowledge of the attacker be $H = \langle E(DB), P, I\rangle$. Without loss of generality, let us assume $d' = d + 1$, i.e., no artificial attributes are added. (Note that the addition of artificial attributes will only increase the security of the scheme.) For any point $p_i \in P$, by definition, a level-3 attacker knows the encrypted values (p'_{ia}, p'_{ib}). If the attacker does not know the splitting configuration, he has to model $\hat{p}_{ia}$ and $\hat{p}_{ib}$ as two random d'-dimensional vectors. The equations for solving the transformation matrices are: $M_1^T \hat{p}_a = p'_a$ and $M_2^T \hat{p}_b = p'_b$, where M_1 and M_2 are two $d' \times d'$ unknown matrices. There are $2d'|P|$ unknowns in $\hat{p}_{ia}$, and $\hat{p}_{ib}$ and $2d'^2$ unknowns in M_1 and M_2. Since there are only $2d'|P|$ equations, which are less than the number of unknowns, the attacker does not have sufficient information to solve for the transformation matrices. Hence, Scheme 2 is resilient against this level-3 attack. □

5. EMPIRICAL EVALUATION

In this section we present two sets of experiments we have conducted to evaluate: (i) the time taken by a level-2 attacker to break a DRE scheme; (ii) the overhead incurred in the three encryption schemes (DPT, Scheme 1 and Scheme 2) in secure kNN computation. All programs are implemented in C++. Experiments are performed on an Intel Core 2 Duo 2.66GHz computer with 2 GB RAM running Windows XP.

5.1 Feasibility of signature linking attack on DRE

We have shown that it is theoretically feasible for a level-2 attacker to break a DRE scheme using the signature linking attack. In this experiment, we study the effectiveness of the attack in two aspects: (i) the smallest size (denoted by σ) of P that is required for the attack to be successful, and (ii) the execution time of a successful attack.

First, we outline an algorithm to implement the signature linking attack. Given a set $P = \{x_1, x_2, ..., x_{|P|}\} \subset DB$ in a level-2 attacker's knowledge H, we want to find a unique ordered set $Q \subset E(DB)$ such that $sig(Q) = sig(P)$. (If more than one such Q with the matching signature are found, a signature collision occurs and the attacker cannot break the DRE scheme.) To find such a Q, we perform incremental binding in a depth-first manner. We start by picking up a $y_1 \in E(DB)$ and bind it to x_1. We denote this binding by $Q = \{y_1, *\}$. (The asterisk denotes that x's on the right are unbound yet.) We extend this binding by picking another $y_2 \in E(DB)$ and bind it to x_2, giving the binding $Q = \{y_1, y_2, *\}$. This binding extension induces the constraint: $d(y_1, y_2) = d(x_1, x_2)$. We check this constraint and if it is satisfied, we continue with the binding extension; otherwise, we backtrack and pick another y'_2 to be bound to x_2. In general, extending a binding $Q = \{y_1, \ldots, y_i, *\}$ to the binding $Q = \{y_1, \ldots, y_i, y_{i+1}, *\}$ induces i constraints: $d(y_j, y_{i+1}) = d(x_j, x_{i+1})$ (for $1 \leq j \leq i$). If any constraint is not satisfied, we rollback the extension and try another y'_{i+1}. When a complete binding $Q = \{y_1, y_2, \ldots, y_{|P|}\}$ is found, Q gives a solution. Then, we continue the process by rolling backing the binding in search of another solution Q.

To determine the size of P that is required to break a DRE scheme, we randomly generate a P of size 3. We execute the above binding procedure until one of the following cases happen: (1) if 2 solutions (i.e., 2 Q's) are found, we stop the binding process, extend the size of P by adding one more point to it and repeat the binding process; (2) if only 1 solution (a unique Q) is found, we stop. The size of P is taken as σ. Also, the time taken to find the solution Q given the final P set is taken as the time to attack the DRE scheme. In our experiment, the above exercise is repeated 10 times. We report the average value of the σ's obtained in the 10 trials.

We perform experiments on both synthetic and real datasets.

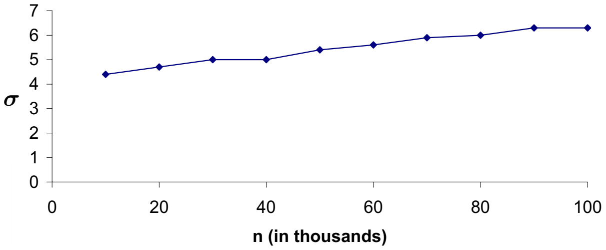

Figure 2: Minimum size of P required for a successful attack (σ) vs. n.

