Inference Attacks in Peer-to-Peer Homogeneous Distributed Data Mining

Josenildo Costa da Silva[1] and **Matthias Klusch**[1] and **Stefano Lodi**[2] and **Gianluca Moro**[2]

Abstract. Spontaneous formation of peer-to-peer agent-based data mining systems seems a plausible scenario in years to come. However, the emergence of peer-to-peer environments further exacerbates privacy and security concerns that arise when performing data mining tasks. We analyze potential threats to data privacy in a peer-to-peer agent-based distributed data mining scenario, and discuss inference attacks which could compromise data privacy in a peer-to-peer distributed clustering scheme known as KDEC.

1 INTRODUCTION

In the last decade, the automated extraction of patterns from large or huge centralized datasets, or *data mining* (DM), has become popular in many organizations. Meanwhile, the emergence of internet as a huge, distributed data sharing system has encouraged the creation of public and private distributed data sources. As a consequence, a considerable amount of research results in the emerging field of *distributed data mining* (DDM) [15] have been accomplished.

Agents have been been employed in the design of DDM systems [26, 14, 15] and could provide a paradigm which naturally fits a DDM environment [17], raising however concerns of data security and trustworthiness. A natural scenario which could emerge in years to come is that of spontaneous formation of agent societies, in which each agent, representing one or more organizations or individuals, offers and receives contributions by interacting with others to achieve similar goals. This kind of agent societies would not have any centralized coordination to permit both high scalability and large participation. Such expectations seem very plausible in view of current practice in the domain of *peer-to-peer* (P2P) systems. It is reasonable to think that soon peer-to-peer networks will go beyond sharing multimedia files by offering more sophisticated services, such as data mining. Performing DDM in such extremely open distributed systems exacerbates data privacy and security issues. In most cases, the peers would deny data sharing due to privacy regulations, or loss of competitive advantage. Unfortunately, the byproducts of executing a DDM algorithm might be sufficient to infer the original data either exactly, approximately, or probabilistically, depending on the pattern. This is an instance of the *inference problem*, which has been investigated in statistical databases, and by the data mining community.

The structure of this paper is the following. Section 2 reviews work on the inference problem. Section 3 describes a P2P homogeneous DDM scheme and classifies its vulnerabilities. Section 4 describes an algorithm to approximately infer data points in a homogeneous distributed data clustering scheme known as KDEC. Section 6 concludes the paper.

[1] Deduction and Multiagent Systems, German Research Centre for Artificial Intelligence Stuhlsatzenhausweg 3 66123 Saarbrücken, Germany, email: {jcsilva,klusch}@dfki.de

[2] Department of Electronics, Computer Science and Systems, University of Bologna, Via Risorgimento 2 I-40136 Bologna BO, Italy, email: {slodi,gmoro}@deis.unibo.it

2 THE INFERENCE PROBLEM

The term *inference problem* appeared first in the work on Statistical Databases in the mid 1970's. The problem is that one might disclose confidential information about individual entity by posing queries on aggregate statistics Inference control in statistics database has been extensively studied and a number of inference control mechanism were developed [8]. Data mining techniques makes the inference problem even worse as data mining provides a powerful tool for information extraction. In the following, we briefly review recent approaches to solve the inference problem in both fields, data mining and distributed data mining.

Related Work in Data Mining. Most recent efforts addressing the privacy issue in data mining include the *sanitization* and the *data distortion* approaches.

Sanitization process aims to clean the data base so that sensitive patterns cannot be mined. It was developed primarily to handle secure issues in association rule mining. These techniques were proposed by Atallah et al. [2] and Dasseni et al. [5]. The idea is basically to remove or modify items in a database to reduce or increase the support of some frequent itemsets. By doing this, the data owner expects to hide some sensitive frequent itemsets with as little as possible impact on other non-sensitive frequent itemsets. Further developments of this technique can be found in Saygin et al. [23, 24], Oliveira and Zaïne [19], Johnsten and Raghavan [10] and Clifton [3].

