HHS Public Access

Author manuscript

Int J Coop Inf Syst. Author manuscript; available in PMC 2019 September 01.

Published in final edited form as:

Int J Coop Inf Syst. 2018 September ; 27(3): . doi:10.1142/S0218843018500053.

DIFFERENTIALLY PRIVATE OUTLIER DETECTION IN A COLLABORATIVE ENVIRONMENT

HAFIZ ASIF,
MSIS Department, Rutgers University, 1 Washington Park Newark, NJ, 07102, USA

TANAY TALUKDAR,
MSIS Department, Rutgers University, 1 Washington Park Newark, NJ, 07102, USA

JAIDEEP VAIDYA,
MSIS Department, Rutgers University, 1 Washington Park Newark, NJ, 07102, USA

BASIT SHAFIQ, and
Department of Computer Science, Lahore University of Management Sciences Lahore, Punjab, 54792, Pakistan

NABIL ADAM
MSIS Department, Rutgers University, 1 Washington Park Newark, NJ, 07102, USA

Abstract

Outlier detection is one of the most important data analytics tasks and is used in numerous applications and domains. The goal of outlier detection is to find abnormal entities that are significantly different from the remaining data. Often the underlying data is distributed across different organizations. If outlier detection is done locally, the results obtained are not as accurate as when outlier detection is done collaboratively over the combined data. However, the data cannot be easily integrated into a single database due to privacy and legal concerns. In this paper, we address precisely this problem. We first define privacy in the context of collaborative outlier detection. We then develop a novel method to find outliers from both horizontally partitioned and vertically partitioned categorical data in a privacy-preserving manner. Our method is based on a scalable outlier detection technique that uses attribute value frequencies. We provide an end-to-end privacy guarantee by using the differential privacy model and secure multiparty computation techniques. Experiments on real data show that our proposed technique is both effective and efficient.

Keywords

Outlier Detection; Privacy; Distributed Data

1. Introduction

Today, data is continuously collected across all facets of our life. As such, data regarding a single entity may be collected by many different organizations. For example, in the health care domain, the relevant parts of the electronic health record of a patient can be found at multiple different sites, such as different healthcare facilities, lab sites, pharmacies, etc.

visited by the patient. Similarly, different sites may collect very similar data regarding different entities. As above, in the health care domain, different medical centers and hospitals will typically collect the same kinds of data, but only of the patients visiting them. In either case, the data is clearly distributed across multiple different sites. While this data can be locally analyzed, the results of local analysis may not provide complete insight. Indeed, the potential of big data can only be realized with appropriate analytics, carried out over global data. However, this data cannot be easily integrated into a single database due to privacy and security constraints. Privacy preserving data analysis[34,2] studies the problem of how to carry out data analysis when the underlying data is distributed between multiple organizations. While solutions have been developed for several different analytics tasks, one of the most fundamental tasks, outlier analysis[3], has not as yet received much attention.

Outlier detection is a key data analytics task, and has numerous applications such as fraud detection in finance, terrorist identification in homeland security, and even hotspot identification in climate science. As such, many different outlier detection techniques have been developed in the literature[3]. However, all of these techniques assume that all of the data is centrally available, and do not worry about data privacy. Therefore, they automatically assume that the underlying data being analyzed is the complete global dataset.

As noted above, analytics carried out over local data can often be inaccurate. A simple example (Figure 1) suffices to show this specifically in the case of outlier detection. Figure 1(a) shows two dimensional data collected at two different sites, where each site collects data for the same set of entities, but only collects one dimension of the data. This is also known as vertical data partitioning and is quite commonly found in the real world (for example, in the first electronic healthcare record example discussed earlier). Note that in this case, the outliers cannot be identified from any one dimension, and therefore neither site locally can find the outliers. Figure 1(b) shows two dimensional data collected at three different sites. Each site collects the same two data features, but the entities regarding which the information is collected are different. This is also known as horizontal data partitioning and is quite commonly found in the real world (for example, in the second electronic healthcare record example discussed earlier). Note that in this case FO_1 shows up as an outlier when seen only in the context of Site 2's data, but turns out to not be an outlier when seen in the context of the whole data. Similarly, while the true outliers TO_1 and TO_2 can be locally found as outliers, if we would like to identify the top-k outliers globally, then their prioritization may differ since the degree of outlyingness needs to be computed in the context of the global data. This shows that while outliers can be locally mined, mining of the global data is necessary to ensure an accurate set of results.

While there has been some work that looks at privacy-preserving outlier detection[31], it still does not provide end-to-end privacy in terms of both privacy of the computation as well privacy implicit in the results. Furthermore, these techniques are not very efficient and do not really scale to large scale datasets. In this article, we seek to fill precisely this void. Specifically, we aim to develop a methodology for outlier detection that provides a complete guarantee of privacy both in terms of the results as well as in terms of the computation. Furthermore, the proposed approach should be scalable with respect to the number of records as well as the number of attributes in the dataset, and effective in preserving utility.

To accomplish this, we make use of a base technique for outlier detection that uses attribute value frequencies to estimate the degree of outlyingness. As such, the technique really works only for categorical data, though it can be extended to work for numeric data as well, if proper discretization is used. The underlying assumption of this technique is that an outlier would have rare values for the majority of attributes. While this assumption limits the kinds of outliers that are detected, it does apply in many real life situations.

The key contributions made by this paper are as follows:

- We formalize the notion of differentially private outlier detection.
- We develop an effective solution for both horizontally and vertically partitioned data that is highly scalable that is, the communication and computational complexity is linear with respect to number of data points in the database or the number of attribute values.
- We develop several interesting primitives that may be useful in other privacy-preserving data analytics tasks.

  - MULTIPARTY_BP: An extension of Blind and Permute protocol [26] from two parties to multiple parties.

  - MULTIPARTY_SELECT: A secure protocol that selects k smallest elements securely from the elements of three or more parties.

The rest of the paper is organized as follows: Section 2 presents some of the preliminaries underlying our definitions and approach. Section 3 formalizes the notion of differentially private outlier detection. Section 4 discusses how outliers can be privately mined if the data is centralized. Section 5 develops the privacy-preserving protocol for data that is horizontally partitioned, while Section 6 presents the protocol for vertically partitioned data. Section 7 discusses the complexity of the approach, while Section 8 presents the security analysis. The experimental results on real data are discussed in Section 9. Section 10 overviews the related work while Section 11 concludes the paper and discusses future work.

2. Preliminaries

Before we present the details of our protocols, we give an overview of some primitives and definitions on which we rely for our construction.

Differential Privacy:

Our privacy definition requires output of each party to be differentially private. Differential privacy, which provides privacy through a randomized mechanism, is a de facto privacy measure since its inception. It provides a formal and quantifiable privacy guarantee to the individuals in a database irrespective of the adversary's background knowledge and available computational power.

In differential privacy[9], a randomized algorithm is considered to be differentially private if for any pair of neighboring databases, the probability of generating the same output, is within a small multiple of each other, for the entire output space[9]. Thus, for any two

databases which are close to one another, the output of a differentially private algorithm will approximately be the same. We use the following definition of differential privacy from[9].

Definition 2.1. (Differential Privacy)—A randomized algorithm $\mathscr{M}$ with domain $\mathbb{N}^{|\mathscr{X}|}$ is (ε,δ)-differentially private if $\forall \mathscr{S} \subseteq \text{Range}(\mathscr{M})$ and $\forall x, y \in \mathbb{N}^{|\mathscr{X}|}$ such that $\|x - y\|_1 \leq 1$:

$$\Pr[\mathscr{M}(x) \in \mathscr{S}] \leq \exp(\varepsilon)\Pr[\mathscr{M}(y) \in \mathscr{S}] + \delta$$

Note that in the definition above, ε bounds the increase in privacy risk due to the output divergence for two neighboring databases, whereas δ relaxes the requirement that an observed output is (almost) equally likely to be observed on every neighboring database, simultaneously. Thus, lower values of ε and δ provide higher privacy. In specific, when $\delta = 0$, definition 2.1 implies that a randomized mechanism, $\mathscr{M}$ that is $(\varepsilon,0)$-differentially private, the outcome of $\mathscr{M}$ for neighboring databases x and y that differ by at most one record, can diverge by ε at the maximum. Divergence for two distribution A and B is defined as

$$D_\infty(A\|B) = max_{T \subset \mathscr{Y}} \ln\left(\frac{Pr[A \in T]}{Pr[B \in T]}\right) = max_{y \in \mathscr{Y}} \ln\left(\frac{Pr[A = y]}{Pr[B = y]}\right)$$

We use Laplace Mechanism[9] as the randomized mechanism to achieve differential privacy, because it is well suited for count queries. Laplace Mechanism, $\mathscr{L}$, which provides $(\varepsilon,0)$-differential privacy for a function $f: \mathbb{N}^{|\mathscr{X}|} \rightarrow \mathbb{R}^k$, as defined in[9] is

$$\mathscr{L}(x, f(\,.\,), \varepsilon) = f(x) + \left(Y_1, Y_2, \ldots, Y_k\right)$$

where $Y_i \, \forall i \in [k]$ is independently chosen from Laplace distribution of mean zero and variance $2 \times \left(\frac{\Delta f}{\varepsilon}\right)^2$, represented as $\text{Lap}\left(\frac{\Delta f}{\varepsilon}\right)$. Since we deal with count queries, $f(x) \in \mathbb{N}^k$ instead of $\mathbb{R}^k$. Hence, to each coordinate i of $f(x)$ we add $\left\lfloor Y_i \right\rfloor$ instead of Y_i. Δf here represents the sensitivity of function f; it measures the extent to which a single record can affect the output of f in worst case. The sensitivity for count queries is usually measured through ℓ_1-distance; the definition of ℓ_1-sensitivity follows:

$$\Delta f = \max_{\substack{x, y \in \mathbb{N}^{|\mathscr{X}|} \\ \|x - y\|_1 = 1}} \|f(x) - f(y)\|_1 \quad (2.1)$$

Secure Two/Multi Party Computation—We employ secure multi party computation (SMC) to preserve privacy of data at organizational level, which is also known as computational privacy. Basically, SMC base protocols are employed to ensures that no information other than the specified output and auxiliary information as per protocol specification is reveled to the parties during execution of the protocol computing a function.

The security of an SMC protocol Π is evaluated through simulation based strategy, where we compare the information leakage of the protocol to that the leakage in *ideal model*, where a trusted party is available for computation. Generally speaking, if an adversary in the ideal paradigm can simulate the same view as of the adversary in the real execution of the protocol Π then Π is considered secure.

Ideal Paradigm:

All the parties $P_1,\ldots,P_n$ send their inputs $x^1,\ldots,x^n$ to a fully trusted third party $\mathcal{T}$, who computes function g, and sends back to each party P_l its output g^l. If a party P_l has no output then $\mathcal{T}$ sends $g^l = \perp$ to P_l. Let $\mathcal{S}_l$ be a probabilistic polynomial time (PPT) adversary controlling a party P_l and having access to auxiliary its input γ^l and the output of P_l then we represent it view as $\text{IDEAL}_{g,\mathcal{S}_l}\left(x^1, \ldots, x^n, k\right)$.

Real Paradigm:

All parties $P_1,\ldots,P_n$ execute the protocol Π to compute g with their respective inputs x^1, $\ldots,x^n$. At the end of the protocol each party P_l gets g^l, its output. If $\mathcal{A}_l$ is a PPT adversary controlling P_l, and has access to the auxiliary input γ^l and output of P_l then we represent its view in real execution as $\text{REAL}_{\Pi,\mathcal{A}_l}\left(x^1, \ldots x^n, k\right)$.

Definition 2.2. (Security)—Let g and Π be as defined above. $\forall l \in [n]$ for every $x^l \in \mathbb{N}^{|\mathcal{X}|}$, Π securely computes g in the presence of static semi-honest adversary only if for all $l \in [n]$ every probabilistic polynomial (PPT) adversary $\mathcal{A}_l$ in real paradigm there exists a PPT adversary $\mathcal{S}_l$ in ideal paradigm such that

$$\text{REAL}_{\Pi,\mathcal{A}_l}\left(x^1,\ldots,x^n,k\right) \overset{c}{\equiv} \text{IDEAL}_{g,\mathcal{S}_l}\left(x^1,\ldots,x^n,k\right)$$

Garbled Circuit:

Garbled circuit, as proposed by Andrew Yao[37], is used to compute any function f: {0,1}* → {0,1}* securely in the presence of semi-honest adversary. In general, for large input sizes garbled circuit has a comparatively large overhead; therefore, we use customized protocols to solve such problems. However, garbled circuits are still used in our protocol as a sub-routine for comparison purposes, since they are quite efficient for this task.

Additive Homomorphic Encryption:

Additive homomorphic encryption (AHE) such as Paillier[25] is a very useful cryptographic primitive. It is a public key encryption scheme that allows for addition of two messages in encrypted form. pk and sk are public and private keys of AHE for security parameter s, while $P(s)$ is the corresponding plain-text domain. Encryption and decryption procedure for $m \in P(s)$ are represented as $c = E_{pk}[m]$ and $D_{sk}[c] = m$ respectively. Because of the additive homomorphism $E_{pk}[m_1] \times E_{pk}[m_2] = E_{pk}[m_1 + m_2]$ for $m_1, m_2 \in P(s)$. It simply follows that

any $a \in P(s)$ can be multiplied with $E_{pk}[m]$ since it is just repeated addition i.e., $E_{pk}[m]^a = E_{pk}[a \times m]$.