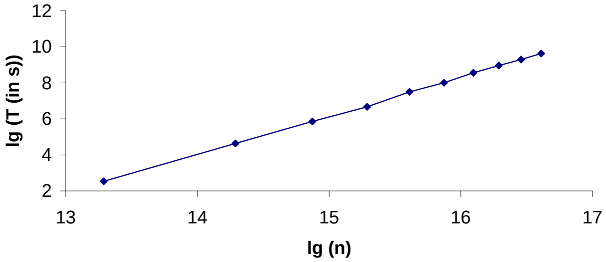

Figure 3: Average execution time T of a successful attack vs. n.

A synthetic database consists of n uniformly distributed random points in a d-dimensional space (for various values of n and d). We also use the dataset 'Shuttle' from the UCI repository as the real dataset [4]. The Shuttle dataset contains 58K points and 9 dimensions.

Figures 2 and 3 show the average value of σ and the average attack time in log scale under various values of n for the synthetic databases.

Figure 3 shows a straight line with a slope of 2.14. This suggests that the execution time required by signature linking attack is of the order of $n^{2.14}$, which is polynomial time. The cost is thus affordable. For example, with n = 100K, an attack takes about 800 seconds. Moreover, σ varies from 4.4 when n = 10K to 6.3 when n = 100K. The knowledge required to attack is generally very small. For the Shuttle dataset, $\sigma = 4.6$ and the average attack time is 314 seconds. These numbers are similar to those obtained from the synthetic dataset. The results show that DRE is easily breakable. This strengthens the motivation in our exploration of non-distance-recoverable encryption schemes.

5.2 Performance of encryption schemes

In this section, we evaluate the overhead incurred by the encryption schemes. We compare their cost to that of executing kNN on plain data. We implemented the three encryption schemes: DPT, Scheme 1 and Scheme 2. We evaluate the performance of the schemes under 4 tasks: (i) key generation; (ii) database encryption; (iii) kNN computation and (iv) query encryption and result decryption. Again, synthetic datasets and the Shuttle dataset are used in the evaluation.

With synthetic datasets, we conduct two experiments. In the first experiment, we vary n from 10K to 100K with a fixed $d = 4$. In the second experiment, we vary d from 2 to 100 with a fixed n = 50K. For Scheme 2, we need to pick the number of dimensions d' of the transformed space $E(DB)$. To make the scheme secure, we set $d' = 80$. For those synthetic datasets with $d > 80$, we pick $d' = d + 1$. (We will further study the effect of d' later.)

n	20K	40K	60K	80K	100K
DPT	0.0045	0.0092	0.0139	0.0186	0.0231
Scheme 1	0.0064	0.0131	0.0208	0.0266	0.0331
Scheme 2	1.4150	2.8298	4.2477	5.7106	7.5997

Table 1: Average database encryption time (in s) vs. n under various schemes ($d = 4$).

	DPT	Scheme 1	Scheme 2
Execution time (in s)	0.0341	0.0456	4.113

Table 2: Encryption time (in s) of the Shuttle dataset under various schemes.

5.2.1 *Key generation*

For Scheme 1, the key is an $O(d) \times O(d)$ matrix. For Scheme 2, the key consists of an $O(d') \times O(d')$ matrix, an $O(d')$ vector and $O(d')$ random numbers. In our experiments, it take less than 1ms to generate the keys for d ranging from 2 to 100 under either scheme.

5.2.2 *Database Encryption*

For each value of n and d, 100 synthetic databases are generated and the average amount of time taken by each scheme to encrypt the database is recorded. Table 1 shows the encryption time for various n with $d = 4$, while Figure 4 shows the encryption time for various d with n = 50K. We also show the encryption time of the Shuttle dataset in Table 2.

Tables 1 and 2 show that the encryption cost scales well against n under all three schemes. In particular, DPT and Scheme 1 encrypt the databases in sub-seconds. Scheme 2 requires more time because the transformation matrices are much bigger ($d' = 80$) than those of Scheme 1 ($d = 4$). Since we pick $d' = 80$ unless $d > 80$, the database encryption time for Scheme 2 stays flat when $d \leq 80$ (see Figure 4). Considering that database encryption generally has a less stringent response time requirement compared with query processing, the database encryption costs are relatively small.

We note that Scheme 2's database encryption cost is affected by the choice of d'. We thus repeat the experiment on Scheme 2 varying d' in the range [56,100]. In this exper-

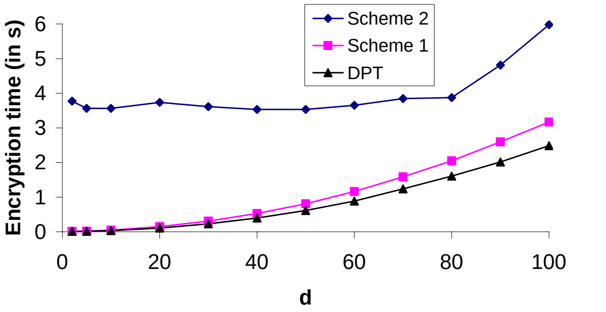

Figure 4: Average database encryption time (in s) vs. d under various schemes (n = 50K).