Data distortion (data *perturbation*, or data *randomization*) emphasizes the protection of the individual data against one miner through modification of the original data. This technique assumes that the distorted data, and distribution function of random data used to distort the original data, can be used to generate an approximation to the original probability distribution, without revealing any of the original values. These works are mainly influenced by the results of research in the field of statistical databases. It can be used by mining algorithms that use probability distributions rather than data values as input such as kernel-based clustering. Data distortion has been applied to decision tree based classification [1] and association rules [22]. This idea has been further elaborated by Evfimievski et al. [7]. However, one weakness of using data distortion for preserving data privacy is that under certain conditions randomization does not prevent an attacker from reconstructing original data with reasonably high probability [12, 13].

Related Work in Distributed Data Mining Preventing data inference attacks in open environments is difficult if not impossible

[11], mainly due to problems of scalability, and trustworthiness of involved parties. The latter problem can be coped with by approaches to secure multi-party computation.

Secure Multi-party Computation (SMC) aims to compute a given function in a distributed fashion with minimum information share between involved parties. After SMC each party only knows the final result but no intermediate results that have been achieved in due course of their computation. In [21, 4], it has been shown in general that SMC can be applied to the data mining process with only a few modifications of the original idea with respect to data input size. Related work to the application of SMC to association rule mining and decision-tree based data classification are [12, 27, 28], respectively, [18, 6].

None of the above approaches were conceived aiming to solve the inference problem. In the literature the *privacy problem* is concerned with protecting individual data, while the *inference problem* relates to secrecy as well. Both problems are closely related but the question is whether the solutions to the privacy problem are sufficient to solve the inference problem.

3 AN AGENT-BASED HOMOGENEOUS DDM SCHEME

In this section we introduce an environment model for the problem of combining local patterns of homogeneously distributed data. Moreover we present our definition of inference in a distributed data mining schema and classify the possible inference attack scenarios in such a model.

3.1 Combining local patterns of homogeneously distributed data

Let Π be a pattern extraction problem, defined by an instance (S, Γ), consisting of a relation $S \subseteq \mathbb{R}^n$ and a set of parameters Γ, and a specification of a solution pattern $p_\Pi[S, \Gamma]$ for every instance (S, Γ). A *homogeneous distributed extraction problem* is defined as an extraction problem in which the single relation is replaced by a homogeneous set of relations $\{D^j : D^j \subseteq \mathbb{R}^n, j = 1, \ldots, m\}$. Every given extraction problem Π can be turned straightforwardly into a distributed one, by (i) allowing sets of homogeneous relations instead of the single relation S in the definition of Π, and (ii) adding the following constraint: When the computation halts, the solution pattern $p_\Pi[S, \Gamma]$ must be stored in the data space of every process which initially stored a D^j. In many cases of practical interest, a solution is independent of data location and should not be altered by horizontally partitioning the data among different sites. We therefore say that a homogeneous distributed extraction problem Ξ *extends* an extraction problem Π if and only if $p_\Xi[\{D^1, \ldots, D^m\}, \Gamma] = p_\Pi[\bigcup_{j=1}^m D^j, \Gamma]$, for every instance $(\{D^1, \ldots, D^m\}, \Gamma)$.

A computable function Φ_Π computing solution patterns for problem Π is a *solution function* for Π. By *additive summary* we mean a function σ on the set of relations on $\mathbb{R}^n$ into a commutative group $(G, +, -, 0)$, such that $\sigma(\bigcup_{j=1}^m S_j) = \sum_{j=1}^m \sigma(S_j)$, where "$\sum$" iterates over "+". A solution function Φ_Π which is a composition of additive summaries forms the basis of a general P2P homogeneous DDM scheme for the solution of problem Ξ. Let Ψ_Π be a p-ary computable function and let $\sigma_1, \ldots, \sigma_p$ be additive summaries such that $\Phi_\Pi(S) = \Psi_\Pi(\sigma_1(S), \ldots, \sigma_p(S))$ for all S. Further let $\mathcal{P} = \{P^j : j = 1, 2, \ldots l\}$ be a set of peer sites cooperating to solve an instance of Ξ, and let $\mathcal{L} = \{L^j : j = 1, \ldots, m \leq l\} \subseteq \mathcal{P}$ be the set of all peers containing a part of the distributed dataset S, such that D^j is stored on L^j. To solve Ξ, there exists a $P \in \mathcal{P}$ executing function Ψ_Π. When in the course of computation, P needs a summary $\sigma_k(S)$, it broadcasts a request to $\mathcal{P}$. Every L^j computes $\sigma_k(D^j)$ and sends it to a *helper* or *facilitator* network in $\mathcal{P}$ where $\sigma_k(S) = \sum_{j=1}^m \sigma_k(D^j)$ is computed and sent back to the requesting peer. The computation halts soon after P has sent the solution pattern to $\mathcal{L}$.