Secure Sum:

This protocol computes the sum $v = \sum_{l \in [n]} v_l$, where party P_l – which is static and semi-honest in its adversarial behavior – has the value v_l, without revealing individual values of the parties[8]. The idea here is quite simple. First, all parties agree on a security parameter $b \in \mathbb{N}$ such that $v \in [0,2^b]$ for all values of v_l, $\forall l \in [n]$. Second, one party, let us say P_1, picks r uniformly from $[0,2^b]$, we would represent this as $r \leftarrow [0,2^b]$, and set $v = r$. Third, each party P_l, starting from P_1, sets $v = (v + v_l) \bmod (2^b+1)$ and sends v to P_{l+1} until v reaches P_n; in the same fashion as above, P_n also updates v based on its input and sends the updated v to P_1. Lastly, P_1 computes $(v + (-r)) \bmod (2^b + 1)$, where $-r = 2^b + 1 - r$ is the additive inverse of r; this gives the desired sum. To improve legibility, we omit the mod in the following text, however, the actual computation is always done by using modular arithmetic.

3. Defining Private Outlier Detection

In this paper we solve the problem of outlier detection in privacy preserving manner, where the data is horizontally or vertically partitioned among participating parties. Horizontal partitioning of data, as per description in work by Vaidya et al.[34], implies that various parties collect same information on different entities. Thus, each record (or row) in the database which contains the complete information for a single individual belongs to only one party. In case of vertical partition, parties collect different information (features/attributes) on same set of individuals. Horizontal and vertical partitions are the two standard models for data partition. We focus on outlier detection for categorical data, where usually distance based outlier detection techniques fail to work. At a high level, we can informally define the problem at hand as follows:

Definition 3.1. (Problem Statement)

A database x is horizontally or vertically partitioned as $x^1,\ldots,x^n$ among multiple parties, $P_1,P_2,\ldots,P_n$. The parties want to find the outliers in global database x, while preserving privacy of their data.

We now formalize the model assumptions and the notion of privacy.

Model Assumptions: We assume the existence of secure communication channels among the parties. Parties are modeled as semi-honest adversaries – such parties will honestly follow the protocol, but will try to obtain extra information (not allowed by the privacy definition) based on their intermediate messages. Parties involved in the protocols are also assumed to be non-colluding in nature, meaning they will not share any information which they are not explicitly instructed to share with other parties. Additionally, adversarial behavior of the parties is static, that is, the behavior of each party will be fixed before the protocol commences. These are standard assumptions in the literature and fit many real life situations.

Considering the above assumptions, we can informally define privacy as follows:

Definition 3.2. (Privacy)

A protocol Π in presence of *static semi-honest* adversaries computes outliers in privacy preserving manner as long as:

1. no party obtains any information except for its output, as per the protocol specification, with probability greater than negligible function, which is smaller than any inverse polynomial in security parameter.
2. the output each party receives is differentially private.

There are many algorithms[19,13,15,14,24] to perform outlier detection for categorical or mixed attributes in the data. However, we focus on an attribute value frequency (AVF) based approach[19], since it has several advantageous features such as being highly efficient and scalable, while providing comparable accuracy to other state of the art algorithms. Furthermore, AVF based algorithm can be naturally implemented in a distributed fashion, a desirable trait in our problem setting; thus it gives good performance even when the data is distributed among different parties. These qualities make AVF based algorithm suitable for computing outliers in our distributed setting, where privacy of individuals and parties is to be preserved without significantly impacting efficiency.

The AVF based outlier detection is based on the premise that any point/record which is uncommon in terms of all of its attribute values should be considered an outlier. In other words, an entity is likely to be an outlier if it has infrequent attribute values for all of its attributes. This idea is central to many outlier detection technique such as density based outlier detection[11], where areas of different relative density are considered outliers.

Koufako et al., in[19], measure the rareness of a particular value of an attribute by computing the frequency of this attribute value i.e., the number of times this value occurs in the database. We refer to this value as the count frequency. Based on count frequency for each of the attribute value of a record, the attribute value frequency score (AVFS) is computed, which is employed by the algorithm in[19] to find outliers in the data.

We now formalize the above mentioned notions for AVF based outlier detection. Let each record in the database belong to $\mathcal{X} = \prod_{j \in [M]} A_j$, where $A_j = \{v_{j,1}, v_{j,2}, \ldots, v_{j,M_j}\}$ is the set of possible values j^{th} attribute can take and $M_j, M \in \mathbb{N}$. The database x is represented as a histogram, that is $x \in \mathbb{N}^{|\mathcal{X}|}$ and $||x||_1 = N$, where each x_i represents the number of records of type $i \in \mathcal{X}$ in x. If we let $f_{j,l}(x)$ to be the count frequency for the attribute $v_{j,l}$ (for $j \in [M]$ and $l \in [M_j]$) in database x then AVFS score of any record r in database x that has a type $i \in \mathcal{X}$, can be defined as follows:

$$\text{AVFS}(i, f(x)) = \sum_{j=1}^{M} f_{j,i_j}(x)$$

here (j, i_j) corresponds to the value, v_{j,i_j}, of j^{th} attribute in type i (or record r) for $i_j \in [M_j]$.

For a database, AVF score measures the rareness of a record in terms of its attribute values; thus, the smaller AVF score is for a record, the more likely it is for the record to be an outlier. Now, the AVF based algorithm, as outlined in[19], is simple: the AVF score for every record is calculated and the k records with the smallest AVF scores are considered as potential outliers. Here k is a parameter that can be set by the user to retrieve the top-k outliers. We can now give the definition for AVF based outliers.

Definition 3.3. (AVF based Outliers)

For a database $x \in \mathbb{N}^{|\mathcal{X}|}$, and $k \in \mathbb{N}$ such that $k \leq \|x\|_1$, a set $S \subseteq x$, such that $|S| = k$, is the set of outliers if and only if for each record in S, its attribute value frequency score is not greater than any record not in S.

We now look at how differential privacy can be applied in mining AVF based outliers. Let $f : \mathbb{N}^{|\mathcal{X}|} \rightarrow \mathbb{N}^m$, where $m = \sum_{j \in [M]} |A_j|$. For $x \in \mathbb{N}^{|\mathcal{X}|}$,

$$f(x) = \left(f_{1,1}(x), \ldots, f_{1,M_1}(x), \ldots, f(x)_{M,1}, \ldots, f(x)_{M,M_M} \right) \quad (3.2)$$

where $f(x)_{j,l}$ is the count frequency of the value, $v_{j,l}$, of j^{th} attribute. Hence, f essentially returns an m-dimensional vector containing individual count frequencies for all the values of all the attributes in x.

If we make the output of f differentially private through ε-differentially private laplace mechanism, $\mathcal{L}$, then computing outliers can be considered as post-processing; hence the output would provide the same level of privacy. The first requirement in designing $\mathcal{L}$ is to establish the sensitivity of f, given by equation-2.1. We claim that ℓ_1 sensitivity for f (which is defined in Equation 3.2) is M i.e. $\Delta f = M$, proof for the claim follows.

Proof. For two neighboring databases $x, y \in \mathbb{N}^{|\mathcal{X}|}$, where $\mathcal{X} = \prod_{j=1}^{M} A_j$, $\|x - y\|_1 \leq 1$, and f, as defined in equation-3.2, ℓ_1-distance between $f(x)$ and $f(y)$ can be given as

$$\|f(x) - f(y)\|_1 = \sum_{j=1}^{M} \sum_{l=1}^{M_j} |f_{j,l}(x) - f_{j,l}(y)| \quad (3.3)$$

Since $\|x - y\|_1 \leq 1$, there is at most one additional record of one particular type in either x or y compared to the other, that is, there exists some type $\hat{i} \in \mathcal{X}$ such that $\left| x_{\hat{i}} - y_{\hat{i}} \right| = 1$ (i.e., there is exactly one extra record of that type). If we let $\hat{i} = \left(v_{1,\hat{i}_1}, v_{2,\hat{i}_2}, \ldots, v_{M,\hat{i}_M} \right)$, where $v_{j,\hat{i}_j}$ is the value of j^{th} attribute in type $\hat{i}$. It should be clear that the counts of these attribute values increase by one due to this additional record. Specifically, $\forall j \in [M]$ and $i \in \mathcal{X}$ if

$\left(j, i_j\right) \in \mathcal{I} = \left\{\left(1, \hat{i}_1\right), \left(2, \hat{i}_2\right), \ldots, \left(M, \hat{i}_M\right)\right\}$, $\left|f_{j,i_j}(x) - f_{j,i_j}(y)\right| = 1$, while $\left|f_{j,i_j}(x) - f_{j,i_j}(y)\right| = 0$ otherwise.

Let I be the indicator function defined as follows:

$$I\left(j, i_j\right) = \begin{cases} 1 \text{ if } \left(j, i_j\right) \in \mathcal{I} \\ 0 \text{ otherwise} \end{cases}$$

Thus, we get $I\left(j, i_j\right) = \left|f_{j,i_j}(x) - f_{j,i_j}(y)\right|$. Now, using I, ℓ_1-distance of f for any two neighboring databases can be rewritten as

$$\underset{\|x-y\|_1 = 1}{\forall x, y \in \mathbb{N}^{|\mathcal{X}|}}, \|f(x) - f(y)\|_1 = \sum_{i \in \mathcal{X}} \sum_{j=1}^{M} I\left(j, i_j\right) = \sum_{j=1}^{M} \sum_{i \in \mathcal{X}} I\left(j, i_j\right) = \sum_{\left(j, i_j\right) \in \mathcal{I}} 1 \Rightarrow \max_{\substack{x, y \in \mathbb{N}^{|\mathcal{X}|} \\ \|x-y\|_1 = 1}}$$

$$\|f(x) - f(y)\|_1 = \max(|\mathcal{I}|)\Delta f = M$$

Since the sensitivity of the frequency counts is M i.e., $\Delta f = M$, we can design $\mathcal{L}$ which will preserve ε-differential privacy by adding noise from $\text{Lap}\left(\frac{M}{\varepsilon}\right)$ to the output of $f(x)$. Once the differentially private frequencies $f(x)$ are available, it is simple to calculate the AVF score for any record of type $i \in \mathcal{X}$ in the database x; this computational step can be considered as post-processing for which differential privacy also provides provable guarantees in term of privacy. If post-processing is defined as $F: R \rightarrow R'$ for a randomized mechanism $\mathcal{M}: \mathbb{N}^{|\mathcal{X}|} \rightarrow R$, which is (ε, δ)-differentially private, then $F \circ \mathcal{M}$ is also (ε, δ)-differentially private[9]. This indicates that if one has differentially private counts frequencies then they can be used to compute differentially private attribute value frequency score that is,

$$\Pr[AVFS(\mathcal{L}(x, f, \varepsilon))] \leq \exp(\varepsilon)\Pr[AVFS(\mathcal{L}(y, f, \varepsilon))]$$

If we compute AVF based outlier using differentially private count frequencies, we would provide differential privacy for individuals in the database.

4. Privacy Preserving AVF based Outlier Detection in Ideal Paradigm

If the parties have access to a fully trusted third party $\mathcal{T}$ then they can use Algorithm 4.1 to find outliers in a privacy preserving and secure fashion. This situation corresponds to the *ideal paradigm*.

In this setting all parties agree on f (function for computing count frequencies), AVF_OLs (function for computing AVF based outliers), k (total number of outliers to be found in

global database). Each party P_l has inputs x^l, the databases fragment, and the auxiliary input γ^l, which provides the following information: n (total number of parties), A_j, $\forall j \in [M]$ (possible values for each attribute), $\mathcal{X}$ (space of all possible types or records), and whatever information it already has on the parties input.

Algorithm 4.1 gives one way to compute the k AVF based outliers in a differentially private manner. All parties send their database fragment to $\mathcal{T}$ who computes differentially private count frequencies and k outliers with respect to the global database $x = \sum_{l \in [n]} x^l$, and sends to each party P_l its output AVF_OLs^l (i.e., P_l's outliers in k AVF based outliers) and differentially private $f(x)$, that is, all the count frequencies.

The centralized algorithm enables us to solve the problem described in definition 3.1 which provides security/privacy as per privacy definition 3.2. This simple setting helps us analyze security/privacy and elaborate some of the design choices for the protocols that we make. Let us first define is privacy in an ideal paradigm, when performing outlier detection using AVF based methodology.

Definition 4.1. (Private Outlier Detection in Ideal Paradigm)

Let ε-f,x^l, and AVF_OLs^l be as defined above. We say the parties $P_1,\ldots,P_n$, which are static and semi-honest, find AVF based outliers securely (or in a privacy preserving manner) in ideal paradigm with help of a fully trusted third party only if no party P_l receives any information beyond its input – x^l and γ^l -, output - ε-f and AVF_OLs^l – and whatever can be inferred based on its input and output with probability that is negligible in input size.

A function μ in negligible in $s \in \mathbb{N}$ if $\mu(s) < \frac{1}{\text{poly}(s)}$, where poly($s$) is any polynomial in s.

Note that the above privacy definition does not imply that nothing is learned since if a party knew something other than what we explicitly specified then it can be made part of auxiliary input γ. Rather, it bounds the information learned through the process of computing outliers by employing a fully trusted third party. As such, it is the most appropriate and necessitates that no party gains any information on exactly what records other parties have or any notion of which (and how many) outliers are owned by any other party, etc. except for the information that can be inferred solely based on the party's input, output and received differentially private count frequencies.