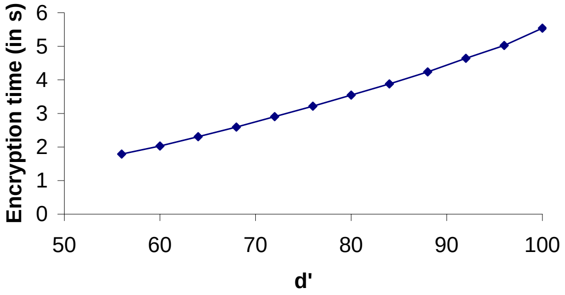

Figure 5: Average database encryption time (in s) vs. d' under Scheme 2 (n = 50K; d = 4).

	Unencrypted / DPT	Scheme 1	Scheme 2
Execution time (in ms)	11.25	10.52	74.48

Table 3: Average query execution time (Shuttle dataset; k = 10).

iment, n = 50K and d = 4. Pervious research has suggested that a search space of 2^{56} (such as 56-bit DES) is generally considered not secure against brute-force search with today's computing power. Also, a commonly accepted standard is 80 bits as a low-end requirement of encryption key length. So, the range [56,100] for d' is a reasonable one for our study. Figure 5 shows the results. From the figure, we see that encryption time is reasonably small within the range studied. For example, when d = 100, it takes about 6.0s to encrypt a 50K database.

5.2.3 kNN computation

In this experiment, we measure the execution time of kNN queries. We implement kNN search using a simple linear scan algorithm which uses a heap structure to maintain the k closest points to the query that the algorithm has seen so far during the execution. (We will discuss how faster kNN algorithms can be implemented using our encryption schemes in Section 6.) For each encryption scheme, we measure its average execution time over 1,000 random kNN queries. We compare their performance against the case in which kNN is executed on plain unencrypted data (called 'unencrypted scheme' in the performance figures). Note that DPT preserves the distances between data points. Hence, the query execution time under DPT is the same as that under the unencrypted scheme. Figures 6 and 7 show the results for synthetic datasets. In the former, we vary n with d = 4 while in the latter, we vary d with n = 50K. In both cases k is 10 and d' = 80 if $d \leq 80$. Table 3 shows the results for the Shuttle dataset.

Figure 6 shows that the cost of kNN computation scales linearly to the number of points in the database under all three encryption schemes. The reason why Scheme 2 is slower than the others is that the default number of dimension of the transformed space (d') is 80 for Scheme 2, while it is only 4 for the other schemes. For this reason, we see from Figure 7 that the query execution time of Scheme 2 stays flat for $2 \leq d' \leq 80$. Also, we see that Scheme 1 is actually faster than the unencrypted scheme in processing queries. It is because we have pre-computed $||p||$ for each point p in the database under Scheme 1. For each query q, Scheme 1 performs d+1 multiplications and d additions to compute $p' \cdot q'$ for each database point p. Under the unencrypted scheme, we compute $d(p, q)^2$ for each database point p and a query point q. This takes d multiplications, d subtractions and d - 1 additions. So, when d is large, Scheme 1 is noticeably faster.

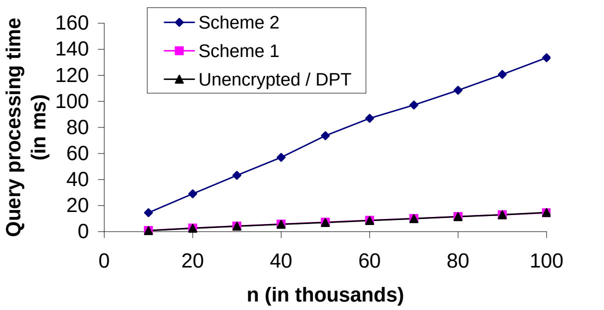

Figure 6: Average query execution time vs. n (Synthetic data; k = 10; d = 4).

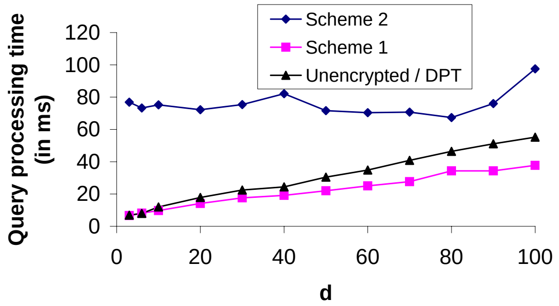

Figure 7: Average query execution time vs. d (Synthetic data; k = 10; n = 50K).

