LETTER

Security Analysis of Collusion-Resistant Nearest Neighbor Query Scheme on Encrypted Cloud Data*

Youwen ZHU[†a)], *Nonmember*, Tsuyoshi TAKAGI[†], *Member*, *and* Rong HU[†], *Nonmember*

SUMMARY Recently, Yuan *et al.* (IEEE Infocom '13, pp.2652–2660) proposed an efficient secure nearest neighbor (SNN) search scheme on encrypted cloud database. Their scheme is claimed to be secure against the collusion attack of query clients and cloud server, because the colluding attackers cannot infer the encryption/decryption key. In this letter, we observe that the encrypted dataset in Yuan's scheme can be broken by the collusion attack without deducing the key, and present a simple but powerful attack to their scheme. Experiment results validate the high efficiency of our attacking approach. Additionally, we also indicate an upper bound of collusion-resistant ability of any accurate SNN query scheme.

key words: *cloud computing, encrypted query, nearest neighbor, attack*

1. Introduction

With the prosperous development of cloud computing, more and more companies and individuals are motivated to store their data in cloud server and utilize the strong computation ability of cloud to query the outsourced dataset, because the cloud services can offer nice flexibility, scalability and IT cost-reduction. Due to privacy concerns, data owner may prefer to encrypt their dataset before outsourcing the data to cloud server. Nevertheless, encryption will make query challenging. As a widely-used algorithm, nearest neighbor query aims to find out a nearest point in a dataset for a given query point. Up to now, several works [1]–[6] have conducted the secure nearest neighbor (SNN) query on encrypted cloud dataset. For security and privacy of the outsourced database, SNN schemes generally consider the following three levels about the knowledge of attackers.

- *Level-1 attacker* only knows the encrypted database and encrypted query points. The level-1 attack corresponds to the ciphertext-only attack in the area of cryptography [7].
- *Level-2 attacker* learns, in addition to what level-1 attacker knows, some plain database points, but does not know their corresponding encrypted items. This attack is the same as the known-sample attack in [8]–[10].
- *Level-3 attacker* is assumed to know several plain database tuples and the corresponding encrypted items, beyond level-1 attacker. The attack follows the known input-output attack model in [8]–[10]. If the attacker can choose plaintexts to be encrypted and obtain their ciphertexts, the attack can be regarded as chosen-plaintext attack [7].

It is obvious that the knowledge of level-1 (or level-2) attacker is the subset of what a higher-level attacker learns. The higher-level attack is more powerful. Wong *et al.* proposed an asymmetric scalar-product-preserving encryption (ASPE) scheme in [1], and ASPE can support SNN query while preserving the data privacy under level-2 attack. The work [1] also presented an enhanced ASPE which aims to achieve data privacy against the above level-3 attack. However, Yao *et al.* [2] observe that the enhanced ASPE cannot resist level-3 attacker, and accurate SNN scheme is at least as hard as order-preserving encryption (OPE) [11]. Because the work [11] shows that secure OPE against the chosen-plaintext attack is unachievable, then, no accurate SNN scheme can be expected to be secure under the level-3 attack. The works [2]–[4] proposed three approximate SNN solution with different security and efficiency. Nevertheless, the existing approaches [1]–[4] assume that query users are full trustworthy, and data owner shares his key for encrypting/decrypting outsourced dataset with each query user. It causes the outsourced database will be completely broken if one untrusted query user disclose the key to attacker even under level-1 attack. The SNN scheme in [5] deals with the problem that query users reveal their knowledge to attacker. While the attacker learns query users' knowledge about the key of data owner, the scheme in [5] still can preserve data privacy against level-1 attack, comparing to be completely broken in [1]–[4]. As yet, [5] still cannot resist the collusion attack of cloud server and query clients in which situation the malicious colluding query clients will disclose all their data to cloud server. A collusion-resistant SNN solution is recently put forward by Yuan *et al.* [6], and the approach is claimed to be secure under level-3 attack even the attacker also learns several plain query points through colluding with malicious query clients, because the collusion attack cannot figure out the key of data owner. However, we observe if the attacker learns the encrypted database, some plain query points and their corresponding encrypted results, the outsourced database in Yuan's scheme [6] can be fully broken without deducing the key of data owner.