The reason we provide each party with differentially private count frequencies ε-f along with its output (outliers) that is calculated based on ε-f, is not by accident, but by choice; it serves two purposes here. Firstly, if we only release differentially private outliers then the amount of noise that we will have to add in order to achieve required privacy level will be huge for any reasonable privacy parameter ε. This is due to the fact that in the worst case, the sensitivity of such a function is k for two neighboring databases, where we will have totally different outliers for two neighbors; as opposed to this, if we release differentially private count frequencies then the effect of the added noise to count frequencies to achieve differential privacy in outlier detection is not as much. Secondly, compared to releasing only the outliers, when we release count frequencies along with outliers, we reveal more information, but this helps gain a better accuracy in term of outlier detection, while at the

same time it does not constitute a huge information leakage in many cases. Thirdly, this choice reduces the number of secure/cryptographic operations needed to find outliers in the privacy preserving SMC protocol.

We now give an overview of the Algorithm 4.1. $\mathcal{T}$, firstly, constructs the global database x, which it uses to compute count frequencies – lines 1 and 2. In line, 4 $\mathcal{T}$ samples a vector Y of length m (the size of f) from laplace distribution of mean zero and scale $\frac{M}{\varepsilon}$ such that for all $t \in [m]$, Y_t is iid, which is then added to the f in line 5 to obtain pf, ε-differentially private frequencies. Now, for each record r in x, $\mathcal{T}$ computes AVFS for r and adds it to a vector. $\mathcal{T}$ then picks k records corresponding to k smallest AVFS and sends outliers for each party along with pf.

5. Private Outlier Detection in Horizontal Fragmentation Case

When the database is partitioned between multiple parties, which collaborate using a protocol Π to find the outliers, the protocol can be considered to preserve privacy only if it fulfills the following privacy definition:

Algorithm 4.1

POLD_Central

Input: $\forall l \in [n]$, database x^l from P_l

Input: ε, k and f as per agreement of all the parites

Output: Each party P_l receives its outliers and $\varepsilon -$ differentially private count frequencies

1: $x = \sum_{l \in [n]} x^l$

2: Set pf $= f(x)\{$computes count frequecies$\}$.

3: Set m to be the size of vector pf.

4: Sample Y from laplace distribution: $Y \overset{iid}{\sim} Lap\left(\frac{M}{\varepsilon}\right)^m$

5: Set pf $=$ pf $+$ Y

6: *Set* $N = \left\|x\right\|_1 \left\{\text{i.e., the number of records in } x\right\}$.

7: Initialize AVFS, array of size N, to 0, and $t = 0$.

8: **for** each record r in x **do**

9: **for** each $j \in [M]$ **do**

10: $\text{AVFS}_t \leftarrow \text{AVFS}_t + \text{pf}_{j, r_j} \left\{\text{pf}_{j, i_j} \text{ is differentially private count frequency of } v_{j, i_j}\right\}$.

11: **end for**

12: Set $t = t + 1$

13: **end for**

14: Insert k records in O that corresponds to the smallest values in AVFS; in case we have more than k such records, we pick $k$records randomly.

15: **for** each record r in O **do**

16: Send r to all the parties that have r along with pf .

17: **end for**

Definition 5.1. (Privacy for Horizontally Partitioned Data)

A protocol Π that computes AVF based outliers will preserve privacy if every party learns only the $(\varepsilon,0)$-differentially private count frequencies and for its records, which (if any) should be considered outliers along with any other information that can be derived only from this explicitly disclosed information.

In this section we propose POLD_HoF, a differentially private protocol for outlier detection for horizontally partitioned database. We suppose that the database x is fragmented horizontally into $x^1,x^2,\ldots, x^n$ among parties $P_1,P_2,\ldots,P_n$ such that x^i is owned by the party P_i and $x = \sum_{i\in[n]} x^i$. These parties are interested in determining k (global) AVF based outliers in x, while preserving privacy of their databases and the individuals in them. Here an underlying assumption, which is also quite reasonable, is that all parties know all the possible values for all attributes in database. This follows from the fact that all parties collect information on the same attributes on different individuals in their respective local databases. However, even in the case where this assumption does not hold, each party can add its local count frequencies, of the attribute values it has, to compute the final count frequencies. This will not affect the correctness of the proposed protocol and will not affect the privacy of the protocol as the privacy definition (Definition 5.2) takes this into account. Alternatively, we can say all parties are aware of the all types $i \in \mathcal{X} = \prod_{j=1}^{n} \mathcal{A}_j$, but no party P_i wants to reveal types $T \subseteq \mathcal{X}$ in its database fragment x^i to other parties. Thus, for each party P_i, the auxiliary information, γ^i is the same as in ideal paradigm. The parties are also concerned about the privacy of individuals in their respective databases; hence they need to employ differential privacy to achieve this goal. The privacy definition of the protocol Π_h that securely computes outliers in horizontal fragmentation case is as follows:

Definition 5.2. (Protocol Privacy)

Let AVFS_OLs, f be as defined above. A protocol Π_h securely computes AVF based outliers securely for horizontally distributed database among parties such that each party P_i has input x^i and auxiliary input γ^i only if for every probabilistic polynomial adversary $\mathcal{A}_i$ in real model controlling P_i there exits PPT adversary $\mathcal{S}_i$ that controls P_i in ideal paradigm such that

$$\text{REAL}_{\Pi_h,\mathcal{A}_i}\left(x^1,\ldots,x^n,k\right) \overset{c}{\equiv} \text{IDEAL}_{\text{AVFS\_OLs},\mathcal{S}_i}\left(x^1,\ldots,x^n,k\right)$$

The approach taken in this case is the same as that of Algorithm 4.1; however, with the caveat that differentially private frequencies and outliers need to be computed in a collaborative fashion. Figure 2 gives an overview of the protocol Π_h. First, the parties collaboratively find differentially private count frequencies for each attribute value. Secondly, each party uses differentially private count frequencies to find AVFS for each of its records. Thirdly, each of the party finds its k local outliers i.e., the k records with the smallest values of AVFS, which gives the set of candidates for the global outliers, because every global outlier must also be a local outlier for some party. However, since no information regarding non-outliers should be revealed to other parties, all parties together find the k global outliers from this candidate set in a secure and privacy preserving manner.

In order to calculate AVFS for database x, parties need to know count frequencies $f(x)$, which can be calculated by summing up count frequencies $f(x^i)$ from all the horizontal fragments of x i.e. $f(x) = \sum_{i=1}^{n} f\left(x^i\right)$. Thus, once the local count frequencies have been computed, the parties can simply use the secure sum protocol to calculate the actual count frequency of each value. In the basic secure sum protocol, a random number is used to disguise the values before summing and then removed from the sum to give the correct result. Although use of secure sum will not reveal the individual count frequencies of the parties, it will not protect the privacy of individuals as per the privacy definition 5.1 since parties would know the actual count frequencies. Therefore, to make count frequencies ε-differentially private one of the parties can be instructed to apply $\mathscr{L}$, discrete laplace mechanism, to its local count frequencies during the execution of secure sum (as shown in expression 5.4); for example, the party P_1 can add the laplacian noise (from $\text{Lap}\left(\frac{M}{\varepsilon}\right)$) to its local count frequencies, while computing secure sum for each of the local count frequency.

$$f(x) = \sum_{i=1}^{n} f\left(x^i\right) + \text{Lap}\left(\frac{M}{\varepsilon}\right)^m \quad (5.4)$$

This procedure will make count frequencies ε-differentially private for all the parties except for the party who actually added the noise because it can subtract the noise to obtain actual count frequency, which will again lead to compromise of privacy. One simple way to avoid this attack is to make more than one party add appropriate laplacian noise. Thus to avoid this attack will need at least two parties each adding noise from $\text{Lap}\left(\frac{M}{\varepsilon}\right)$. If now either of the two parties removes their noise, the count frequencies will still be at least ε-differentially private. Note that in our case, addition of noise by two parties will be sufficient for providing privacy guarantee as per definition 5.1 because we assumed parties to be non-colluding, which is a standard assumption in literature, though this can be easily relaxed as well. However, the main problem with this approach is degradation in utility (accuracy) since the amount of added noise is essentially doubled. In order to solve this problem, without affecting the utility beyond what is necessary, we employ a semi-honest and non-colluding party T – e.g., it can be a server in the cloud- that will receive the secure sum without the random number, add noise from Laplace distribution to the secure sum, and send the noisy secure sum to the first party.

Once all the parties have calculated differentially private AVF scores, the next step is to identify k-smallest AVF scores and inform each party of their records that are outliers. In order to achieve this, the parties should identify their local outliers, that is, the records with k-smallest AVFS in each of their horizontal database fragment, x^i. Next we need to find the global outliers, which will be the records with k-smallest AVFS in x. The global outliers, of course, will be among the local outliers of the parties.

5.1. Two Party Protocols for Secure Retrieval of k-smallest Items

To achieve secure retrieval of of outliers, in case of two parties, we use the following protocols to find the items with k-smallest values, which are described in[26].

Blind and Permute Protocol (BP from [26]): Given a vector $S = (s_1,\ldots,s_u)$, which is kept by a party P_1, it creates additive random share $S' = \left(s'_1, \ldots, s'_u\right)$ kept by P_1 and $S'' = \left(s''_1, \ldots, s''_u\right)$ kept by P_2 such that for each s_l in S, $s_l = s'_l + s''_l$ and P_2 does not know the original value s_l.

Secure Select Protocol (SELECT from[26]): Given S in additive split form (as described above) and k, select k smallest values in S, which are also in additive split form. We make a slight change in the original protocol described in[26]: instead of using garbled circuit to compute XOR, we use a comparator garbled circuit, where the secure comparison ($a > b$) for $a = a_1 + a_2$ and $b = b_1 + b_2$ is evaluated by comparing $a_1 - b_1 > b_2 - a_2$.

It is important to note that for SELECT (from[26]) to successfully terminate the values in S should be unique, but for us this assumption will not be satisfied in most of the cases. We solve this problem by selecting $\alpha > |S|$ and multiplying it with each of s_l and adding $l \in \{1, \ldots,|S|\}$ to the product i.e., for each s_l, we compute $\alpha s_l + l$. This will achieve uniqueness without disturbing the original order of AVFS.

If there are only *two parties involved* then each party firstly uses BP to create random shares of AVFS of its local outliers and then uses SELECT to get global outliers. This will not reveal any privacy compromising information since each party gets knowledge of its outliers. Of course it will also enable each party to learn the number of outliers in the database fragment of the other party because both the parties know k. This information leakage cannot be avoided even in *ideal paradigm*.

5.2. Multi-Party Protocols for Secure Retrieval of k-smallest Items

We now consider the case where there are more than two parties that need to find k items with smallest values (or k AVF based outliers). In this case the above described scheme will not work because running SELECT iteratively between the parties would potentially leak the number of items/records of one or more parties included in the k-smallest items to other parties. For example, if there are three parties P_1, P_2, and P_3, even though all parties know k, if the standard SELECT protocol is used, each party will not only learn how many of its outliers are in the global output, but also how many of each of the other parties' local outliers are present in the global output. To circumvent this and many other problems that such a naive strategy presents, we propose a multiparty blind and permute protocol and a multiparty select protocol.

Multiparty Blind & Permute Protocol (MULTIPARTY BP): Similar to BP, the aim of this protocol is also to create random shares of vector S, where $S = (S_1,\ldots,S_n)$ with $S_l = \left(s_1^l, \ldots, s_k^l\right)$ that is kept by the party $P_l \forall l \in \{1,\ldots,n\}$ with $n \geq 3$, and distribute them between two parties in such a fashion that no party knows either the actual value or the

ownership i.e., which value belongs to whom. Algorithm 5.1 presents the detail for this protocol.

Algorithm 5.1

MULTIPARTY_BP

Input: Vectors of values and their IDs, $S_l = \left(s_1^l, \ldots, s_k^l\right)$ and $I_l = \left(i_1^l, \ldots, i_k^l\right)$ respectively, by each party P_i

Output: P_1 and P_2 each gets blinded and permuted random shares of $S = (S_1, \ldots, S_n)$ and $I = (I_1, \ldots, I_n)$

1: P_1 generates public − private key pair (pk, sk) for AHE and sends it to all the parties

2: **for** each party P_l **do**

3: generate sk_l {secret key for private key encryption}

4: send$E_{pk}\left[S_l\right]$ and $E_{sk_l}\left[I_l\right]$ to P_n

5: **end for**

6: $\left\{P_n \text{ performs following steps:}\right\}$

7: Sets $E[S] = \left(E_{pk}\left[S_1\right], \ldots, E_{pk}\left[S_n\right]\right)$

8: Sets $E[I] = \left(E_{sk_1}\left[I_1\right], \ldots, E_{sk_n}\left[I_n\right]\right)$

9: Randomly and additively splits S into S' and S''

10: Randomly and additively splits $E[I]$ into $E[I]'$ and $E[I]''$

11: Picks a random permutation, π

12: Sends $\pi(E[S'])$ and $\pi(E[I]')$ to P_1

13: Sends $\pi(S'')$ and $\pi(E[I]'')$ to P_2

We wish to split all the values in S, coming from n parties, between the two parties (P_1 and P_2). In order to get all values in one place without compromising security we use additive homomorphic encryption (AHE). P_1 generates public-private key pair (pk,sk) of AHE and sends the public key, pk, to all the parties – the reason we choose AHE is to allow any party with access to public key pk to randomly and additively split an encrypted value $E_{pk}\left[s_j^l\right]$. Each party P_l is assumed to have a vector $S_l = \left(s_1^l, \ldots, s_k^l\right)$ that contains values to be blinded, permuted and distributed between P_1 and P_2. P_l also has a vector $I_l = \left(i_1^l, \ldots, i_k^l\right)$ of IDs for the values in S_l such that i_j^l is the ID of value s_j^l for $j \in \{1,\ldots,k\}$. All the values and IDs by all the parties are to be sent to P_n so that it can create S and I, perform a random permutation on them and create random shares of both S and I. Random shares of S and I will be distributed between P_1 (who will get encrypted random shares) and P_2 (who will get random shares in plain-text). Once P_n receives all the encrypted values (i.e. $E_{pk}[S_l]\ \forall l \in \{1,\ldots,n\}$) and their IDs I_l from all the parties, it puts them together as $E[S] = \left(E_{pk}\left[s_1^1\right], \ldots, E_{pk}\left[s_k^n\right]\right)$ and $I = (I_1, \ldots, I_n)$. Since P_n has access to $E[S]$ and pk it can create random additive splits $E[S']$ and S''

of $E[S]$, where $E[S'] = \left(E_{pk}\left[s_1^1 - r_1\right], \ldots, E_{pk}\left[s_k^n - r_{kn}\right]\right), S'' = \left(r_1, \ldots, r_{k \times n}\right)$ and $r_j \forall j \in \{1, \ldots, kn\}$ is a uniformly and independently picked random number. P_n creates random shares I' and I'' of I in a similar manner as well. Next, P_n picks a random permutation π and sends $\pi(E[S'])$ and $\pi(I')$ to P_1, and $\pi(S'')$ and $\pi(I'')$ to P_2. The use of same random permutation π for both the vectors is essential to preserve the correspondence between values of S and their IDs in I. After receiving $E[S']$, P_1 decrypts it to obtain S'.