Next, we vary d' in the range [56,100] and measure query execution time under Scheme 2. In this experiment, we set n = 50K, d = 4 and k = 10. Figure 8 shows the result. We see that the execution time scales linearly with d'. We can thus use a reasonably large d' to provide good security protection without significantly degrading query processing time.

5.2.4 Query encryption and result decryption

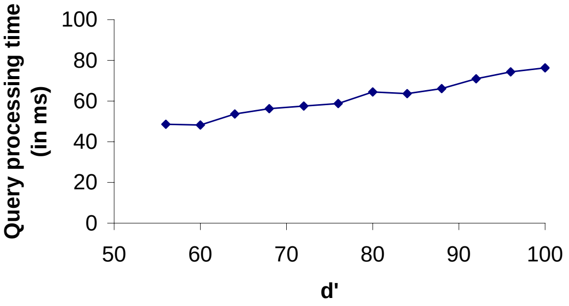

Figure 8: Average query execution time of Scheme 2 vs. d' (Synthetic data; n = 50K; d = 4; k = 10)

For each query, player 2 needs to perform one encryption and k decryptions. In our experiments, it takes at most 1ms. The cost is negligible.

5.3 Summary

We have shown that signature linking attack only requires a small number of known points in P to break a DRE and the attack cost is not expensive. So, a level-2 attacker can easily compromise the security of a DRE scheme. This justifies the need of non-distance-recoverable encryption schemes for secure kNN computation. We evaluated the performance of the two proposed schemes. Scheme 1 has a very low cost and it resists level-2 attacks. We remark that targeting level-2 attacks in security protection is practical and Scheme 1 is suitable for this requirement. Relatively speaking, Scheme 2 has higher overheads due to the introduction of additional dimensions. We remark that the additional dimensions are a necessary evil if the primary goal of the scheme is to provide the best protection against attackers. Thus, there is a tradeoff between Scheme 2, which is resilient to level-3 attacks, and Scheme 1, which allows more efficient query processing.

6. FAST KNN SEARCH

So far, we have discussed kNN computation on encrypted data using a simple linear scan algorithm. Although we have shown that query processing is generally very fast in our experimental results, it is interesting to see how more advanced kNN algorithms can be employed under the SCONEDB model. Such techniques would be very useful particularly in applications where there is a massive amount of data and/or queries. In this section, we briefly discuss how the kNN problem in the original data space can be easily transformed into a top-k problem in the encrypted transformed space. This observation allows the service provider to apply any existing efficient top-k algorithm on the encrypted data to answer kNN queries very efficiently.

In our encryption schemes, the EDBMS uses the distance comparison operator A_e to find the nearest point to a query point. Specifically, given two encrypted points $p'_1 = E_T(p_1, K)$, $p'_2 = E_T(p_2, K)$ and an encrypted query $q' = E_Q(q, K)$, A_e compares $p'_1 \cdot q'$ and $p'_2 \cdot q'$ (see Scheme 1). If $p'_1 \cdot q' > p'_2 \cdot q'$ then p_1 is closer to q than p_2 is. We can generalize this and observe that the k encrypted points p_i's that give the largest type-2 scalar products $p'_i \cdot q'$ are the k nearest neighbors of q. Hence, given q, the kNN query can be treated as a top-k query with the linear ranking function F: $F(p') = p' \cdot q'$ for all $p' \in E(DB)$.

The problem of top-k queries has been studied extensively in the literature [23, 22, 24]. For a linear ranking function, an efficient solution is to build a ranking cube that stores the top-k results [23]. While data cube suffers from the curse of dimensionality, a novel technique of 'ranking fragments' is introduced to handle high-dimensional rank queries [23]. We remark that the choice of a "top-k" algorithm is orthogonal to our study (as long as the algorithm can handle linear ranking functions[3]). The service provider is thus free to choose an efficient implementation for fast kNN search. This decoupling of the search algorithm and our distance comparison operator allows much flexibility in the design of the SCONEDB model.

7. RELATED WORK

The outsourced database (ODB) model was introduced in [12]. A data owner (DO) outsources its database hosting and query answering to a service provider (SP), who is not trusted. Security in ODB is studied, similarly to our SCONEDB model. The goal is to protect the data using encryption and allow query processing on the encrypted database. For example, an order-preserving encryption scheme (OPES) [2] applies an encryption function E to an ordinal domain, such that $E(x) < E(y)$ for every pair of values for which $x < y$. OPES facilitates evaluation of range queries. In addition, an additive and multiplicative homomorphic encryption function E (i.e. $E(x)+E(y) = E(x+y)$ and $E(x)E(y) = E(xy)$) is proposed in [11] to support aggregate queries on encrypted data. However, as shown in [19], the homomorphic scheme is not secure even at the lowest security level (level-1 in our model). In summary, in previous ODB models only simple numerical domains and SQL operations on them have been considered, while we target more sophisticated operations (i.e., kNN search). In addition, every study in ODB assumes one single type of attack, while SCONEDB considers different levels of attacks and it is being adapted to each of them.