Instantiations of the scheme already exist in the literature, or can be derived straightforwardly, for a significant number of distributed pattern extraction problems, including *density-based data clustering* [25], *multivariate regression*, and *association rules*. Note also that the data owning peers L^j could use the well-known *secure sum* protocol [4] to compute the sum of local summaries $\sigma_k(D^j)$ instead of sending the summaries directly to the helpers. Although using such protocol would make the opportunities to learn the summaries less likely, we will be concerned about inference techniques and their application scenarios in the more traditional setting with helpers acting as buffers for the peers. Scenarios in which helpers are not malicious apply to the secure sum setting as well.

3.2 Inference attacks in a DDM system

Informally, we see inference in a DDM system as the process where one or more peer sites learn any confidential information (models, patterns, or the data themselves) about the dataset owned by other peers during a data mining session. We consider different inference attack scenarios in DDM system from inside and outside the set of mining parties.

Inside Attack Scenarios. In an *inside attack scenario*, one or multiple peer sites $L^j \in \mathcal{P}$ try to infer useful information from the original data owned by other peers in the mining group $\mathcal{P}$. Hence, the attacker has complete knowledge of the data mining process and parameters that are used by the group for individual mining tasks at hand. We distinguish between single and coalition attacks depending on the number of attackers. In a *single attack* only one of the peers behaves maliciously, trying to disclose information. One example of this attack is the Malicious Helper Attack. In this case a helper peer behaves maliciously gathering all information the sites sent to it and uses this information later to disclose information about the data in the sites. In this case, the malicious helper has the possibility to assign precisely each data inferred to the different participating sites. In a *coalition attack* scenario, with n sites, k sites collude to attack a chosen site $(1 < k \leq n - 1)$.

Another important aspect of the attack, which is independent of the number of peer sites that take part in it, is the attack strategy. Some examples of strategies are *probe* and *eavesdropping*. In a *probe attack* one peer sets its contributed data to some null value, so that the global result of a distributed data mining process will reflect the mining results of the other peer only. In the 2-sites case it is equivalent to mining the database from the other peer directly. In an *eavesdropping attack* the malicious peer behaves quite normally inside the group, but it stores all information possible about the other participants in the group.

Outside Attack Scenarios. Let $\mathcal{P}$ be a set of peer sites whose members are engaged in a cooperative data mining task. An *outside attack* is a scenario where a malicious peer site $P \notin \mathcal{P}$ tries to infer some information from the data owned by the members of $\mathcal{P}$. Since $P \notin \mathcal{P}$, we can assume that P knows nothing (or knows just little) about the mining parameters that have been agreed upon by the group $\mathcal{P}$. However, we assume that the attacker site P can steal information via an eavesdropping channel. The different possibility of an outside attack

scenarios are based on what information P has stolen.

In the next section, we provide an example of agent-based distributed data mining and discuss instances of different types of inference attacks.

4 AGENT-BASED DISTRIBUTED DATA CLUSTERING: THE KDEC SCHEME

Let $S = \{\vec{x}[i] : i = 1, \ldots, N\} \subseteq \mathbb{R}^n$ be a dataset of points. For any S, let $\mathcal{C}(S) = \{C_k(S)\} \subseteq 2^S$ be a clustering of S, whose elements are pairwise disjoint. Let L^j, $j = 1, \ldots, m$ be a finite set of sites. Let each site L^j store one local dataset D^j and $S = \bigcup_{j=1}^{m} D^j$. The problem of *homogeneous distributed data clustering* (homogeneous DDC) is to find for $j = 1, \ldots, m$, a site clustering $\mathcal{C}_j$ residing in the data space of L^j, such that $\mathcal{C}_j = \{C_k(S) \cap D^j : k = 1, \ldots, |\mathcal{C}(S)|\}$ (*correctness requirement*), time and communications costs are minimized (*efficiency requirement*), and, at the end of the computation, the size of the subset of S which has been transferred out of the data space of any site L^j is minimized (*privacy requirement*). Homogeneous DDC is a homogeneous distributed extraction problem which extends the problem specified by the pair $(S, \mathcal{C}(\cdot))$.