In this letter, we review the scheme by Yuan *et al.* [6], and present a simple but powerful attack to it by providing detail attack steps and extensive experiment results on both synthetic and real datasets. Then, we examine the security bound of collusion-resistant SNN scheme, and our results show that no accurate SNN solution can achieve data privacy against level-2 attack if the attacker simultaneously colludes

Manuscript received August 22, 2013.

[†]The authors are with the Institute of Mathematics for Industry, Kyushu University, Fukuoka-shi, 819–0395 Japan.

[*]This work was supported in part by the Japan Society for the Promotion of Science postdoctoral fellowship (No. P12045), the National Natural Science Foundation of China (No. 61370224).

a) E-mail: ywzhu@imi.kyushu-u.ac.jp

DOI: 10.1587/transinf.E97.D.326

with several query users.

2. Problem Formulation

Generally, the SNN scheme involves three types of participants: a data owner, a cloud server and several query clients.

The data owner has a private database **D** consisting of a certain number of n-dimensional points/tuples, and he wants to outsource **D** to cloud server in encrypted form. Assume **D** is a collection of m points, then **D** can be represented by the set $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_m\}$. For any $i \in [1, m]$, each tuple $\boldsymbol{b}_i = (b_{i1}, b_{i2}, \cdots, b_{in})$ can be regarded as a vector in n-dimensional space.

The cloud server has huge but limited computation and storage resource, and it is assumed to be semi-honest [12], [13], a.k.a., honest-but-curious, which means the cloud server will honestly follow the protocol steps and return correct computation output to data owner and query users, but the cloud server will try to infer the private information of other parties as much as possible based on the data it can legally access.

Each query client holds some private query points with the same form as the points in **D**. For each query point $\boldsymbol{q}_x = (q_{x1}, q_{x2}, \cdots, q_{xn})$, the client would like to obtain the index of its nearest neighbor (NN) point in **D** according to the Euclidean distance $\boldsymbol{dst}(\boldsymbol{b}_i, \boldsymbol{q}_x) = \sqrt{\sum_{t=1}^{n}(b_{it} - q_{xt})^2}$. Here, q_{xt} denotes the t-th dimension of query point $\boldsymbol{q}_x$.

The general processing steps of the system are (I) data owner generates the key for encrypting outsourced database and query points, (II) data owner encrypts database **D** into **D′**, and uploads the encrypted results **D′** to cloud server, (III) the query client submits query point $\boldsymbol{q}$ to data owner, (IV) data owner computes the encrypted query point $\boldsymbol{q}'$, and sends the encrypted values to cloud server, (V) at last, cloud server searches the NN point based on **D′** and $\boldsymbol{q}'$, and returns the index of NN point to the corresponding query client.

In the outsourcing storage and SNN query system, cloud server provides nothing but the computation and storage resource, thus, we only need to consider the privacy of other parties, and the privacy concerns include two aspects: privacy of outsourced database, privacy of query points. To protect the confidentiality of **D** and query points, cloud server is the potential attacker, since it can legally obtain **D′** and $\boldsymbol{q}'$. The attack model can be classified into three levels shown in Sect. 1. Besides, cloud server may collude with malicious query clients who will reveal all their data (such as query points) to the attacker: cloud server. Collusion-resistant SNN scheme aims to preserve data privacy against the collusion attack. In this paper, we analyze the collusion-resistant SNN scheme in [6] which is claimed to be secure against collusion attack of query clients and cloud server with knowledge of level-3 attacker. Our proposed simple but powerful attack shows that the scheme in [6] cannot resist the collusion attack of cloud server and query clients, even the cloud server only knows the encrypted database and nothing about any plain point in **D**. Further, we analyze the security bound of collusion-resistant SNN scheme, and it shows that no accurate SNN solution can securely resist the collusion attack if the cloud server has the knowledge of level-2 attack. Our results indicate, to resist collusion attack and achieve high security (against level-2 attack or higher), approximate search approach is the only feasible choice in outsourcing storage and SNN query environment.