Apart from encryption techniques with special properties, there are other data protection methods that facilitate secure computation of queries. "Coarse" (a.k.a. bucket-based) indexes are introduced to facilitate execution of SQL statements in the ODB model [10]. The tuples are encrypted by common encryption schemes like RSA. The domain of each database attribute is divided into partitions. Each partition is assigned an ID by a hash function. DO sends to SP the encrypted tuples together with their partition ID to serve as the "coarse" index. The query is transformed to retrieve the partitions that contain the targeted tuples. SP returns a superset of the query result. The users, who own the key, decrypt the result and perform a post-processing to filter false positives. For advanced queries, the number of false positives can be huge, causing a heavy load at the users. For example, distances between database points to the query point are required in k-NN computation and cannot be easily captured by the partition IDs. Therefore, the direct application of this technique may cause the SP to return the entire database and let the user to compute the query result on his own. In addition, the bucket-based approach is not suitable when users have limited processing power, e.g. they use mobile devices.

Another approach for protected query processing makes use of special hardware: the secure co-processor [18, 1]. This is a secure computation unit; no parties can observe the performed computations and the data stored in it. The use of the unit is straightforward; we install the encryption and decryption key and deploy the application logic directly at the co-processor. On the other hand, the device is much slower than usual processors. It is not suitable for sophisticated applications that demand heavy computations. In addition, the co-processor must be maintained by the users. For example, if the device is down, the user has to re-deploy the co-processor. This conflicts to the paradigm of cloud computing in which users purchase the services without maintaining the resources themselves.

Various data anonymization models, like k-anonymity, have been proposed for privacy preserving data publishing [21, 14, 16]. The basic idea is to make the quasi-identifiers (sets of

[3] Indeed, most existing top-k algorithms such as [23, 22] satisfy this requirement.

attributes that can be linked with external data to uniquely identify an individual) of each tuple in the database indistinguishable from at least $k-1$ other tuples (k is a user defined parameter). k-anonymity can be achieved by generalizing the quasi-identifiers (e.g., replacing exact attribute values by domain ranges), by suppressing tuples, or by perturbing them [3, 6]. Apart from the information loss incurred by this process (queries are no longer applied on the original data), the model itself suffers from certain drawbacks. For example, as pointed out in [16], the indistinguishable groups of anonymized quasi-identifiers should also contain a diverse set of sensitive values, while information leakage may also occur with the help of limited background knowledge by the attacker. In addition, the generalized ranges of values may give an adversary an accurate estimation on the original database or valuable statistics (e.g. the number of customers who have more than $10M$ in their accounts). Note that the goals of data privacy in publication and data security in ODB are not the same; data privacy strives to avoid the linkage of published data to specific individuals, while data security in ODB aims at the protection of the stored data against unauthorized database users.

In this paper, we focus on kNN search as a case-study operation for the SCONEDB model. Distance-preserving transformation (DPT) is proposed in [20] as an encryption method for such queries. DPT transforms a given point x to $Nx + t$ where N is an orthogonal $d \times d$ matrix and t is a d-dimensional vector. The main feature of DPT is that distance between points is preserved in the transformation, i.e., $d(x, y) = d(E(x), E(y))$ where d is the Euclidean distance, E is the encryption/transformation function. Since distance is preserved, kNN queries can be computed correctly. However, the authors in [15] show that DPT is not secure w.r.t. level-2 and level-3 attacks. For a level-3 attack, we can observe a number of points $\{x_1, x_2, ..., n_m\}$ in DB and their corresponding encrypted values $\{y_1, y_2, ..., y_m\}$. We can then set up a number of $y_i = Nx_i + t$ equations, forming a system of linear equations where there are d^2 unknowns in N and d unknowns in t. So, if $m \geq d + 1$, the system is generally solvable. For a level-2 attack, an attacker observes a number of points P in DB. Since DPT preserves the correlations between dimensions, [15] uses PCA to identify the principle components in the set of points P and in the transformed database. By matching the principle components, the attacker can obtain an accurate estimation of N and t. Note that the attacks proposed in [15] do not overlap with our proposed attacks described in Section 2. [15] discusses how to attack DPT specifically and the proposed attacks are not applicable to general DRE, which is vulnerable to attacks shown in Section 2. Though the safety level of DPT is not high, we remark that DPT is useful in situations where only level-1 attacks are concerned.