In [16], a distributed clustering scheme called KDEC was introduced. KDEC can be applied as a solution to homogeneous DDC when the clustering specification $\mathcal{C}(\cdot)$ is based on a nonparametric kernel density estimate of the data [20, 25]. The main idea is that kernel density estimates are: (i) additive for homogeneous distributed datasets, and (ii) can be transmitted in sampled form in order to hide the data points, which are otherwise explicit in the representation of a kernel estimate. Each site L^j computes its local kernel density estimate (LDE), which we denote by $\hat{\varphi}_{K,h}[D^j](\cdot)$, as follows:

$$\hat{\varphi}_{K,h}[D^j](\vec{x}) = \sum_{\vec{x}[i] \in D^j} K\left(\frac{d(\vec{x}, \vec{x}[i])}{h}\right) \tag{1}$$

where h is a scaling parameter called *window width*, d is a distance function, and K is a real-valued, non-negative, non-increasing function on $\mathbb{R}_+ \cup \{0\}$ called *kernel function* (e.g. the restriction of the Gaussian function to $\mathbb{R}_+ \cup \{0\}$). Equation (1) defines the estimate at $\vec{x} \in \mathbb{R}^n$ as a weighted sum of scaled distances of data points from $\vec{x}$, i.e., according to the proximity of the data points to $\vec{x}$. In KDEC it is assumed that all sites agree on using the same distance d, kernel K, and window width h. (For brevity, we will omit K, h from our notation in the following.) Therefore, the sum of the LDEs equals the global density estimate (GDE) $\hat{\varphi}[S](\cdot)$.

The approach exploits multi-dimensional information sampling to minimize communications among sites and to increase privacy, for it adopts a representation of the estimate which makes no explicit reference to the data points. Before sending its LDE to the helper, each site transforms it into sampled form. For any $\vec{x} \in \mathbb{R}^n$, let $x_1, \ldots, x_n$ be its components. Let $\vec{\tau} = [\tau_1, \ldots, \tau_n] \in \mathbb{R}^n$ be a vector of sampling periods and let $\vec{z} \bullet \vec{\tau}$ denote $[\tau_1 z_1, \ldots, \tau_n z_n]$, where $\vec{z} \in \mathbb{Z}^n$. Let $R(\vec{z_1}, \vec{z_2}) \subseteq \mathbb{Z}^n$ be the n-dimensional rectangle having diagonal $(\vec{z_1}, \vec{z_2})$. The sampled form of $\hat{\varphi}[D^j](\cdot)$ is the finite real sequence $\{\hat{\varphi}_{\vec{z}}[D^j]\}$ defined by:

$$\hat{\varphi}_{\vec{z}}[D^j] = \{\hat{\varphi}[D^j](\vec{z} \bullet \vec{\tau}) \ : \ \vec{z} \in R(\vec{z_1}, \vec{z_2})\} \tag{2}$$

where the sampling parameters $\vec{z_1}, \vec{z_2} \in \mathbb{Z}^n$ and $\vec{\tau} \in \mathbb{R}^n$ are previously agreed among the peer sites.

Since density estimates in sampled form are additive, the helper is able to compute the sampled GDE using Equation (3) for all $\vec{z} \in R(\vec{z_1}, \vec{z_2})$ as sum of the sampled LDEs, and send it back to the peers:

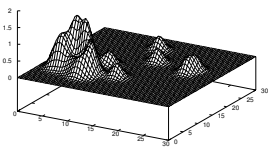

Figure 1. LDE at site j

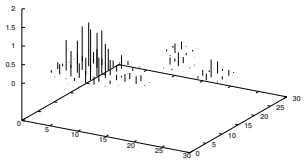

Figure 2. Sampled LDE at site j

$$\hat{\varphi}_{\vec{z}}[S] = \sum_{j=1}^{m} \hat{\varphi}_{\vec{z}}[D^j] \tag{3}$$