3. Insecurity of the Approach by Yuan *et al.* [6]

3.1 Scheme by Yuan *et al.* [6]

As shown in [1], [5], [6], let $\boldsymbol{B}_i = \left(b_{i1}, b_{i2}, \cdots, b_{in}, (-0.5 * \sum_{t=1}^{n} b_{in}^2)\right)$, and $\boldsymbol{Q}_x = (q_{x1}, q_{x2}, \cdots, q_{xn}, 1)$, then, their dot products meet

$$\boldsymbol{B}_i \cdot \boldsymbol{Q}_x - \boldsymbol{B}_j \cdot \boldsymbol{Q}_x = -0.5 * \left(\boldsymbol{dst}(\boldsymbol{b}_i, \boldsymbol{q}_x)^2 - \boldsymbol{dst}(\boldsymbol{b}_j, \boldsymbol{q}_x)^2\right). \tag{1}$$

Therefore, the distance comparison can be exactly completed by checking the sign of $\left(\boldsymbol{B}_i \cdot \boldsymbol{Q}_x - \boldsymbol{B}_j \cdot \boldsymbol{Q}_x\right)$ without directly computing the Euclidean distance. Based on this conclusion, the SNN scheme by Yuan *et al.* [6] consists of the following two stages.

Stage 1: **Key Generation and Database Encryption**. This stage is locally completed by data owner, and the steps are (I) randomly generating two $(n+1)$-dimensional vectors $H = (h_1, h_2, \cdots, h_{n+1})$, $R = (r_1, r_2, \cdots, r_{n+1})$, and three invertible $(n+1) \times (n+1)$ matrices M_1, M_2, M_3 as the secret key, (II) for each $i \in [1, m]$, computing the $(n+1) \times (n+1)$ matrix D_i as follows,

$$D_i = \begin{Bmatrix} a_{i11} * b_{i1} & a_{i21} * b_{i2} & \cdots & a_{i(n+1)1} * b_{i(n+1)} \\ a_{i12} * b_{i1} & a_{i22} * b_{i2} & \cdots & a_{i(n+1)2} * b_{i(n+1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1(n+1)} * b_{i1} & a_{i2(n+1)} * b_{i2} & \cdots & a_{i(n+1)(n+1)} * b_{i(n+1)} \end{Bmatrix}.$$

Here, $b_{i(n+1)} = -0.5 * \sum_{t=1}^{n} b_{in}^2$ denote the $(n+1)$-th dimension of $\boldsymbol{B}_i$, and for $1 \leqslant k \leqslant n+1$, $A_{ik} = (a_{ik1}, a_{ik2}, \cdots, a_{ik(n+1)})$ is a random vector which meets $H \cdot A_{ik} = \sum_{j=1}^{n+1} h_j * a_{ikj} = 1$, (III) at last, data owner computes $C_H = H \times M_1^{-1}$ and $C_R = M_3^{-1} \times R^T$, and obtains $C_i = M_1 \times D_i \times M_2$ the ciphertext of $\boldsymbol{b}_i$, then, uploads $\{C_H, C_R, C_i\}$ to cloud server.

Stage 2: **Query Encryption and Computation**. The steps of this stage are (I) query client sends the query vector $\boldsymbol{q}_x = (q_{x1}, q_{x2}, \cdots, q_{xn})$ to data owner, (II) data owner extends $\boldsymbol{q}_x$ into $(n+1)$-dimensional $\boldsymbol{Q}_x = (q_{x1}, q_{x2}, \cdots, q_{xn}, 1)$, i.e., $q_{x(n+1)} = 1$, then, computes a $(n+1) \times (n+1)$ matrix F_c,

$$F_c = \begin{Bmatrix} e_{11} * q_{x1} & e_{12} * q_{x1} & \cdots & e_{1(n+1)} * q_{x1} \\ e_{21} * q_{x2} & e_{22} * q_{x2} & \cdots & e_{2(n+1)} * q_{x2} \\ \vdots & \vdots & \ddots & \vdots \\ e_{n1} * q_{xn} & e_{n2} * q_{xn} & \cdots & e_{n(n+1)} * q_{xn} \\ e_{(n+1)1} & e_{(n+1)2} & \cdots & e_{(n+1)(n+1)} \end{Bmatrix}.$$

For $1 \leqslant k \leqslant n+1$, let E_k denote the $(n+1)$-dimensional vector $(e_{k1}, e_{k2}, \cdots, e_{k(n+1)})$, then, the vector E_k is properly selected so that the equation $E_k \cdot R = \sum_{j=1}^{n+1} e_{kj} * r_j = 1$ holds, (III) data owner attains the encrypted query point $C_F = M_2^{-1} \times F_c \times M_3$, and submits C_F to the cloud, (IV) after receiving the encrypted query C_F, cloud server computes $P_i = C_H \times C_i \times C_F \times C_R$, then, searches the NN point through checking the sign of $(P_i - P_j)$ to determine which one of $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ is closer to $\boldsymbol{q}_x$.