Algorithm 5.2

MULTIPARTY_SELECT

Input: k; Random shares of elements (in S) and IDs (in I) with P_1 and P_2.

Output: Each party gets informed of its elements in S which are among the k smallest elements.

1: P_1 and P_2 follow the protocol SELECT to find k smallest elements in S.

2: P_2 sends its random share of encrypted IDs corresponding to k smallest elements to P_1.

3: P_1 obtains IDs, I^k, of k smallest elements by adding corresponding random shares of IDs.

4: **for** each $l \in [n]$ **do**

5: **if** I^k contains P_l's IDs **then**

6: Identify the elements in S corresponding to P_l's IDs.

7: Replace all $Pl's$ IDs in I^k with a randomly picked ID not in I^k.

8: **end if**

9: Send I^k *to* P_{l+1} $\left\{P_{n+1} \text{ is null}\right\}$

10: **end for**

Secure Multiparty Select Protocol (MULTIPARTY_SELECT): We now describe the secure protocol for selecting k smallest elements from the set S, which has been blinded and permuted by MULTIPARTY_BP i.e., two static semi-honest parties have additive random shares of S and the IDs, I, of elements in S such that one party has S' and I', while the other party has S'' and I'', where $S = S' + S''$ and $I = I' + I''$. The elements in S belong to multiple parties ($n \geq 3$), but each element in S only belongs to one of the participating parties. The protocol assumes that no two parties have the same value for their IDs and domain for IDs is "large" (we will elaborate the need for such a requirement in description of the protocol below); however, fulfilling these requirements is quite easy.

Here the main security requirement, which was not being fulfilled by the previous protocol, is to find k smallest elements from blinded and permuted S in such a manner that for each party P_l, the protocol reveals nothing but the IDs of elements that belong to P_l and are among the k smallest elements. Additionally, if a party does not have any elements in the set of k smallest elements then it should not get any information except that none of its elements in S are included in the k smallest elements.

There are two main problems to be solved here (i) identify k smallest elements, (ii) inform each party only of its outliers, while making sure that no party gains any extra information. This is done as follows. P_1 and P_2 together can execute SELECT and identify random shares corresponding to the k smallest elements. After this P_2 sends the random shares of IDs corresponding to k smallest elements to P_1. P_1 can easily obtain the actual IDs, I^k, of k smallest elements by adding corresponding random shares. Now for each ID in I^k that is same as that of the ID of P_1's element in S, P_1 replaces this ID in I^k with a randomly picked element from ID's domain. This is done in order to stop other parties from learning the number of IDs belonging to P_1. P_1 then sends this updated I^k to the next party (P_2), where it also follows the same steps followed by its predecessor. Note that when picking random numbers to add to I^k, each party needs to make sure that the random number picked should not be equal to any of the IDs already existing in I^k. Eventually, the updated I^k reaches P_n, which can finds its IDs belong to the k smallest elements, and conclude the protocol.

Note that we need IDs to be unique so that no party mistakes someone else's IDs as their own, whereas requirement for large domain for IDs is to make sure that chosen random number are not same as one of the IDs in I, but not in I^k.

5.3. Secure Protocol for AVF based Outlier Detection

Now that we have the sub-protocols needed for our construction, we can finally specify the complete protocol for private outlier detection in case of horizontal fragmentation (POLD HoF) to compute the k attribute frequency based outliers. First, parties collaboratively calculate differentially private count frequencies, using which each party P_l computes AVFS for all of its records in x^l. Second, each party P_l finds k local outliers, the records with k smallest AVFS in x^l and stores these AVFS in S_l. To generate ID for each value in S_l, P_l samples a random number uniformly from $[u]$ for $u \in \mathbb{N}$ and stores it in I_l. We choose u to be large enough to make collision probability (the probability for two randomly picked IDs to be the same) is negligible in kn.

The next step is to make all locally smallest AVFS and their IDs unique because the protocol, MULTIPARTY_SELECT, which is used for selecting globally k smallest AVFS securely, works correctly only if the values are unique. To achieve uniqueness property each party P_l makes its AVFS (s^l_j) in S_l and IDs unique using similar procedure as described earlier except for one change that now P_l will add $(j-1)\times n+l$ to the scaled s^l_j or ID instead of j i.e., $\alpha s^l_j + (j-1)\times n + l$ with $\alpha > n$, where n is total number of parties (we assume the values in S_l are in ascending order). The rationale behind using this specific perturbation and order is to make sure that if an outlier record is kept by more than one party, where at least one party has more than one such row, then all parties owning this outlier should get it as long as the number of such parties are less than targeted number of outliers that is k. For example, consider the following scenario, where $k = n = 3$ and a type $i \in \mathcal{X}$ such

Algorithm 5.3

POLD_HoF

Input: $\frac{\Delta f}{\varepsilon}$ and k at P_l $\forall l$; n, total number of parties; m, size of f; b_1; b_2 and b_3, security parameters.

Output: $\forall l \in [n]$, P_l receives ε-differentially private count frequencies and IDs for the records in x^l that are outliers in the global database .

P_1 picks $\mathscr{R}$ uniformly from $\left[0, 2^{b_1}\right]^m$ and sets pf $= \mathscr{R}$.

for each party P_l **do**

  if $P_l \neq P_n$ **then**

    Send pf $=$ pf $+ f\left(x^l\right)$ to $P_{(l+1)}$.

  else

    Send pf $=$ pf $+ f\left(x^l\right)$ to T, a third party .

  end if

end for

T sends pf $=$ pf $+ Y$ to P_1, where $Y \sim \text{Lap}\left(\frac{M}{\varepsilon}\right)^m$.

P_1 receives pf and sets pf $=$ pf $- \mathscr{R}$.

P_1 send pf to all the parties except for P_{n+1} .

for each party P_l **do**

  Set $t = 0$.

  for each $i \in \mathscr{X}$ s.t. $x_i^l > 0$ **do**

    Set $\text{AVFS}_t^l = 0$.

    for each attribute j in $[M]$ **do**

      Set $\text{AVFS}_t^l = \text{AVFS}_t^l + \text{pf}_{j, i_j}\left\{j, i_j \text{ corresponds to the value of attribute } j \text{ in } i\right\}$.

    end for

    $\forall p \in \left[x_i^l - 1\right]$, Set $\text{AVFS}_{t+p}^l = \text{AVFS}_t^l$.

    Set $t = t + x_i^l$.

  end for

end for

for each party P_l **do**

  Find k smallest AVFS in AVFS^l, store them in S_l and their randomly generated IDs in I_l such that for all $t, t' \in \left[\|x^l\|_1\right]$, s_t^l and $s_{t'}^l$ in S_l, $t > t' \Rightarrow s_t^l \geq s_{t'}^l$.

  Make values in S_l and I_l unique $\left\{s_t^l = \alpha s_t^l + n(t-1) + l\right\}$

end for

All parties execute MULTIPARTY_BP with their respective inputs and security parameter b_2 .

All parties together execute MULTIPARTY_SELECT with their respective inputs and security parameter b_3 .

that it has the smallest AVFS globally, also assume that each of three parties has at least three record of type *i*. In this case, if one had achieved uniqueness without being cautious then it could have been possible that only one of the parties gets all the outliers. But our methodology of achieving uniqueness will make sure that each of the party identifies at least one outlier in such a case.

After making AVFS and IDs unique, all parties together execute MULTIPARTY_BP and then MULTIPARTY_SELECT. Hence at the end of POLD_HoF, every party will know outliers in its database along with ε-differentially private count frequencies. The details are specified in Algorithm 5.3.

We would like to stress that the release of differentially private count frequencies in horizontally fragmented database not only helps to enhance efficiency and accuracy, but also enables parties to update the model individually and find outliers locally based on the updated model. Especially, if a party had at least one record that was global outlier then after executing POLD_HoF, it has a threshold for AVFS to asses the outlying behavior of records. Hence, with new records coming in over time, it can not only update differentially private count frequencies, which consequently updates the model, but also determine the outlying behavior based on previously established threshold.

6. Private Outlier Detection in Vertical Fragmentation Case

In this section we present a protocol POLD_VeF for private outlier detection in vertically fragmented database. In such a setting each party collects different features or attributes on the same set of individuals. Thus, we say in vertical fragmentation case $\mathcal{X} = \prod_{l \in [n]} \mathcal{X}_l$, and only party P_l knows $\mathcal{X}_l = \prod_{j \in [M_l]} A_j^l$, where A_j^l is the set of all possible values of j^{th} attribute in P_l's database fragment $x^l \in \mathbb{N}^{\mathcal{X}_l}$, and M_l is the total number of attributes that P_l has. We represent $i = \left(i^1, \ldots, i^l, \ldots, i^n\right) \in \mathcal{X}$ for $i^l \in \mathcal{X}_l, \quad \forall l \in [n]$. We assume that there exist global record identifiers, which give a unique id to each of the local records in each of the database fragment such that the ids for each records in all database fragments corresponding to one global record are the same (i.e., we assume that record linkage is not a problem). Let $\mathcal{I}$ be the collection of such identities for all the records in the database. Thus we can have function φ that computes the global database from its vertical fragments and $\varphi\left(x^1, \ldots, x^n, \mathcal{I}\right) = x$.

Let us now look at how we can compute AVF based outliers in ideal paradigm, when database is vertically distributed among multiple parties. *Note the privacy here corresponds gaining knowledge of the exact value of an attribute for an individual rather than identifying the existence of an individual in the database*. Consider the following example: if two parties want to find average of their salaries without revealing their actual salaries then the identity of the parties is already known; therefore, it is the exact amount that is to be protected.

Ideal Paradigm:

We can use Algorithm 4.1, with couple of modifications to compute outliers, where the database is vertically distributed among parties $P_1,\ldots,P_n$ such that P_l has vertical fragment x^l. Each party P_l also has its auxiliary input γ^l. Following are the changes needed in Algorithm 4.1:

Algorithm 6.1

POLD_VeF

Input: Every party has: $\varepsilon, \mathcal{I}, k, n, b_1, b_2,$ and b_3;

P_l has $\mathcal{X}_l$ and x^l $\forall l \in [n]$

Output: Every party gets IDs of the k global outliers.

1: **for** each party P_l **do**

2: Set $\text{pf}^l = f\left(x^l\right) + \text{Lap}\left(\frac{M_l}{\varepsilon}\right)^{m_l}\left\{m_l = \sum_{j \in [M_l]}^{|A_j^l|}\right\}$.

3: **end for**

4: P_1 picks $\mathcal{R}$ uniformly from $\left[0, 2^{b_1}\right]^{|\mathcal{I}|}$ i.e., $\mathcal{R} \leftarrow \left[0, 2^{b_1}\right]^{|\mathcal{I}|}$.

5: P_1 generates $(pk,\ sk)$ for AHE (for security parameter b_2).

6: P_1 *sends* $E_{pk}[\mathcal{R}]$ and pk to P_n.

7: P_1 sets AVFS $= -\mathcal{R} + (1, \ldots, |\mathcal{I}|)$.

8: All parties agree on the parameter $\alpha > |\mathcal{I}|$.

9: **for** each party P_l for $l \in [n]$ **do**

10: **for** each global record $\left(i^1, \ldots, i^n\right)$ with id in $\mathcal{I}$ do

11: **for** each j in $[M_l]$ **do**

12: Set $\text{AVFS}_{\text{id}} = \text{AVFS}_{\text{id}} + \alpha \times \text{pf}^l_{j, i^l_j}$

$\left\{\left(j, i^l_j\right)\right.$ corresponds to the j^{th} attribute's value in i^l for global record with ID = id$\}$.

13: **end for**

14: **end for**

15: **if** $l \neq n$ **then**

16: Send AVFS to P_{l+1}.

17: **end if**

18: **end for**

19: P_n picks $\mathcal{R}' \leftarrow \left[0, 2^{b_1}\right]^{|\mathcal{I}|}$ and random permutation π.

20: P_n *sends* $E_{pk}[\pi(\mathcal{R}' + \mathcal{R})]$ to P_1 and $\pi(\text{AVFS} - \mathcal{R}')$ to P_2.

21: P_1 and P_2 execute protocol SELECT with their respective inputs $\pi(\mathcal{R}' + \mathcal{R})$ and $\pi(\text{AVFS} - \mathcal{R}')$ and security parameter b_3 to find indices of k smallest entries in AVFS.

22: P_1 through P_n informs all the parties of all IDs corresponding to k outliers in global database x.