From the global sampled density estimate $\hat{\varphi}_{\vec{z}}[S]$ the local sites can approximate the true GDE using the interpolation formula

$$\sum_{\vec{z} \in R(\vec{z_1}, \vec{z_2})} \hat{\varphi}_{\vec{z}}[S] \ sinc\left(\frac{x_1}{\tau_1} - z_1\right) \cdots sinc\left(\frac{x_n}{\tau_n} - z_n\right) \tag{4}$$

where

$$sinc(x) = \begin{cases} 1 & \text{if } x = 0, \\ \frac{\sin \pi x}{\pi x} & \text{otherwise.} \end{cases}$$

Expression (4) is an application of the well-known Whittaker-Shannon sampling series (see e.g. [9]) to the domain of density estimates. Note that the function represented by (4) is not extensionally equal to the kernel global estimate $\hat{\varphi}[S](\cdot)$ both because kernel estimates are not band-limited to any frequency region, and because of the truncation in the series. However, it was shown in [16] that the approximation introduces only a small error and consequently we can choose $\vec{\tau}$ so that the Fourier transform of the estimate $\hat{\varphi}[S](\cdot)$ is negligible in $\mathbb{R}^n \setminus [-\pi/\tau_1, \pi/\tau_1) \times \cdots \times [-\pi/\tau_n, \pi/\tau_n)$, and $\vec{z_1}, \vec{z_2}$ such that the estimate is negligible in $\mathbb{R}^n \setminus R(\vec{z_1}, \vec{z_2})$.

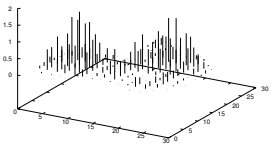

Figure 3. Sampled GDE at the Helper Agent

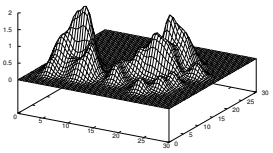

Figure 4. Reconstructed GDE at Site j

Finally, each peer site can use the reconstructed global density estimate to cluster its local data. To this end, it uses a gradient-driven hill climbing procedure to find local maxima in the global density estimate. All points that can be connected to one given local maximum are labeled to the same cluster.

5 INFERENCE ATTACKS IN KDEC

Let us analyse the KDEC scheme with respect to our inference attack framework.

5.1 Inside attacks in KDEC

In KDEC the inside attack is always possible. Remember that an inside attacker knows all the parameters that are agreed before the KDEC protocol starts, i.e., the kernel function K, the window width h, and the distance function d. Moreover, the sampled global density estimation is distributed back to all sites that cooperate in the

Algorithm 1 *pointhunt*

Input: x_0, S_0;
Output: x, S';
1: $x \leftarrow \mathit{search}(x_0, S_0)$;
2: $\delta \leftarrow |\mathit{density}(x) - \hat{\varphi}[S_0](x)|$;
3: $S' \leftarrow S_0$;
4: **if** $x \neq x_0$ **then**
5: $\quad s_{new} \leftarrow \mathit{guess}(x, \delta)$;
6: $\quad S' \leftarrow S_0 \cup \{s_{new}\}$;
7: **end if**
8: **return** x, S'

function *guess* (x, δ);
1: **return** $x + h * K^{-1}(\delta)$;
end function

function *search* (x, S_0);
1: $Y \leftarrow \{y \in [x, max] \ : \ |\mathit{density}(y) - \hat{\varphi}[S_0](y)| > \varepsilon\}$
2: **if** $Y \neq \emptyset$ **then**
3: $\quad$ **return** $\inf Y$;
4: **end if**
5: **return** x;
end function

data mining process, including the attacker. Consequently, a malicious peer can use a reconstruction algorithm to infer non-local data.

To exemplify the inside attack in KDEC we have developed *pointhunt*, a simple data reconstruction algorithm for datasets of reals which can be used to reconstruct original data points from a given density estimate.

As input *pointhunt* needs: A starting point x_0 and a collection of already reconstructed data points S_0. Furthermore, the following are needed to run *pointhunt*: a function *density* computing the estimate of the dataset; the kernel function K and its inverse K^{-1}; the window width h; a threshold ε representing the deviation of the current tentative density estimation from *density*; an interval $[min, max] \supseteq S$ such that $\{\mathit{density}(min), \mathit{density}(max)\} \subset [0, \varepsilon]$. Note that this information is known in an inside attack scenario. The algorithm, which must be iterated until a fixed point is reached, works by reconstructing one data point at each iteration, from left to right, as follows. Initially, $x_0 = min$ and S_0 must equal a (possibly empty) set of points that are already known to be in S, e.g. the attackers' local dataset. Function *search* locates the leftmost point x to the right of x_0 where the difference between the actual and reconstructed density is not negligible, i.e. exceeds ε. Given x, a point s_{new}, which is likely to be in S, is calculated by function *guess* using hK^{-1}. This heuristics can be informally justified by noting that x is the leftmost location that is significantly influenced by $S \setminus S_0$ and therefore x is likely to be significantly influenced by only one point in $S \setminus S_0$.