The correctness of NN search in [6] can be guaranteed by the following reason:

$$\begin{aligned} P_i &= C_H \times C_i \times C_F \times C_R = H \times D_i \times F_c \times R^T \\ &= \Big(b_{i1} \sum_{j=1}^{n+1} h_j * a_{i1j}, b_{i2} \sum_{j=1}^{n+1} h_j * a_{i2j}, \cdots, \\ &\qquad b_{i(n+1)} \sum_{j=1}^{n+1} h_j * a_{i(n+1)j}\Big) \\ &\quad \cdot \Big(q_{x1} \sum_{j=1}^{n+1} e_{1j} * r_j, q_{x2} \sum_{j=1}^{n+1} e_{2j} * r_j, \cdots, \\ &\qquad q_{x(n+1)} \sum_{j=1}^{n+1} e_{(n+1)j} * r_j\Big) \\ &= \sum_{j=1}^{n+1} b_{ij} * q_{xj} = \boldsymbol{B}_i \cdot \boldsymbol{Q}_x \end{aligned} \tag{2}$$

Based on Eqs. (1) and (2), the cloud server can exactly obtain $\boldsymbol{dst}(\boldsymbol{b}_i, \boldsymbol{q}_x) > \boldsymbol{dst}(\boldsymbol{b}_j, \boldsymbol{q}_x)$ if $(P_i - P_j) < 0$, otherwise $\boldsymbol{dst}(\boldsymbol{b}_i, \boldsymbol{q}_x) \leqslant \boldsymbol{dst}(\boldsymbol{b}_j, \boldsymbol{q}_x)$.

3.2 Security Guarantee and Our Attack

Yuan *et al.* [6] claims their scheme is secure against the collusion attack of a level-3 attacker and some query clients, because the collusion cannot recover the secret key $\{H, R, M_1, M_2, M_3\}$ of data owner. However, we observe the scheme by Yuan *et al.* [6] can be broken through an efficient attacking approach. Our attack shows that even a level-1 attacker, if colluding with several query clients, can completely recover the private points without solving the key of data owner. For a level-2 or level-3 attacker, it can also break the encryption scheme in [6] by collusion with query clients, since the higher-level attacker knows all the information that level-1 attacker learns.