Input: Each party inputs its vertical fragment of the database along with $\mathcal{I}$, the global record identifiers. Line 1: Global database is computed using φ; that is,

$$x = \varphi\left(x^1, \ldots, x^n, \mathcal{F}\right)$$

Line 14: Send only outlier records to the party: there is no need to send count frequencies.

As for the security definition, the changes are of course in regards to input, output and auxiliary input of the parties since once the global database is constructed, the remaining procedure is the same: thus, static semi-honest parties securely compute AVF based outliers in ideal paradigm only if no party receives information beyond its outliers in global database, which are computed using ε-differentially private count frequencies by $\mathcal{T}$, with negligible probability in the input size.

As opposed to horizontal fragmentation case, in this case we do not reveal the count frequencies. Revealing count frequencies would only compromise privacy unnecessarily: among other things it will reveal to each party the information on the attributes of all other parties – in case of horizontal fragmentation, parties already had this knowledge. Furthermore, even if the parties are provided with the count frequencies in vertical fragmentation case, such information will be useless on its own without the knowledge of correspondence of records across database fragments. Therefore, we opt for a strategy where count frequencies are kept hidden; this design choice results in a protocol which provides higher privacy than the POLD_HoF, but requires cryptographic operations linear in term of number of records.

Real Paradigm:

Let $x \in \mathbb{N}^{\mathcal{X}}$ be vertically fragmented among parties $P_1,\ldots,P_n$ with P_l in possession of vertical fragment x^l and having the auxiliary input γ^l which is the same as in the ideal paradigm. The parties want to compute AVFS OLs, AVF based outliers, such that each party receives AVFS_OLs^l, id's of its outliers in global context. It should be noted that in case of vertical fragmentation, for all $l,l' \in [n]$, AVFS_OLs^l = AVFS_OLs$^{l'}$. We consider a protocol Π_v, which computes AVF based outliers in the vertical fragmentation case, to be secure if it fulfills the following security/privacy definition.

Definition 6.1. (Privacy for Vertically Partitioned Data)—Let φ, $\mathcal{F}$ and $\mathcal{X}_l \forall l \in [n]$ be as defined previously. For all $x = (x^1,\ldots,x^n)$, where $x^l \in \mathcal{X}_l$, the corresponding $\mathcal{F}$, and $k < \|\varphi\left(x^1, \ldots, x^n, \mathcal{F}\right)\|_1$, we say a protocol Π_v securely computes AVF based outliers in database x in the presence of static semi-honest parties, among whom the database x is vertically distributed, if for every PPT adversary $\mathcal{A}_l$(for $l \in [n]$) in real paradigm there exits a PPT adversary $\mathcal{S}_l$ in ideal paradigm such that

$$\text{REAL}_{\Pi_v, \mathcal{A}_i}\left(x^1, \ldots, x^n, k, \mathcal{F}\right) \overset{c}{\equiv} \text{IDEAL}_{\text{AVFS\_OLs}, \mathcal{S}_i}\left(x^1, \ldots, x^n, k, \mathcal{F}\right).$$

We propose protocol POLD_VeF to realize Π_v, the details of which are given in Algorithm 6.1. The overall idea in Algorithm 6.1 is to compute ε-differentially private count frequencies (ε-f) without revealing ε-f to any of the party; further, parties compute AVFS using ε-f, making sure that AVFS are also kept secret. Finally, the parties identify the records with k smallest AVFS without revealing anything else.

First, notice that no collaboration is required to calculate the count frequencies for each party. Thus each party P_l can compute count frequencies for its database fragment x^l i.e., $f(x^l)$. P_l can then add laplacian noise from $\text{Lap}\left(\frac{M_l}{\varepsilon}\right)$ to each of the count frequency that is $\text{pf}^l = f\left(x^l\right) + \text{Lap}\left(\frac{M_l}{\varepsilon}\right)^{m_l}$, where $m_l = \sum_{j \in [M_l]} \left|A_j\right|$. The scale for laplace noise for party P_l is $\frac{M_l}{\varepsilon}$ because P_l has M_l attributes and at the maximum for an individual all the values for M_l attributes may change.

Once all the parties have computed differentially private count frequencies for their respective attributes, they can employ global record identifiers $\mathcal{I}$ to compute AVFS for each record, but as per privacy definition 6.1 we cannot release count frequencies to any of the parties. We employ secure sum to accomplish this, but the final value is kept split between the two parties, let us say P_1 and P_2.

Next, the parties need to identify the k smallest AVFS: this could have easily been done by P_1 and P_2 through collaboratively executing SELECT protocol on the split AVFS. But this is not possible because of the following two restrictions: (i) AVFS are not unique, (ii) SELECT would reveal ordering over AVFS to P_1 and P_n; thus, some extra processing is needed before we can execute SELECT.

In order to make AVFS unique we can follow a procedure similar to the one we used earlier in POLD HoF; that is, scale each AVFS, v_i, by multiplying with a positive integer a, which is greater than the total number of records, and adding a unique positive integer value to it i.e., $(a v_i + i)$. The details of the procedure can be found in Algorithm 6.1.

In order to stop leakage of the information related to the ordering of AVFS, we make P_1 generate public private key pair (pk,sk) for additive homomorphic encryption, and then send its encrypted shares along with pk to P_n. P_n randomizes the received P_1's encrypted shares by adding uniformly random numbers to each of the encrypted AVFS of P_1; P_n then performs a random permutation on the newly generate shares of P_1 and send these new encrypted shares to P_1. Next it updates its shares in accordance with the randomization that P_n applied on P_1's encrypted share so that the splits when combined should give the correct value for AVFS. P_n then performs the permutation π on its updated random shares, and it send them to P_2.

P_1 and P_2 now have random shares of the AVFS without knowing the correspondence between the global record IDs and the splits. Both parties together run SELECT protocol to identify the splits corresponding to the k smallest values. P_1 sends P_n the indices

corresponding to the k smallest AVFS values. Since P_n knows the permutation π, it identifies the corresponding record IDs and sends them to all the parties; this informs each party of the records that are outliers.

This procedure computes AVF based outliers by employing ε-differentially private AVFS. As compared to POLD HoF, in this case the over all noise is less; this is due to the fact that attributes are divided among multiple parties, who in order to attain differential privacy need to add noise with magnitude proportional to the number of attributes each party locally has. Since for $M_l < M \forall l \in [n]$, the utility should be higher.

7. Complexity Analysis

In this section we discuss the computational and communication complexities of the proposed protocols. We first, analyze POLD_HoF, the protocol for outlier detection when database is fragmented horizontally among the parties, then we carry out similar analysis for POLD_VeF, that detects outliers in vertically fragmented databases. The analysis is presented in term of dimensionality and size of the database.

POLD_HoF:

The computational complexity for this protocol grows linearly with respect to the number of records (N) in the distributed database x. Computational complexity for each party to compute count frequencies is $O(MN_l)$ because we just need to count all the values in database for each attribute, here M is the number of attributes in database x and N_l is the number of records in horizontal fragment x^l of x, which is kept by party P_l. Computational overhead to make count frequencies differentially private is $O(ML)$, where $L = \max_{j \in [n]} \left| \mathcal{A}_j \right|$. In most of the real world situations L is not large compared to the size of database i.e., the number of records in each of the fragment; it is important to note that in high dimensional datasets conventional and basic outlier detection techniques do not work well; hence our assumption is quite reasonable. It costs $O(MN_l)$ operations to compute AVFS for each party P_l. Since at the very least, by our assumption, $L < N_l$, we can say that overall computational cost for computing differentially private AVFS is $O(MN_l)$ for party P_l, where as the over all computational cost for n parties is $O(MN)$ with $N = \sum_{l \in [n]} N_l$. In order to compute differentially private AVFS each party only transmits and receives $O(ML)$ messages.

Once the parties have AVFS for each of their record, each party P_l finds k local outliers with computational complexity $O(N_l \log(k))$. This computation requires no collaboration among the parties; thus no communication required.

So far we have not employed any (expensive) cryptographic operations; and the only data items for which we would require cryptographic operations are the local outliers, which are kn in totals. In the context of big data, the size of database x would be immense when compared to kn (i.e., $kn << N$). Therefore, POLD_HoF requires cryptographic operations, which are the main cause of slowing down secure/private computation protocols, on a very tiny fraction of the database.

In Algorithm 5.3, to find global outliers from local outliers, firstly the IDs and AVFS are made unique that requires $O(kn)$ operations. Parties then collaboratively execute MULTIPARTY_BP, which carries $O(k)$ computation and communication overhead per each party except for P_1,P_2 and P_n for which the overhead reaches $O(nk)$; this is the same as overall overhead incurred in total on all the parties. In Algorithm 5.2, P_1 and P_2 take $O(kn\log(k))$ to find the k smallest cryptographic operation that is computing comparator garbled circuit, the details on communication and computation complexity for garbled circuit can be found in[37]. Next each party, bears a communication and computation overhead of $O(k)$ to find its global outliers, which in total amount to $O(nk)$.

The overall asymptomatic computational complexity fo the protocol per party and in total is $O(ML + MN_L + N_l \log(k) + kn\log(k)) = O((M + \log(k))N_l)$ and $O((M + \log(k)N))$ respectively. Whereas the communication complexity is $O(ML)$ for all the parties except for P_1,P_2 and P_3; for P_1 and P_2 it amounts to $O(ML+kn\log(k))$, and for P_3 it is $O(ML+kn)$. It is apparent that communication and computation complexity is contingent on the number of participating parties and the targeted number of outlier; thus our proposed scheme is highly scalable.

POLD_Vef:

This protocol has a comparatively simpler analysis. First, all the parties calculate differentially private count frequencies for all the attribute values in their database fragment, which requires $O(M_l L_l)$ operation, where M_l is the number of attributes in database fragment kept by party P_l and L_l is the maximum number of values P_l has for its attribute. To compute AVFS in split form every party P_l bears the computation and communication complexity $O(M_l N)$ and $O(N)$ respectively. This in total amounts to $O(MN)$ and $O(nN)$ computation and communication complexity respectively to compute AVFS in split form.

P_1 and P_2 need $O(k \log(N))$ communication and computation overhead in computing k-smallest AVFS i.e., top-k AVF based outliers. Here again for most realistic settings k will be really small compared to N; thus the overhead for this computational step will be linear in term of total records in the database.

To compute the global outliers from the permuted and re-randomized splits of AVFS, it requires $O(N \log(k))$ garbled circuit computation for P_1 and P_2 and $O(k)$ messages to inform the parties of global outliers. Thus overall computational and communicational cost is $O((M + \log(k)N))$ and $O((n + \log(k))N)$.

8. Security Analysis

We now present the security analysis for the proposed protocols through modular sequential composition (MCS)[7]. We begin by analyzing POLD HoF for $n \geq 3$. Most of the analysis for $n = 2$ is similar to the one presented in[26] except for the noise addition for making count frequencies differentially private, which is straightforward and directly follows from the assumption of non-collusion and semi-trusted nature of the parties. For $n \geq 3$, we analyze the security of sub-protocols first.

MULTIPARTY_BP only reveals random shares of permuted vectors to two parties P_1 and P_2, who neither know random shares of the other party nor are they aware of randomly picked permutation by P_n. P_n only has access to encrypted AVFS and their randomly generated unique IDs. Although P_n does know the order of values in all S_i, this does not reveal any information due to AHE being semantically secure; P_n will not be able to distinguish between an actual S_i and a random input. The view of the adversary in real paradigm can be simulated by the adversary in ideal paradigm.

MULTIPARTY_SELECT uses garbled circuits for secure comparison. Garbled circuits are known to be secure and reveal no information except for the output of comparator circuit[38]. Again the order obtained on the random shares does not expose any extra information to P_1 or P_2 as this view is simulatable by the adversary in real paradigm. For any party who gets the I^k will obtain no information beyond the IDs of its AVFS that are part of globally k smallest AVFS. If parties choose $N' \in \mathbb{N}$ for $\mathbb{Z}_{N'}$, the field from which IDs are picked, such that $\log(N') > kn$ then the collision probability that two randomly chosen IDs are the same will be approximately $\frac{1}{2^{O(kn)}}$, which is exponentially small.

From above and by following *Modular Sequential Composition*[7] it follows that the protocol POLD_HoF is secure since all the sub-protocols it uses are secure and the view of each party is simulatable by the adversary in the ideal paradigm.

POLD_VeF, Algorithm 6.1, uses secure sum[8] to create random shares of all the AVFS, wherein P_n receives the random shares of AVFS: one in encrypted form and one in plain-text form. Neither of these reveal any information since inferring information from the plain-text share is equivalent to breaking the security of the pseudorandom generator used to generate the random number added and inferring information from the encrypted share would imply breaking semantically secure encryption. Therefore, P_n's view is simulatable by the adversary in ideal paradigm. The same is true for the view of the adversaries controlling P_1 and P_2. Achieving uniqueness of AVFS values assures the correct results from SELECT (as it only works on unique values). Though SELECT reveals differentially private AVFS based ordering on tuples for each party, it does not disclose any extra information that parties would not have if differentially private AVFS were provided to them. Finally, the order information on the splits between P_1 and P_2 is just a partial order over random shares: this view can be simulated by the adversary in the ideal paradigm; thus the protocol is secure.

9. Experimental Evaluation

We first present the experimental setup and then discuss the results obtained.

9.1 Experimental Setup

We carried out extensive empirical analysis using five real datasets obtained from the UCI Machine Learning repository[1]. The algorithms were implemented in python using the panda and numpy framework and experiments conducted on a workstation with an i5 2.5 Ghz processor and 6 GB of RAM.