The ideal case for an attacker occurs when K, and consequently the estimate, has bounded support. For example, this is the case for the square or triangular pulse, or Epanechnikov's kernel. Then ε can be set to zero and *search* returns a point of the border of the support of the function $\mathit{density}(x) - \hat{\varphi}[S_0](x)$. Assuming without loss of generality $K^{-1}\colon [0, w_{max}] \rightarrow [0, 1]$, there must be one data point that equals exactly $x + h$, which *guess* returns.

Whereas experiments performed by us with such kernels have led to a full disclosure of the dataset, if the kernel has unbounded support, e.g. the Gaussian kernel, the attacker is less likely to be successful. In general, we can say that the best results are obtained when points are not too close to each other given a value of h. Figure 5 shows the density estimate and the contribution of each kernel.

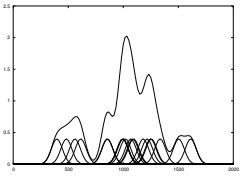

Figure 5. Original points

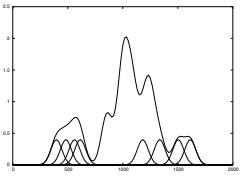

Figure 6. Reconstructed points

In Figure 6 the data points which were correctly reconstructed by *pointhunt* are displayed. Note that the algorithm was unable to find the points in the region where the points are too close to each other. These issues are under investigation.

Single Attack in KDEC. The attacker executes all steps of the KDEC protocol and, after receiving the global density, it tries to reconstruct the original data points. The level of this attack leads to a *full disclosure*. Nonetheless, we observe that the attacker is unable to assign the reconstructed data points to their owner site.

Coalition Attack in KDEC. This attack can be implemented, for example, if all k attacker sites exchange their own local density estimates. In this case, the global density subtracted by the summation of all attackers density will reveal the victims' local density. From the density the attackers can use a reconstruction algorithm and disclose the original data. In this case the attack leads to a *full disclosure*. However, the attackers will be unable to assign the data points to the victims, unless $k = n - 1$.

Probe Attack. The probe attack in KDEC can be achieved if the attacker sets its local density to zero. In this case it will get back a global density function on the others' data. This attack leads to a *full disclosure*, but again the attacker is unable to assign the data points. Note also that the helper could be instructed not to send the sampled GDE to a probing peer.

Malicious Helper Attack in KDEC. The helper in KDEC knows all sampled local densities. It could know the parameters if the malicious helper acted as helper during parameter negotiation. In the latter case, the helper can reconstruct all the data owned by peers in the group. This is a *full disclosure* attack, including assignment of data points to the owning peers. Otherwise the scenario the helper operates in is similar to an outside attack scenario without knowledge of the window width, the kernel, the distance and the sampling parameters, but without the need to steal any of the sampled LDEs.

5.2 Outside attacks in KDEC

Reconstructing data from other peers is relatively easy if the malicious peer is inside the data mining group. From the outside, using only an eavesdropped global density estimate, it is a hard task. In the following we discuss some different outside attack scenarios in KDEC, with respect to information that the attacker does not have.

Extreme Case. The extreme case, where the attacker has stolen all the parameters, degenerates to the inside attack case and the attacker can try to solve the problem with the *pointhunt* algorithm.