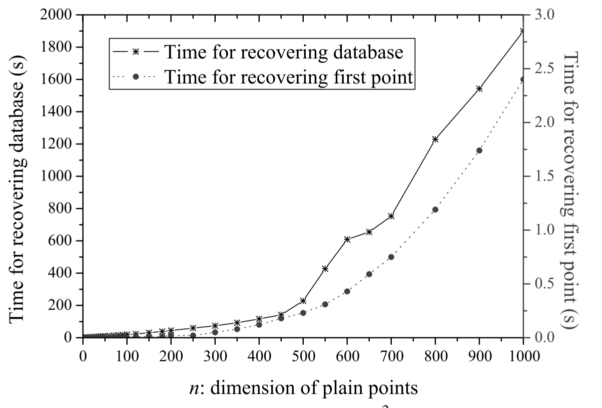
We introduce our attack by showing the method that level-1 attacker and some colluding query clients recover the private database points $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_m\}$. While colluding with query clients, the level-1 attacker can learn not only the encrypted database $\mathbf{D}'$, but also some query points and the corresponding encrypted query values. Assume the attacker obtains s plain query points $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_s\}$ and the encrypted query $C_F^{(a)}$ for each $\boldsymbol{q}_a$ $(a \in [1, s])$, in addition to the encrypted database points in $\mathbf{D}'$. Then, based on the foregoing equation (2), for each $1 \leqslant a \leqslant s$, the attacker can build an equation $\boldsymbol{B}_i \cdot \boldsymbol{Q}_a = P_i^{(a)}$ where only $\boldsymbol{B}_i$ is unknown. Here, $P_i^{(a)} = C_H \times C_i \times C_F^{(a)} \times C_R$, $\boldsymbol{Q}_a = (q_{a1}, q_{a2}, \cdots, q_{an}, 1)$, i.e., for $1 \leqslant j \leqslant n$, the j-th dimensions of $\boldsymbol{Q}_a$ and $\boldsymbol{q}_a$ are equal to each other, and the last dimension $q_{a(n+1)}$ of $\boldsymbol{Q}_a$ is the constant 1. While s is large enough $(s > n)$, the attacker can find out $(n + 1)$ points $\{\boldsymbol{q}_{v_1}, \boldsymbol{q}_{v_2}, \cdots, \boldsymbol{q}_{v_{n+1}}\}$ so that the matrix $M_Q = [\boldsymbol{Q}_{v_1}^T, \boldsymbol{Q}_{v_2}^T, \cdots, \boldsymbol{Q}_{v_{n+1}}^T]$ is invertible. For all $j \in [1, n + 1]$, let $C_F^{(v_j)}$ be the encrypted value of $\boldsymbol{q}_{v_j}$ and $P_i^{(v_j)} = C_H \times C_i \times C_F^{(v_j)} \times C_R$. Then, $\boldsymbol{B}_i \times M_Q = V_C^{(i)}$ where $V_C^{(i)}$ is the $(n+1)$-dimensional vector $\left(P_i^{(v_1)}, P_i^{(v_2)}, \cdots, P_i^{(v_{n+1})}\right)^T$. At last, the attacker can figure out $\boldsymbol{B}_i = V_C^{(i)} \times M_Q^{-1}$, and the first n dimensions of $\boldsymbol{B}_i$ is just the plain database point $\boldsymbol{b}_i$. Similarly, for any other $j \in [1, m]$, the point $\boldsymbol{b}_j$ can be recovered through computing the first n rows of $V_C^{(j)} \times M_Q^{-1}$. Therefore, if colluding with several query clients, the level-1 attacker can efficiently break the scheme by Yuan *et al.* [6]. To attain the plain database, it requires $O(n^3)$ computation time to implement matrix inversion (for computing M_Q^{-1}) once, and then $O(n^2)$ to compute $V_C^{(i)} \times M_Q^{-1}$ for each point $\boldsymbol{b}_i$, thus, the total computation complexity of recovering all points in $\mathbf{D}$ is $O(n^3 + m * n^2)$.

3.3 Experiment Results

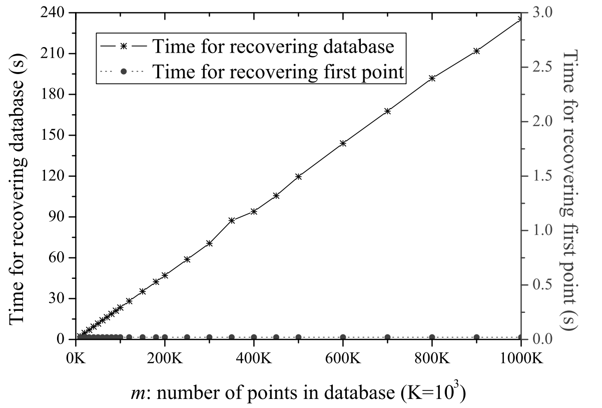
To further study the effectiveness of our attack, we implement it on both synthetic and real datasets. All experiments are performed on Linux with Intel Core i5 2.4 GHz CPU and 3.8 GB memory.

Using the scheme by Yuan *et al.* [6], we encrypt a real dataset ‘Shuttle’ from the UCI repository [14]. The ‘Shuttle’ dataset contains 58K points ($\mathrm{K} = 10^3$), and each point has 9 numerical dimensions. Then, we implement our attack algorithm by taking as input the encrypted ‘Shuttle’ dataset, 10 random plain query points, 10 corresponding encrypted queries and the intermediate values P_i. The experiment results show that our approach can recover the plain ‘Shuttle’ dataset within 0.5 seconds.