9.2 Results and Discussion

Since quantifiable privacy is provided through the use of differential privacy, the proposed approach only needs to be evaluated in terms of its effectiveness in detecting outliers and its scalability. Since the suitability and effectiveness of the base AVF technique has been evaluated in the literature[19], in this work we simply measure effectiveness of the privacy-preserving approach by comparing how well it performs in terms of detecting outliers with respect to the original AVF approach. Specifically, we report the *relative accuracy* – the number of outliers found by the differentially private algorithm as compared to the number of outliers captured by the base algorithm. This allows us to determine the effect of privacy on the approach. In all of the following experiments, we limited the number of outliers to within 10% of the size of each dataset, which is quite a reasonable assumption. We then ran the differentially private algorithm and considered the records with the smallest k AVF scores to be outliers. k was varied to have the values 2%, 4%, 6%, 8% and 10% for each dataset. To get an understanding of how well the proposed technique works in real life, we used datasets having only categorical, only numeric or a mixed set of attributes. Numeric attributes were converted into categorical by discretization using bucketing with an appropriate range. Experiments were run 10 times and average results are reported.

Wisconsin Breast Cancer: This dataset has 699 records and 9 attributes. Each record is labeled as either benign or malignant. Following Harkins et al.[12], we randomly picked a subset of records labeled as malignant such that they made up 8% of the dataset, and we considered them to be outliers. Figure 3 shows the results obtained for different levels of ε (corresponding to different privacy levels) and different values of k. In each case, the relative accuracy is computed. As is to be expected, as ε increases, the accuracy of the privacy preserving algorithm approaches that of the base algorithm. Indeed, for epsilon values greater than 0.1, the relative accuracy is more than 90%, while for epsilon values close to .38 our relative accuracy is almost the same as that of the base algorithm for all values of k.

Tic-Tac-Toe Endgame: This dataset contains 958 records and 9 attributes. Each record encodes one possible configuration of the end game, where the class is positive if the first player wins and negative otherwise. Since the number of negative records was smaller, we considered them to be outliers. Furthermore, for evaluation we subsampled the negative records to ensure that number of outliers was no more than 8% of the dataset. From Figure 4 one might infer that our scheme is doing poorly. The fact we seem to be under performing is because of the degraded performance of the base algorithm that only finds exactly 12 outliers for all values of k i.e. 2%, 4%, 6%, 8% and 10%. In this dataset the assumption made of AVF based algorithm to work does not hold really well. AVF scores of outlier and non-outliers are quite close; hence addition of noise perturbed AVFS in such a way that it scatters these particular 12 outliers among the rows that have AVF scores among k smallest values for k = 10%. This is why when value of k increase our accuracy approaches that of the base algorithm.

Chess Endgame Database: This dataset contains 28056 records and 6 attributes. It has eighteen classes from 0 to 17. Each of the class labels from 0 to 16 represents the optimal depth-of-win (number of moves) for White King or Rook to win the game. Whereas class

label 18 depicts a draw against the Black King. We have taken class with label zero and one as outliers. It can be seen in Figure 5 that from $\varepsilon = 0.04$ we are getting very good results for all values of k. It should be noted that outliers detected by base algorithm at 2% are 11 but from 4% onwards the count of true outliers remained fixed at 17.

Breast Cancer: In this dataset we introduced 21 outliers to check how the base and the modified algorithm work. We created outliers by following the definition of an ideal outlier[19], which states an ideal outlier point in a categorical dataset is one whose each and every attribute value is extremely irregular (or infrequent). From Figure 6 we can see the modified algorithm starts providing comparable results to the base algorithm as the ε increases.

Spam Base: Spambase contains samples of spam and ham emails, where ham is eight percent of the data. There are fifty-seven numeric attributes. We took a random sample from the spam to make it 10% of the dataset. Next, attributes were discretized and only attributes having difference in their count frequency for spam and ham were retained. This gave us sixteen attributes. We carried out outlier detection on this modified dataset. Figure 7 shows the results, which are really promising.

Looking into the two most diverse result obtained in our experiments, we thought of further exploring the results of **Wisconsin Breast Cancer** and **Tic-Tac-Toe Endgame** to better understand why the differentially private algorithm provides comparable results to the base algorithm in case of one dataset and why it fails to replicate the result of the base algorithm for the other dataset. To see this difference we designed an experiment where we first find out the maximum AVF value for Outliers and the minimum AVF value for Non-Outliers for all values of k i.e. 2%, 4%, 6%, 8% and 10% at different values. Then we took the average of the AVF scores corresponding to Outliers and Non-Outliers at each $\epsilon$and plotted them for each dataset (Fig8, Fig9), along with the corresponding values for the base algorithm (with no noise added).

Figure 8 shows this for the Wisconsin Breast Cancer dataset. We can see that there is a difference in the average AVF scores of the outliers with respect to the average AVF scores of non-outliers in the base case algorithm (where no noise is added). As can be seen, this difference still persists for different values of ϵ, which enables the differentiation of outliers and non-outliers even for the differentially private algorithm. Due to this, the differentially private algorithm provides comparable results to the base algorithm. Figure 9 shows the corresponding results for the Tic-Tac-Toe dataset. Here, it is clear that there is a big difference between the average AVF scores for outliers and non-outliers in case of the base algorithm(where no noise is added), but once we add noise to the data to make it differentially private (for all values of ϵ) we see that the average AVF score of max outlier and min nonoutlier are almost the same. This prevents the differentiation between outliers and non-outliers due to which the results of the differentially private algorithm deviate significantly from the base case.

Key Conclusions: The experiments carried out over various datasets with different number of attributes and sizes of outlier set as compared to the size of dataset tells us that

the number of outliers, which usually is small, or the number of attributes themselves are not among the main reasons for degradation of utility in achieving differential privacy for reasonable values of the privacy parameter ϵ. Rather, it is the gap between the maximum AVFS for outliers and minimum AVFS for non-outliers in a dataset that decides how good the utility will be: the larger the gap, the better the utility. This critically depends on the dataset itself. In our experiments, we generally found the gap to be sufficient to ensure good utility. However, in cases where the accuracy is critical, the use of differential privacy is still questionable as it cannot be achieved without some loss of utility which occurs due to the addition of noise, but if the AVFS gap is sufficient, we can reliably find outliers even without compromising privacy of individuals.

Performance: We have also included the simulation results that empirically analyze the computational overhead caused by the secure operation that is encryption, operation carried out on encrypted messages and secure comparison. Figure 10 shows the overhead introduced due to the secure operations in the POLD_HoF protocol with respect to varying the number of participating parties, while setting k to be one. On the other hand, Figure 11 presents the computational overhead for $n = 10$ parties while varying values of k. In both cases computational cost increases linearly, which shows our approach is quite efficient.

10. Related Work

The research in privacy-preserving data mining spans many areas: data perturbation techniques[4,27,22], and cryptographic (secure multiparty computation based) techniques[34]. Data-perturbation-based privacy-preserving techniques, as the name suggests, perturb values of attributes by adding noise. Special techniques are used to reconstruct the original distribution (not the actual data values). They all rely on fundamental property that the randomized data set may not reveal private data, while still allowing data analysis to be performed on them. Kargupta et al.[17] questioned the use of random additive noise and pointed out that additive noise can be easily filtered out using spectral filtering techniques causing a privacy breach of the data. Other problems have also been pointed[23,16]. Cryptographic techniques Fig. 10: Computational overhead for varying number of parties (n), while $k = 1$ have been developed for many data analysis tasks such as classification[21,33,35], clustering[30,20], and association analysis[29,32], but outlier detection has not been studied to a large extent.

For outlier detection on categorical data, *distance-based* approaches do not make sense. Distance based approaches are geared towards numerical data and thus are more applicable to numerical datasets or ordinal data that can be easily transformed to suitable numerical values. Also Distance-based approaches do not make assumptions for the distribution of the data since they essentially compute distances among points. These approaches become impractical for large datasets. Knorr et al.[18], had proposed an improved version of the original-distance based approach but the complexity of their approach is still quadratic in the number of nearest neighbors.

Density-based outlier detection algorithm used in[28] is based on the concept of relative density-based, which is a simplified version of Local Outlier Factor (LOF) described in[6]. Of

course, it cannot obtain the same accuracy as the original LOF. Though[28] suggested that extending simplified LOF to the original LOF is straightforward, it still requires some degree of effort. Furthermore, data is sparse in highdimensional spaces rendering density-based methods problematic[36].

In this paper we are using the differential privacy framework[9] along with some multi-party computation methods to ensure the privacy of data, while finding outliers in data using attribute value frequency based approach[19] in a setting where data is either vertically of horizontally distributed among multiple parties. Though[5] provides a way to add noise for a Laplace distribution securely to an aggregation of data values from multiple parties, but their model is quite different; in our case parties, instead of some independent server, should be able to add noise.[10] also proposed protocols for addition of laplacian noise by two parties for both malicious and rational adversary cases, but they have a huge overhead as compared to our proposal, where parties are assumed to be semi-honest.[26] presented protocols for distributing shares of values kept by two parties and then selecting k smallest values securely. However these protocols only work for the two party case. Overall, our work is orthogonal to the existing work on outlier detection and improves on it by developing an efficient solution that provides end-to-end privacy.

11. Conclusion and Future Work

Outlier detection is a critical analytics task that can be computationally quite intensive, especially when the data is split between multiple parties and privacy needs to be protected. In this paper, we have developed a very scalable and efficient AVF based outlier detection algorithm that provides end-to-end privacy in terms of both the computation as well as the results. We then develop protocols that can find AVF based outliers in a private manner, when the data is centralized, and when the data is either horizontally or vertically distributed. The experimental results on real data are quite promising.

In this work, we compromised a bit on privacy by releasing differentially private count frequencies to improve utility. In the future we would like explore ways to reduce the information leakage, while not letting the utility, for example, false positive rate, suffer as compared to the reference non-private AVF based detection methodology. Our study is also limited to semi-honest adversaries. Though semi-honest models are a good point of entry and are representative of many of the real-world situations, there are cases where the parties would try to cheat or act maliciously. In the future, we would like to extend the protocols to make them resilient towards such covert or malicious adversaries, while still being efficient. Finally, while our focus was limited to the AVF based outlier detection model, which we believe is a very good starting point for outlier detection, in the future, we will work on extending this to other outlier models such as distance or density based outliers.

Acknowledgements

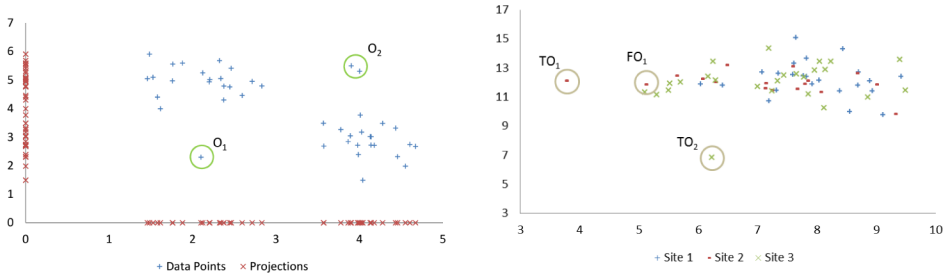
Research reported in this publication was supported by the National Institutes of Health under award R01GM118574, by the National Science Foundation under awards CNS-1422501, CNS-1624503, and CNS-1564034. The work of Shafiq is supported by HEC grant under the PAK-US Science and Technology Cooperation Program. The work of Adam is supported by the National Academies of Sciences, Engineering, and

Medicine under the PAK-US Science and Technology Cooperation Program. The content is solely the responsibility of the authors and does not necessarily represent the official views of the agencies funding the research.

References

1. Uci machine learning repository. http://archive.ics.uci.edu/ml/, jul 2016.
2. Aggarwal CC Privacy-Preserving Data Mining: Models and Algorithms, volume 34 of Advances in Database Systems. Springer, 2008.
3. Aggarwal CC. Outlier analysis. 2013.
4. Agrawal R and Srikant R. Privacy-preserving data mining In ACM Sigmod Record, volume 29, pages 439–450. ACM, 2000.
5. Anandan B and Clifton C. Laplace noise generation for two-party computational differential privacy In Privacy, Security and Trust (PST), 2015 13th Annual Conference on, pages 54–61. IEEE, 2015.
6. Breunig MM, Kriegel H-P, Ng RT, and Sander J. Lof: identifying density-basedlocal outliers In ACM sigmod record, volume 29, pages 93–104. ACM, 2000.
7. Canetti R. Security and composition of multiparty cryptographic protocols. Journal of CRYPTOLOGY, 13(1):143–202, 2000.
8. Clifton C, Kantarcioglu M, Vaidya J, Lin X, and Zhu MY. Tools for privacypreserving distributed data mining. ACM Sigkdd Explorations Newsletter, 4(2):28–34, 2002.
9. Dwork C, Roth A, et al. The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, 9(3–4):211–407, 2014.
10. EIGNER F, KATE A, MAFFEI M, and PAMPALONI F. Achieving optimal utilityfor distributed differential privacy using secure multiparty computation. Applications of Secure Multiparty Computation, 13:81, 2015.
11. Ester M, Kriegel H-P, Sander J, Xu X, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd, volume 96, pages 226–231, 1996.
12. Hawkins S, He H, Williams G, and Baxter R. Outlier detection using replicatorneural networks In International Conference on Data Warehousing and Knowledge Discovery, pages 170–180. Springer, 2002.
13. He Z, Deng S, and Xu X. An optimization model for outlier detection in categoricaldata In International Conference on Intelligent Computing, pages 400–409. Springer, 2005.
14. He Z, Deng S, Xu X, and Huang JZ. A fast greedy algorithm for outlier mining In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 567–576. Springer, 2006.
15. He Z, Xu X, Huang JZ, and Deng S. Fp-outlier: Frequent pattern based outlierdetection. Comput. Sci. Inf. Syst, 2(1):103–118, 2005.
16. Huang Z, Du W, and Chen B. Deriving private information from randomized data In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, Baltimore, MD, 6 13–16 2005 ACM.
17. Kargupta H, Datta S, Wang Q, and Sivakumar K. On the privacy preserving properties of random data perturbation techniques In Data Mining, 2003. ICDM 2003. Third IEEE International Conference on, pages 99–106. IEEE, 2003.
18. Knorr EM, Ng RT, and Tucakov V. Distance-based outliers: algorithms andapplications. The VLDB JournalThe International Journal on Very Large Data Bases, 8(3–4):237–253, 2000.
19. Koufakou A, Ortiz EG, Georgiopoulos M, Anagnostopoulos GC, and Reynolds KM. A scalable and efficient outlier detection strategy for categorical data In 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), volume 2, pages 210–217. IEEE, 2007.
20. Lin X, Clifton C, and Zhu M. Privacy preserving clustering with distributed EMmixture modeling. Knowledge and Information Systems, 8(1):68–81, 7 2005.
21. Lindell Y and Pinkas B. Privacy preserving data mining In Advances in Cryptology – CRYPTO 2000, pages 36–54, New York, NY, Aug. 20–24 2000 Springer-Verlag.
22. Liu K, Kargupta H, and Ryan J. Random projection-based multiplicative dataperturbation for privacy preserving distributed data mining. IEEE Transactions on knowledge and Data Engineering, 18(1):92–106, 2006.