kNN computation at an untrusted platform is also considered in location-based service (LBS) systems [8, 17, 9, 13]. In LBS models, a server holds a set of tuples namely points of interest (POI). Users submits queries (range query or kNN queries) to the server and retrieves the desired POIs. The main security goal is to protect the location of the query issuers, while some models also consider the privacy of POI [8, 17]. The main model used is k-anonymity; the goal is to convert the location of the query into a spatial range, such that at least $k-1$ other locations are included in that range and the service provider cannot distinguish the user in a set of k candidates. Although such a model can be used for our problem, it has certain drawbacks. First, the anonymized database reveals approximately the original values which is not desired in our model. Second, in certain models [9], the database is assumed to be owned by the server and hence the server can observe the original database. Third, in such systems (as in bucket-based indexes), the server usually returns a superset of the query result, which then has to be post-processed by the user. This requires additional effort at the user's side which may be considerable for light-weight clients. An LBS model that performs encryption for the purpose of kNN search is proposed in [13]. The main idea is to use a Hilbert curve to 'encrypt' database points and the queries. The Hilbert value of the points are given to the server. kNN is then computed on the Hilbert transformed space to give an approximate result. Except from returning an approximate result, this method has a similar problem to DPT; the Hilbert curve can be considered as an advanced type of linear transformation which can be easily attacked as we discussed in Section 2.

8. DISCUSSION AND CONCLUSION

In this paper, we introduced the SCONEDB model, which captures the behavior of the users and the attacker on encrypted databases. Many existing techniques in database service outsourcing can be incorporated into SCONEDB, e.g., OPES for range queries; homomorphic encryptions for aggregate queries. All these schemes and our proposed schemes use different encryption strategies. However, they can be integrated on a single encrypted database. For example, in a bank database, we can encrypt account balances by OPES, encrypt branch locations by Scheme 1, and encrypt names and addresses by a strong encryption like RSA. Range and kNN queries can be performed on balances and locations independently. The possibility of integrating different schemes in the SCONEDB model to support a wide range of applications makes EDBMS a practical solution to service outsourcing. In the integrated solution, we need to ensure that the different schemes are aligned on the same security level.

Emerging computing paradigms such as database service outsourcing and cloud computing call for the study of secure computation on encrypted data. In this work, we have made the following contributions: (1) We have formulated a general secure database model SCONEDB which is defined independent of the query type. The model incorporates the attacker capability as a distinct component and uses it to measure the security level of the encryption scheme. It also defines the notion of an encrypted database which can support secure computation such as secure kNN search on encrypted data. (2) We have defined distance-recoverable encryption which supports kNN computation and show that it is not secure. (3) We have developed a new asymmetric encryption function ASPE that preserves a special type of scalar product and use it to construct two secure encryption schemes that resist attackers of different levels and yet support secure and accurate kNN query computation. (4) We have carried out a performance evaluation of the proposed schemes. We remark that the SCONEDB model consolidates the requirements and goals in a single model which can integrate the previous work on database service outsourcing.

A future research issue is the systematic study on different operators that can be supported on an encrypted database w.r.t different security levels and goals. In the SCONEDB model, the capability of the attacker is specified by its background knowledge, which is intuitive. It is possible to extend the attack model to include other aspects, e.g., the amount of available computational power. Also, the security goal in SCONEDB is to prevent the attacker from breaking into the system and acquiring unauthorized data. There could be other security goals, like for example, the protection of the location privacy of query issuers [8, 17]. How to include security goal as another component into the SCONEDB model is a subject for future work.

9. REFERENCES

[1] R. Agrawal, D. Asonov, M. Kantarcioglu, and Y. Li. Sovereign joins. In *ICDE*, 2006.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *SIGMOD*, 2002.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD*, 2000.

[4] A. Asuncion and D. Newman. UCI Machine Learning Repository, 2007.

[5] H. Delfs and H. Knebl. *Introduction to Cryptography: Principles and Applications.* Springer, 2002.

[6] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, 2002.

[7] Gartner. Assessing the Security Risks of Cloud Computing (ID Number: G00157782), 2008.

[8] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *ICDCS*, 2005.

[9] 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.

[10] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *SIGMOD*, 2002.

[11] H. Hacigumus, B. Iyer, and S. Mehrotra. Efficient execution of aggregation queries over encrypted relational databases. In *DASFAA*, 2004.

[12] H. Hacigumus, S. Mehrotra, and B. Iyer. Providing database as a service. In *ICDE*, 2002.

[13] A. Khoshgozaran and C. Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *SSTD*, 2007.

[14] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, 2007.

[15] K. Liu, C. Giannella, and H. Kargupta. An attacker's view of distance preserving maps for privacy preserving data mining. In *PKDD*, 2006.

[16] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.

[17] M. F. Mokbel, C. Y. Chow, and W. G. Aref. The new casper: Query processing for location services without compromising privacy. In *VLDB*, 2006.