For synthetic databases, we conduct two groups of experiments. In the first group, we fix m = 200K, and the dimension n ranges from 2 to 1000. The second group varies m from 10K to 1000K, and fixes n = 200. The computation times of our attack are shown in Fig. 1. While recovering first point in each database, we need to find the matrix M_Q and compute its inverse M_Q^{-1}. Then, the matrix M_Q^{-1} can be used for recovering other points in the same database, and it is unnecessary to compute it again. Figure 1 shows our attack can fast recover the plain database. If fixing m = 200K, even the dimension is as high as 1000, we can figure out the first point within 2.5 seconds, and it costs about 1900 seconds to recover all the 200K points in the 1000-dimensional database. While the dimension n is set to 200, computation time of our attack almost scales linearly to the number of points in database, and our attack takes less than 240 seconds to recover the large scale database consisting of 1000K points. Since the dimension is fixed, the time for recovering first point shown in Fig. 1 (b) is always 0.02 seconds. In general, the results show that the scheme by Yuan *et al.* [6] can be fast broken, and our attack is of high efficiency.



(a) m = 200K (K = 10^3)



(b) n = 200

Fig. 1 Computation time of our attack.

4. Security Bound of Collusion-Resistant SNN Query over Encrypted Cloud Data

In the outsourcing and SNN query scenario, the computation for searching NN point is completed by cloud server. Assume $\boldsymbol{b}_{u_1}$ is the nearest point to query point $\boldsymbol{q}_x$. For any encryption scheme with supporting accurate SNN query, cloud server (the potential attacker) always can obtain $\boldsymbol{dst}(\boldsymbol{b}_{u_1}, \boldsymbol{q}_x) \leqslant \boldsymbol{dst}(\boldsymbol{b}_y, \boldsymbol{q}_x)$ for any $1 \leqslant y \leqslant m$ and $y \neq u_1$. Then, the attacker can further second NN point $\boldsymbol{b}_{u_2}$ from the remaining $(m-1)$ points, and it can be deduced $\boldsymbol{dst}(\boldsymbol{b}_{u_1}, \boldsymbol{q}_x) \leqslant \boldsymbol{dst}(\boldsymbol{b}_{u_2}, \boldsymbol{q}_x) \leqslant \boldsymbol{dst}(\boldsymbol{b}_y, \boldsymbol{q}_x)$ for any $y \neq u_1, u_2$. Similarly, the attacker can exactly determine the distance ordering $\boldsymbol{dst}(\boldsymbol{b}_{u_1}, \boldsymbol{q}_x) \leqslant \boldsymbol{dst}(\boldsymbol{b}_{u_2}, \boldsymbol{q}_x) \leqslant \cdots \leqslant \boldsymbol{dst}(\boldsymbol{b}_{u_m}, \boldsymbol{q}_x)$.

Assume a level-3 attacker knows $\mathbf{D}'$, the plain database tuples $\{\boldsymbol{b}_{z_1}, \boldsymbol{b}_{z_2}, \cdots, \boldsymbol{b}_{z_\theta}\}$ and the corresponding encrypted value C_{z_i} of each $\boldsymbol{b}_{z_i}$ $(1 \leqslant i \leqslant \theta)$. While colluding with some query clients, the attacker can acquire some plain query points $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_s\}$, in addition to the foregoing distance ordering and $\{\mathbf{D}', \boldsymbol{b}_{z_i}, C_{z_i} : 1 \leqslant i \leqslant \theta\}$. Correspondingly, the attacker can also figure out the distances $\boldsymbol{dst}(\boldsymbol{b}_i, \boldsymbol{q}_x)$ for all $i \in [1, \theta]$, $x \in [1, s]$. Since Boldyreva *et al.* [11] have verified that it is impossible to attain a secure OPE against chosen-plaintext attack, the attacker can recover all the distances $\{\boldsymbol{dst}(\boldsymbol{b}_i, \boldsymbol{q}_x) : 1 \leqslant i \leqslant m, 1 \leqslant x \leqslant s\}$ based on his knowledge about the distance sorting and several known distances. Then, for any unknown database point $\boldsymbol{b}_i$ $(1 \leqslant i \leqslant m$, and $i \notin \{z_1, z_2, \cdots, z_\theta\})$, the attacher can attain the dot product $\boldsymbol{B}_i{\cdot}\boldsymbol{Q}_x = \boldsymbol{b}_i{\cdot}\boldsymbol{q}_x - 0.5{*}\sum_{j=1}^{n} b_{ij}^2 = -0.5{*}\Big(\boldsymbol{dst}(\boldsymbol{b}_i, \boldsymbol{q}_x) - \sum_{j=1}^{n} q_{xj}^2\Big)$ for all $x \in [1, s]$, and further recover the value of $\boldsymbol{b}_i$ based on our attacking method proposed in Sect. 3.2. As a result, it is impossible to construct an accurate SNN query scheme on encrypted cloud dataset while preserving data privacy against the collusion of a level-3 attacker and query clients.