23. Mielikainen T. Privacy problems with anonymized transaction databases In Discovery Science: 7th International Conference Proceedings, volume 3245 of Lecture Notes in Computer Science, pages 219 – 229. Springer-Verlag, Jan. 2004.
24. Otey ME, Ghoting A, and Parthasarathy S. Fast distributed outlier detection inmixed-attribute data sets. Data Mining and Knowledge Discovery, 12(2–3):203–228, 2006.
25. Paillier P et al. Public-key cryptosystems based on composite degree residuosityclasses In Eurocrypt, volume 99, pages 223–238. Springer, 1999.
26. Qi Y and Atallah MJ. Efficient privacy-preserving k-nearest neighbor search In Distributed Computing Systems, 2008. ICDCS'08. The 28th International Conference on, pages 311–319. IEEE, 2008.
27. Rizvi SJ and Haritsa JR. Maintaining data privacy in association rule mining In Proceedings of 28th International Conference on Very Large Data Bases, pages 682–693, Hong Kong, Aug. 20–23 2002 VLDB, VLDB Endowment.
28. Shaneck M, Kim Y, and Kumar V. Privacy preserving nearest neighbor search In Machine Learning in Cyber Trust, pages 247–276. Springer, 2009.
29. Vaidya J and Clifton C. Privacy preserving association rule mining in vertically partitioned data In The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 639–644, Edmonton, Alberta, Canada, 7 23–26 2002 ACM.
30. Vaidya J and Clifton C. Privacy-preserving k-means clustering over vertically partitioned data In The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 206–215, Washington, DC, Aug. 24–27 2003 ACM.
31. Vaidya J and Clifton C. Privacy-preserving outlier detection In Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on, pages 233–240. IEEE, 2004.
32. Vaidya J and Clifton C. Secure set intersection cardinality with application to association rule mining. Journal of Computer Security, 13(4):593–622, Nov. 2005.
33. Vaidya J, Clifton C, Kantarcioglu M, and Patterson AS. Privacy-preserving decision trees over vertically partitioned data. ACM Trans. Knowl. Discov. Data, 2(3):1–27, 2008.
34. Vaidya J, Clifton CW, and Zhu YM. Privacy preserving data mining, volume 19. Springer Science & Business Media, 2006.
35. Vaidya J, Kantarcioglu M, and Clifton C. Privacy preserving naive bayes classification. International Journal on Very Large Data Bases, 17(4):879–898, 7 2008.
36. Wei L, Qian W, Zhou A, Jin W, and Jeffrey XY. Hot: Hypergraph-based outliertest for categorical data In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 399–410. Springer, 2003.
37. Yao AC. Protocols for secure computation. In Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science, pages 160–164, 1982.
38. Yao AC-C. How to generate and exchange secrets In Foundations of Computer Science, 1986., 27th Annual Symposium on, pages 162–167. IEEE, 1986.



(a) Vertically partitioned data (b) Horizontally partitioned data

Fig. 1:
Illustrative example of why purely local computation is typically inaccurate