[18] E. Mykletun and G. Tsudik. Incorporating a secure coprocessor in the database-as-a-service model. In *IWIA*, 2005.

[19] E. Mykletun and G. Tsudik. Aggregation queries in the database-as-a-service model. In *ESORICS*, 2006.

[20] S. R. M. Oliveira and O. R. Zaiane. Privacy preserving clustering by data transformation. In *SBBD*, Manaus, Amazonas, Brazil, 2003.

[21] L. Sweeney. k-anonymity: A model for protecting privacy. In *IJUFKS*, 2002.

[22] D. Xin, J. Han, and K. C.-C. Chang. Progressive and selective merge: computing top-k with ad-hoc ranking functions. In *SIGMOD*, 2007.

[23] D. Xin, J. Han, H. Cheng, and X. Li. Answering topk queries with multidimensional selections: The ranking cube approach. In *VLDB*, 2006.

[24] L. Zou and L. Chen. Dominant graph: An efficient indexing structure to answer top-k queries. In *ICDE*, 2008.

APPENDIX

A. PROOF OF UNIQUENESS OF INTERSECTION OF HYPERSPHERES

LEMMA 1. *Given $d+1$ hyperspheres in d-dimensional space that have common intersections. Let g_i and μ_i be the center and radius of the i-th hypersphere. If the d vectors $g_j - g_1$ for $j = 2$ to $d+1$ are linearly independent, the common intersection of the $d+1$ hyperspheres is exactly a point.*

PROOF. Let x be the intersection of the hyperspheres. We want to show that there is a unique solution to x. Since x lies on the $d+1$ hyperspheres, we have $d+1$ equations: $||g_i - x|| = \mu_i \Leftrightarrow ||x||^2 - 2g_i \cdot x = \mu_i^2 - ||g_i||^2$ for $i = 1$ to $d+1$. We can eliminate the term $||x||^2$ by subtracting the equations by the first equation. We have d equations: $(g_1 - g_j) \cdot x = \frac{1}{2}(\mu_j^2 - \mu_1^2 + ||g_1||^2 - ||g_j||^2)$ for $j = 2$ to $d+1$. Note that RHS of each equation is a constant and the d equations form a system of linear equations. If the vectors $g_j - g_1$ are linearly independent, there is a unique solution to the system of linear equations. □

B. ANALYSIS OF SIGNATURE LINKING ATTACK IN ATTACKING DRE

LEMMA 2. *Assume a DRE E is used to encrypt DB to get $E(DB)$. With a level-2 attack in which $H = \langle E(DB), P\rangle$, $\forall \epsilon > 0$, if $|P| \geq d+1+\frac{ln\epsilon}{ln\phi}$, Pr(signature collision) $< \epsilon$ where ϕ denotes the probability of a point being present in DB.*

PROOF. The probability of signature collision is the same as the probability of having a set of Q in $E(DB)$ that matches the signature generated by an arbitrary set of points P (so P is independent to $E(DB)$). Let $P_i \subseteq P$ ($Q_i \subseteq Q$ resp.) denotes the set of first i points in P (Q resp.). Let e_i be the event that the signature generated by P_i collides with the signature generated a randomly picked Q_i. We have $Pr(e_i) = Pr(e_{i-1})Pr(e_i|e_{i-1})$. In the base case $i = 1$, $sig(P_1) = \emptyset$. Any Q_1 gives the same signature. So, $Pr(e_1) = 1$. Next, we evaluate the value of $Pr(e_i|e_{i-1})$. Let x_i be the i-th point in P and y_i be the i-th point in Q. With one more point included in P_i, $sig(P_i)$ has $i-1$ more values (the pairwise distance from x_i to each of the first $i-1$ points in P)) than $sig(P_{i-1})$. $Pr(e_i|e_{i-1})$ depends on the probability of Q_i having the same $i-1$ values in the signature. These $i-1$ values depends on the value of y_i. When $i \geq d+2$, there are other $d+1$ points in Q_i and the distances from these points to y_i is given in $sig(P)$. We can establish $d+1$ equations of hyperspheres and y_i lies on the intersection of the hyperspheres. Since there are $d+1$ hyperspheres, there is only 1 possible solution of y_i (or no solution if the intersection falls outside the possible values domain). Each point in the domain has a probability of ϕ to be present in DB. So, $Pr(e_i|e_{i-1}) \leq \phi$ for $i \geq d+2$. When $i < d+2$, we can also establish $i-1$ equations of hyperspheres but the solution to y_i is not unique in general. Assume the domain on each dimension is J and J is discrete. We can pick $d+1-i$ dimensions in y_i and try all possible values for these dimensions ($|J|^{d+1-i}$ cases). In each case, there are $i-1$ variables remained in the equations and the dimension of the hyperspheres reduces to $i-1$. Given $i-1$ equations in $(i-1)$-dimensional domain, it gives 2 possible values of y_i (because the equations are quadratic). In summary, there are at most $\gamma = 2|J|^{d-i}$ possible solutions to y_i if $i < d+2$. This gives $Pr(e_i|e_{i-1}) \leq 1-(1-\phi)^\gamma$ for $i < d+2$. Hence, we can compute an upper bound for $Pr(e_i)$ for any i by $\Pi_{j=2}^{i} Pr(e_j|e_{j-1})$. When $|P| \geq d+1+\frac{ln\epsilon}{ln\phi}$, $Pr(e_{|P|}) \leq \phi^{|P|-d-1}\Pi_{j=2}^{d+1} 1-(1-\phi)^\gamma \leq \phi^{\frac{ln\epsilon}{ln\phi}} = \epsilon$ □