We further analyze the collusion of a level-2 attacker and query clients. Assume the level-2 attacker learns $\mathbf{D}'$ and the plain database tuples $\{\boldsymbol{b}_{z_1}, \boldsymbol{b}_{z_2}, \cdots, \boldsymbol{b}_{z_\theta}\}$, but does not know which items in $\mathbf{D}'$ are the corresponding encrypted results of the known plain points. Nevertheless, the attacker can require the colluding query clients to submit θ queries $\{\boldsymbol{q}_{z_1}, \boldsymbol{q}_{z_2}, \cdots, \boldsymbol{q}_{z_\theta}\}$ such that $\boldsymbol{q}_{z_i} = \boldsymbol{b}_{z_i}$. For $i \in [1, \theta]$, let $C_F^{(z_i)}$ be the encrypted query of $\boldsymbol{q}_{z_i}$, then, the nearest point of $C_F^{(z_i)}$ in $\mathbf{D}'$ will just be the encrypted item of $\boldsymbol{b}_{z_i}$, due to $\boldsymbol{dst}(\boldsymbol{b}_{z_i}, \boldsymbol{q}_{z_i}) = 0$. Therefore, the level-2 attacker can determine the encrypted value of each $\boldsymbol{b}_{z_i}$, and it will become a level-3 attacker after learning the corresponding encrypted items. Since we have shown that no accurate SNN solution can resist the collusion of a level-3 attacker and query clients, it is also impossible for any accurate SNN scheme to achieve data privacy against the collusion of a level-2 attacker and query clients.

Consequently, we attain the security bound that any accurate SNN scheme on encrypted cloud data cannot resist the collusion of a level-2 (or higher-level) attacker and query clients.

5. Conclusion

In this letter, we proposed an efficient attacking method which can fast break the SNN scheme by Yuan *et al.* [6]. Additionally, we confirmed that no accurate SNN solution over encrypted cloud dataset can resist the collusion of a level-2 (or higher-level) attacker and query clients.

References

[1] W.K. Wong, D.W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," Proc. 35th SIGMOD, pp.139–152, 2009.

[2] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," Proc. 29th IEEE ICDE, pp.733–744, 2013.

[3] H. Xu, S. Guo, and K. Chen, "Building confidential and efficient query services in the cloud with RASP data perturbation," IEEE Trans. Knowl. Data Eng., DOI: 10.1109/TKDE.2012.251, Dec. 2012.

[4] M.L. Yiu, I. Assent, C.S. Jensen, and P. Kalnis, "Outsourced similarity search on metric data assets," IEEE Trans. Knowledge Data Eng., vol.24, no.2, pp.338–352, 2012.

[5] Y. Zhu, R. Xu, and T. Takagi, "Secure k-NN computation on encrypted cloud data without sharing key with query users," Proc. ASIACCS Workshop on Security in Cloud Computing, pp.55–60, 2013.

[6] J. Yuan and S. Yu, "Efficient privacy-preserving biometric identification in cloud computing," Proc. 32nd IEEE INFOCOM, pp.2652–2660, 2013.

[7] H. Delfs and H. Knebl, Introduction to cryptography: Principles and applications, Springer, 2002.
[8] K. Liu, C. Giannella, and H. Kargupta, "An attacker's view of distance preserving maps for privacy preserving data mining," Proc. 10th PKDD, pp.297–308, 2006.
[9] S. Guo and X. Wu, "Deriving private information from arbitrarily projected data," Proc. 11th PAKDD, pp.84–95, 2007.
[10] K. Liu, C. Giannella, and H. Kargupta, "A survey of attack techniques on privacy-preserving data perturbation methods," Privacy-Preserving Data Mining, pp.359–381. Springer, 2008.
[11] A. Boldyreva, N. Chenette, Y. Lee, and A. O'neill, "Order-preserving symmetric encryption," EUROCRYPT, pp.224–241, 2009.
[12] O. Goldreich, Foundations of Cryptography: Volume II, Basic Applications, Cambridge University Press, Cambridge, 2004.
[13] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," Proc. ACM CCS, pp.965–976, 2012.
[14] UCI machine learning repository, http://archive.ics.uci.edu/ml/