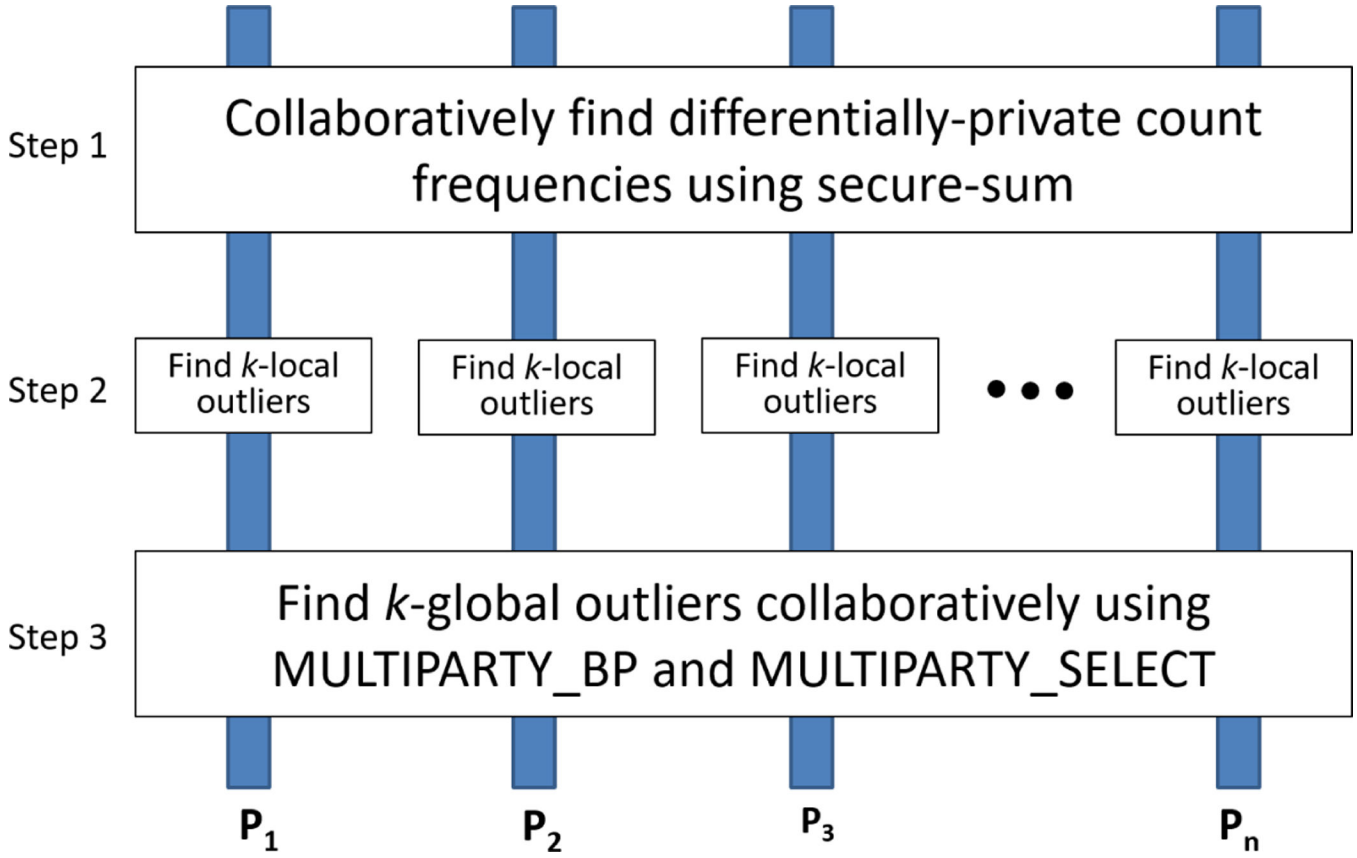


Fig. 2:
Protocol overview

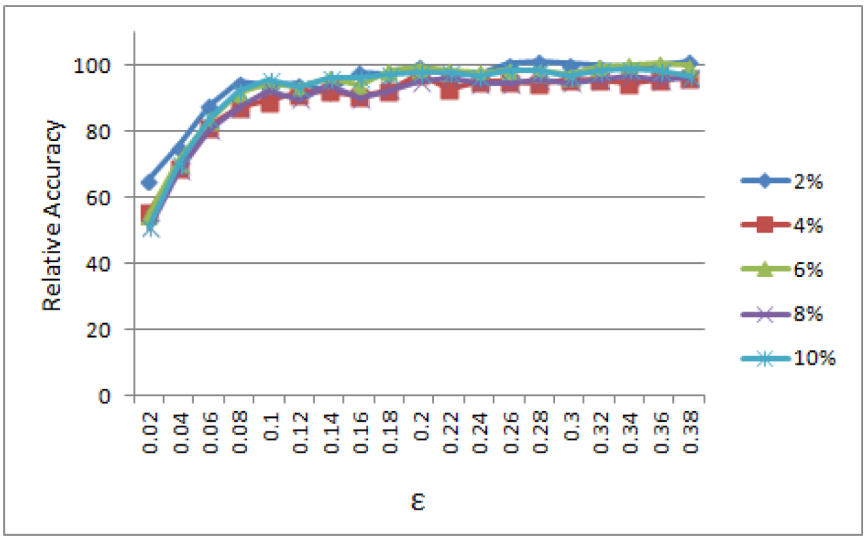


Fig. 3:
Wisconsin Breast Cancer

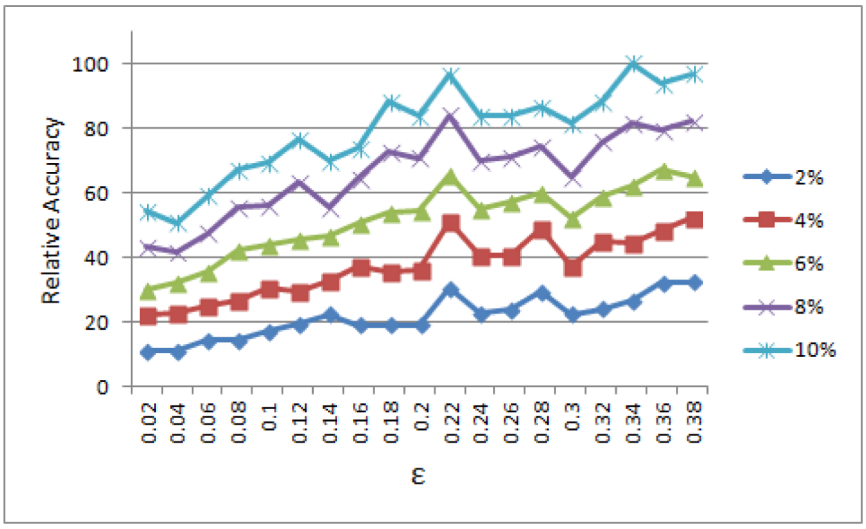


Fig. 4:
Tictac-Tac-Toe

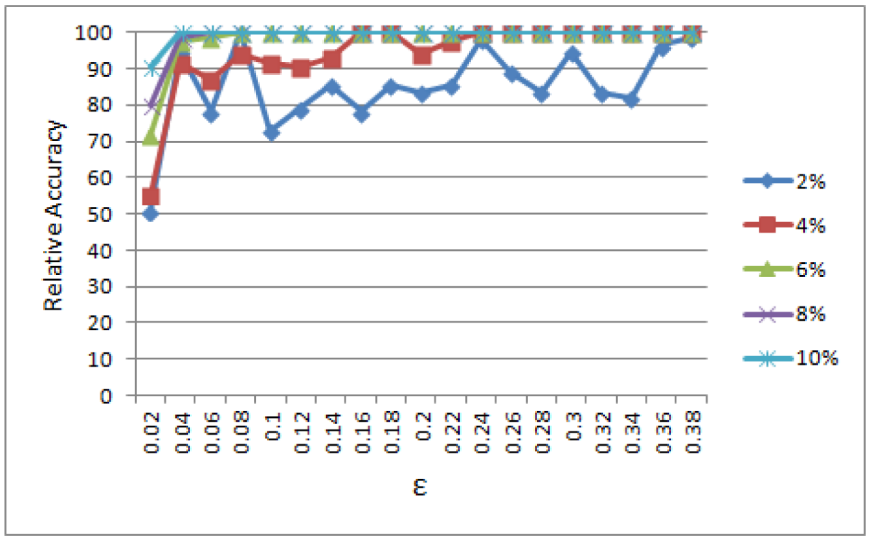


Fig. 5:
Chess Endgame

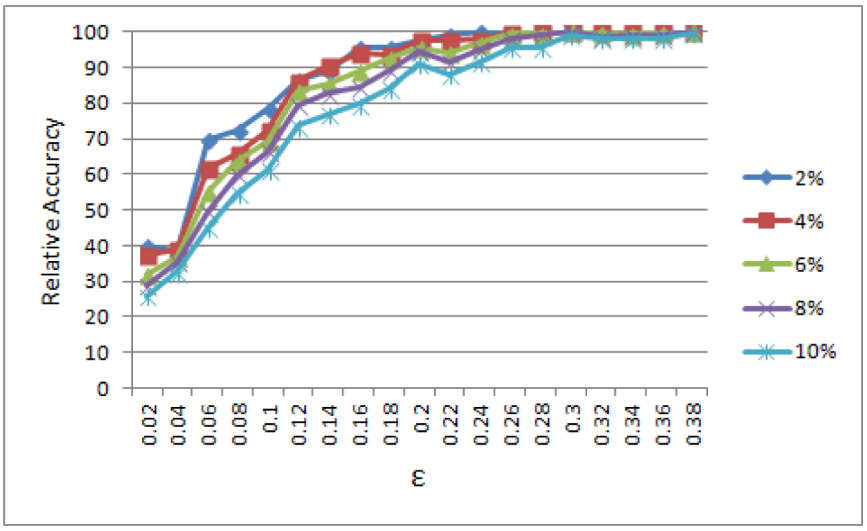


Fig. 6:
Breast Cancer

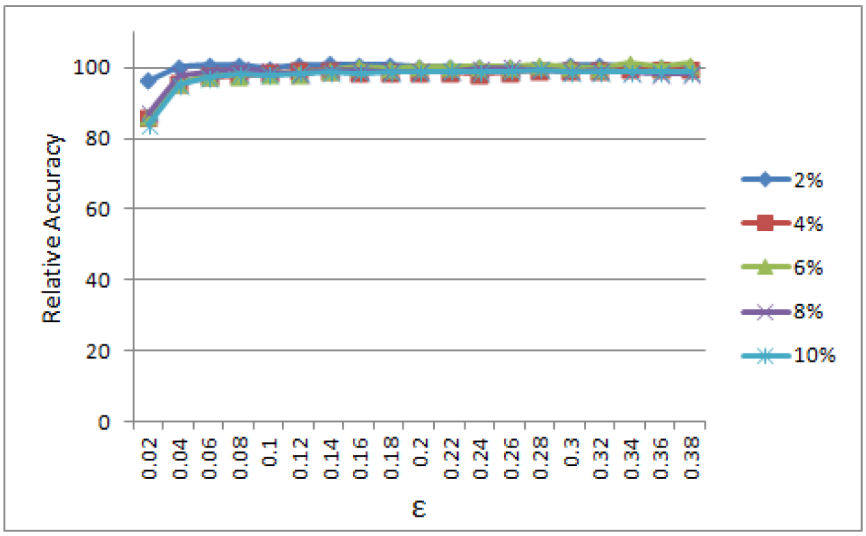


Fig. 7:
Spam Base

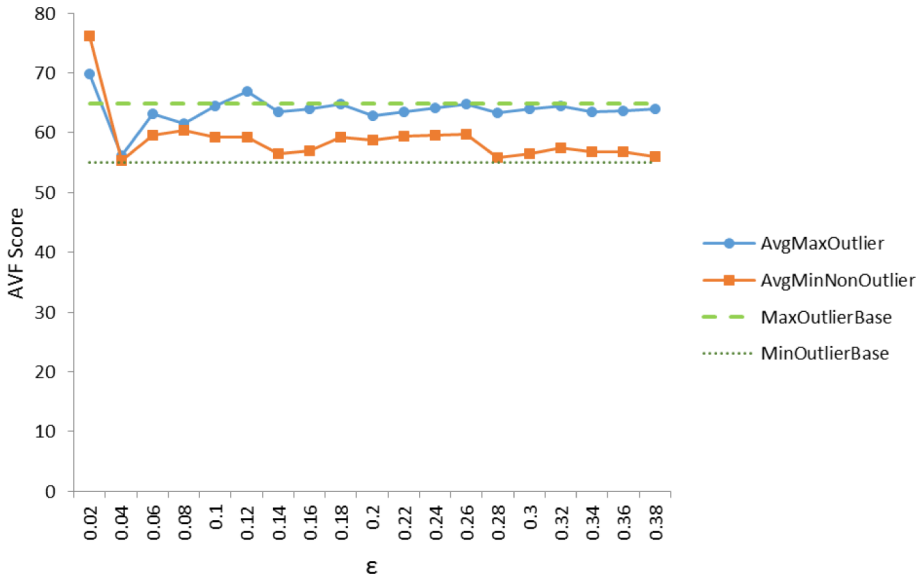


Fig. 8:
Outlier vs Non-Outlier for Wisconsin Breast Cancer

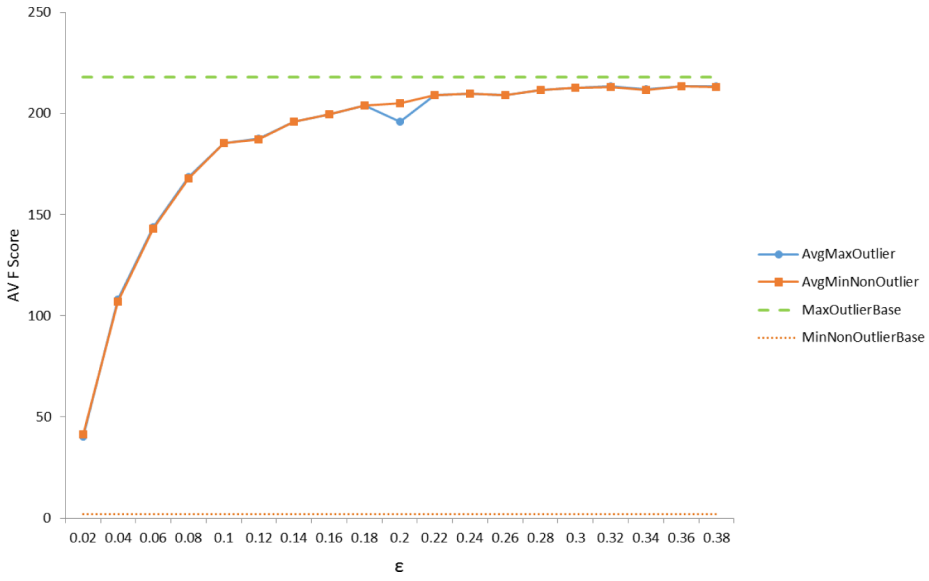


Fig. 9:
Outlier vs Non-Outlier for Tictac-Tac-Toe

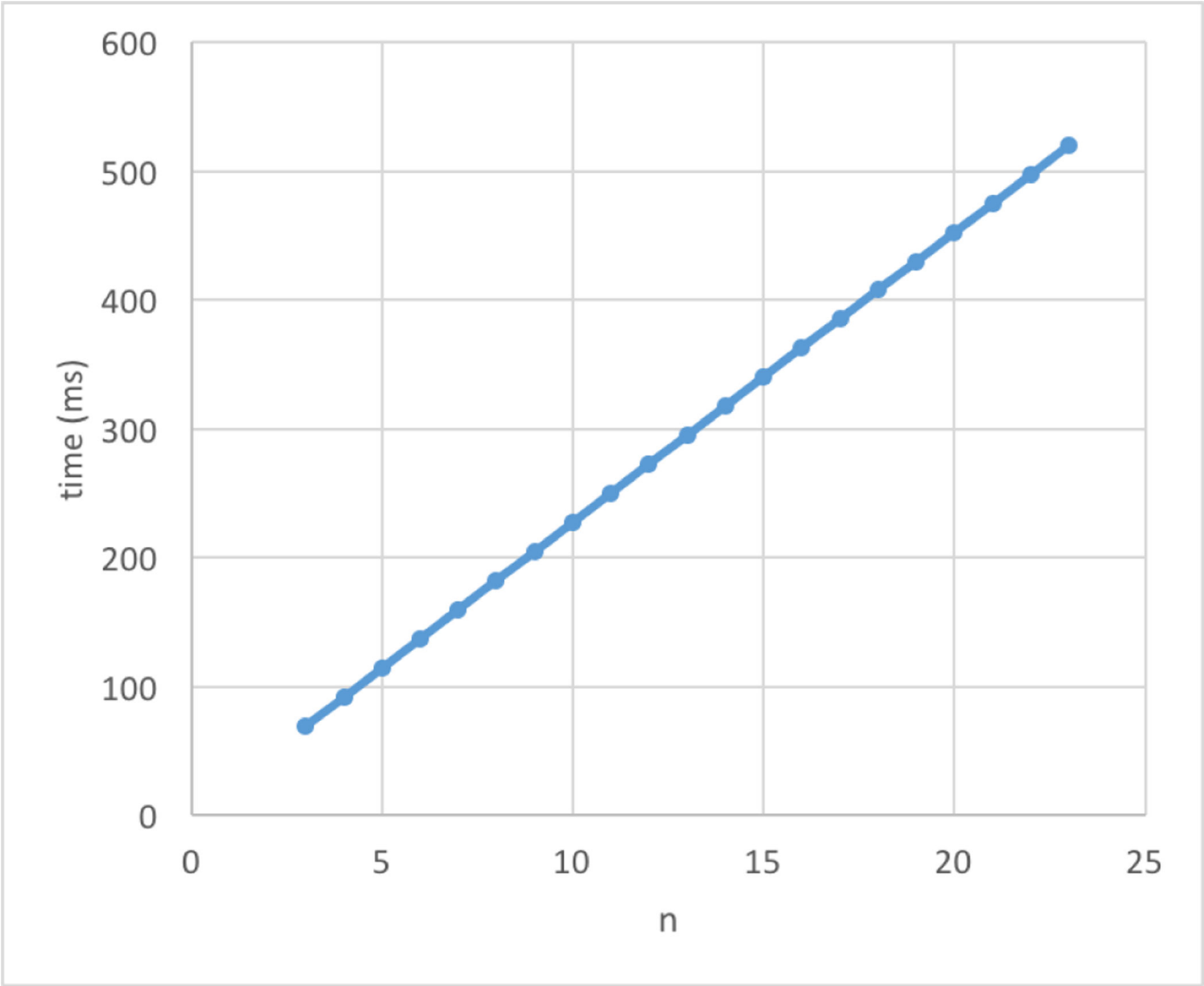


Fig. 10:
Computational overhead for varying number of parties (n), while $k = 1$

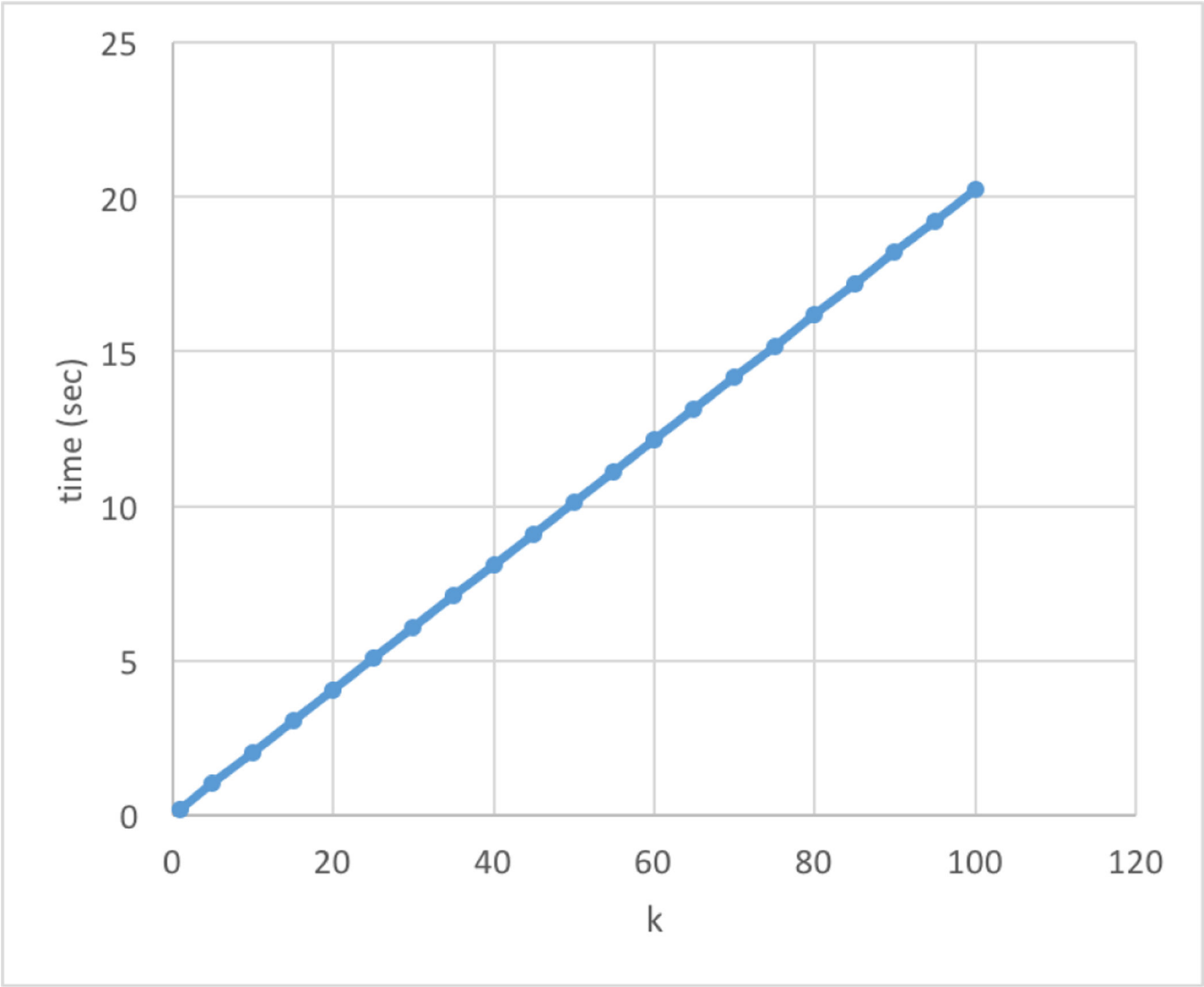


Fig. 11:
Computational over head for varying k, while $n = 10$