Attack without the window width h. The parameter h defines how the kernel function will be stretched in the x-axis, i.e., h determines the range of influence of one point over its neighbors. If h is unknown we cannot compute $\varphi[S']$ and consequently *pointhunt* cannot be used. However, if there is at least one outlier point we can compute h in the following way. Let the set $X^* = \{x^* : \varphi[S](x^*) = K(0)\}$ represent the "*small*" local maxima, which are generated by points

with no close neighbors. X^* can be built in a single pass through the points in $\varphi[S]$ with time complexity $O(n)$ in the number of points in $\varphi[S]$. Let us choose a point x_c close to x^* such that $\varphi[S](x_c) = w < K(0)$. Using the kernel inverse $K^{-1}\colon [0, w_{max}] \to \mathbb{R}_+ \cup \{0\}$, where $w_{max} = K(0)$, we can compute $K^{-1}(\varphi[S](x_c)) = K^{-1}(w) = d_w$ representing the distance from x^* where one point x_c must be placed to receive the influence w from x^*. But x_c lies at $d(x^*, x_c)$ from x^* because it was scaled by h in the computation of $\varphi[S]$. We have that $K(\frac{d(x^*,x_c)}{h}) = w = K(d_w)$ what give us $\frac{d_c}{h} = d_w$. After substitutions we get $\frac{d(x^*,x_c)}{h} = K^{-1}(\varphi[S](x_c))$. Finally, h can be computed with:

$$h = \frac{d(x^*, x_c)}{K^{-1}(\varphi[S](x_c))} \tag{5}$$

Attack without the distance function. The distance function plays a crucial role in the computation of the GDE. Its inverse is very important as well, for our attack algorithm use it to guess the points. Let us assume that the distance is unknown to the attacker. In this case he/she may try to use well known distance functions, e.g., Euclidean distance. To test how good is the distance function chosen, the attacker can use $hK^{-1}(\varphi[S](x_i)) = d(x^*, x_i)$, where x^* is an outlier point, and $x_i, i = 1, ..., n$ are points close to x^*. The success of this approach will depend on how many different candidate functions are chosen and if there is at least one outlier in the GDE.

Attack without the kernel function. Without the kernel function the attacker cannot use the trial-and-error approach used in the inside attack to guess the points. There are two options. The first option is to try to guess the kernel function. The second option is to try to find the points with methods that are independent of knowing the kernel function. Guessing the kernel can be accomplished if the dataset contains outliers. Then the attacker can build one table with the points around an outlier point x^*. This table can be interpolated to get a candidate kernel function $\hat{K}$. This $\hat{K}$ is simple to build, however it may include many approximation errors, which compromise its use in *pointhunt*. Finding points without kernel can be accomplished if the kernel function, or its derivatives, has discontinuities. We can find the points using the observation that the distances between discontinuities on one axis are equal to the distances between data points on the same axis.

Attack without the sampling parameters. KDEC uses a multidimensional sampling technique to transform the density estimates into a sequence of indexed values. These indexes allow the peer sites to transmit information without explicit reference to the original data points. The sampling parameter is $\vec{\tau} \in \mathbb{R}^n$, which is chosen in the initial phase of the protocol. Without $\vec{\tau}$ we cannot reconstruct the data points. However, if h and K are known we can choose two successive values with respect to one axis, $w_{z1}, w_{z2} < K(0)$ such that $hK^{-1}(w_{z1}) = d_1$ and $hK^{-1}(w_{z2}) = d_2$ and attempt to find $\tau = \frac{|d_1 - d_2|}{|z_1 - z_2|}$.

6 CONCLUSIONS AND FUTURE WORK

In this paper we have discussed various types of inference attacks peer-to-peer agent-based networks performing data mining tasks on homogeneous data could be vulnerable to. The potential attack types to particular scheme for homogeneous distributed clustering, known as KDEC, have been investigated, and an algorithm which could reconstruct the data from the kernel density estimates employed by KDEC has been presented. Future work will concentrate on improving the accuracy of the algorithm to expose further possible weaknesses of the KDEC scheme and providing countermeasures to these attacks in KDEC.

REFERENCES

[1] R. Agrawal and R. Srikant, 'Privacy-preserving data mining', in *Proc. of the ACM SIGMOD Conf. on Management of Data*, pp. 439–450. ACM Press, (2000).

[2] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios, 'Disclosure limitation of sensitive rules', in *Proc. of IEEE KDEX'99)*, pp. 45–52, (Nov. 1999).

[3] C. Clifton, 'Using sample size to limit exposure to data mining', *Journal of Computer Security*, **8**(4), 281–307, (2000).

[4] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M.Y. Zhu, 'Tools for privacy preserving data mining', *ACM SIGKDD Explorations*, **4**(2), 28–34, (2002).