THEOREM 7. *Assume a DRE E is used to encrypt DB to get $E(DB)$. With a level-2 attack in which $H = \langle E(DB), P\rangle$, $\forall\epsilon > 0$, if $|P| \geq d+1+\frac{ln\epsilon}{ln\phi}$, $Pr(DB_A = DB) > 1-\epsilon$ where ϕ denotes the probability of a point being present in DB.*

PROOF. Followed from Lemma 2, the attacker can identify a unique I with probability $1-\epsilon$. So, he obtains the necessary knowledge for a level-3 attack. Next, he can execute the attack as shown in the proof of Theorem 1 and recover $DB_A = DB$.

In order to achieve a small signature collision probability ϵ, the number of points required in P in the attack should be at least $d+1+\frac{ln\epsilon}{ln\phi}$. As the points in the database are usually sparse, especially when the dimension is high, we expect a low probability of having a particular point present in the database. So $|ln\phi|$ is large. Hence $\frac{ln\epsilon}{ln\phi}$ is small and the number of points required to attack is practically small. So, if the attacker can find the distances between database points, the scheme cannot be secure at level-2 attacks. □

C. PROOF OF THEOREM 5

PROOF. If an encryption E is distance-recoverable, i.e., E is DRE, there exists a computational procedure f such that for all database points p_1, p_2 and any encryption key K such that $f(E(p_1,K), E(p_2,K)) = d(p_1,p_2)$. To prove E_T in Scheme 1 is not distance-recoverable, we show that such f does not exist. Given any two different database points p_1, p_2 and any encryption key M (a $(d+1)\times(d+1)$ invertible matrix), we will show later that we can construct two points x and y and an encryption key M_2 such that

(i) $a_1 = E_T(p_1, M_1) = E_T(x, M_2)$; and

(ii) $a_2 = E_T(p_2, M_1) = E_T(y, M_2)$; and

(iii) $d(p_1,p_2) \neq d(x,y)$.

If E is DRE, we have

$f(a_1,a_2) = f(E(p_1,K_1), E(p_2,K_1)) = d(p_1,p_2)$ and

$f(a_1,a_2) = f(E(x,K_2), E(y,K_2)) = d(x,y)$

It leads to a contradiction since $d(p_1,p_2) \neq d(x,y)$. E_T in Scheme 1 is not distance-recoverable. In the following, we describe how to construct x, y and M_2 that satisfies the above three conditions for any p_1, p_2, M_1 that $p_1 \neq p_2$.

Given two d-dimensional points p_1, p_2 and a $(d+1)\times(d+1)$ invertible matrix M. The encrypted value of p_1 (p_2 resp.) is $a_1 = E_T(p_1, M_1)$ ($a_2 = E_T(p_2, M_1)$ resp.). We construct $x = mp_1$ and $y = mp_2$ where $|m| \neq 0, 1$. So $d(x,y) \neq d(p_1,p_2)$. Condition (iii) is satisfied. Next, we construct $M_2 = \begin{pmatrix} \frac{1}{m}I_d & 0 \\ 0 & \frac{1}{m^2} \end{pmatrix} M_1$ where I_d is a $d \times d$ identity matrix. Note that M_2 must be invertible because $det(M_2) = \frac{1}{m^{d+2}} \neq 0$. We can use E_T to encrypt x and y using M_2. $E_T(x, M_2) = M_2^T(x, -0.5||x||^2)^T$

$$\begin{aligned} &= M_1^T \begin{pmatrix} \frac{1}{m}I_d & 0 \\ 0 & \frac{1}{m^2} \end{pmatrix} (mp_1, -0.5m^2||p_1||^2)^T \\ &= M_1^T(p_1, -0.5||p_1||^2) \\ &= E_T(p_1, M_1) = a_1 \end{aligned}$$

Similarly, we have $E_T(y, M_2) = a_2$. Condition (i) and (ii) are satisfied. All the three conditions are satisfied. □