[5] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino, 'Hiding association rules by using confidence and support', *LNCS*, **2137**, 369–383, (2001).

[6] Wenliang Du and Zhijun Zhan, 'Building decision tree classifier on private data', in *IEEE ICDM Workshop on Privacy, Security and Data Mining*, volume 14, pp. 1–8, Japan, (2002).

[7] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, 'Privacy preserving mining of association rules', in *Proc. KDD-02*, (2002).

[8] Csilla Farkas and Sushil Jajodia, 'The inference problem: A survey', *ACM SIGKDD Explorations*, **4**(2), 6–11, (2002).

[9] J. R. Higgins, *Sampling Theory in Fourier and Signal Analysis*, Clarendon Press, Oxford, 1996.

[10] T. Johnsten and V. V. Raghavan, 'A methodology for hiding knowledge in databases', in *IEEE ICDM Workshop on Privacy, Security and Data Mining*, volume 14, pp. 9–17, Maebashi City, Japan, (2002). ACS.

[11] Chris Jones, John Hall, and John Hale, 'Secure distributed database mining: Principle of design', in *Advances in Distributed and Parallel Knowledge Discovery*, 277–294, AAAI Press / MIT Press, (2000).

[12] M. Kantarcioglu and C. Clifton, 'Privacy-preserving distributed mining of association rules on horizontally partitioned data', in *DMKD'02*, (June 2002).

[13] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. Random data perturbation techniques and privacy preserving data mining., 2003.

[14] H. Kargupta, I. Hamzaoglu, and B. Stafford, 'Scalable, distributed data mining using an agent-based architecture', in *Proc. KDD-97*, pp. 211–214. AAAI Press, (1997).

[15] H. Kargupta, B. Park, D. Hershberger, and E. Johnson, *Advances in Distributed and Parallel Knowledge Discovery*, chapter 5, Collective Data Mining: A New Perspective Toward Distributed Data Mining, 131–178, AAAI/MIT Press, 2000.

[16] M. Klusch, S. Lodi, and G. Moro, 'Distributed clustering based on sampling local density estimates', in *Proc. IJCAI-03*, Acapulco, Mexico, (2003). AAAI Press.

[17] M. Klusch, S. Lodi, and G. Moro, 'The role of agents in distributed data mining: Issues and benefits', in *Proc. IEEE/WIC IAT2003*, Halifax, Canada, (2003). IEEE Computer Society Press.

[18] Yehuda Lindell and Benny Pinkas, 'Privacy preserving data mining', *LNCS*, **1880**, 36–54, (2000).

[19] S. R. M. Oliveira and O. R. Zaiane, 'Privacy preserving frequent itemset mining', in *IEEE ICDM Workshop on Privacy, Security and Data Mining*, volume 14, pp. 43–54, Maebashi City, Japan, (2002). ACS.

[20] E. Parzen, 'On estimation of a probability density function and mode', *Ann. Math. Statist.*, **33**, 1065–1076, (1962).

[21] Benny Pinkas, 'Cryptographic techniques for privacy-preserving data mining', *ACM SIGKDD Explorations*, **4**(2), 12–19, (2002).

[22] S. J. Rizvi and J. R. Haritsa, 'Maintaining data privacy in association rule mining', in *VLDB 2002*, pp. 682–693, Hong Kong, China, (2002).

[23] Y. Saygin, V. S. Verykios, and C. Clifton, 'Using unknowns to prevent discovery of association rules', *ACM SIGMOD Record*, **30**, 45–54.

[24] Y. Saygin, V. S. Verykios, and A. K. Elmagarmid, 'Privacy preserving association rule mining', in *RIDE 2002*, San Jose, USA, (2002).

[25] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.

[26] S. J. Stolfo, A. L. Prodromidis, S. Tselepis, W. Lee, D. W. Fan, and P. K. Chan, 'JAM: Java agents for meta-learning over distributed databases', in *Proc. of KDD–97*, pp. 74–81, (1997).

[27] J. Vaidya and C. Clifton, 'Privacy preserving association rules mining in vertically partitioned data', in *Proc. KDD-02*, pp. 639–644, (2002).

[28] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining, 2003. Subm. to ACM Trans. on Information and Systems Security.