Incentive Compatible Privacy-Preserving Data Analysis

Murat Kantarcioglu, U. of Texas at Dallas, muratk@utdallas.edu
Wei Jiang, Purdue University, wjiang@cs.purdue.edu

***Abstract*— In many cases, competing parties who have private data may collaboratively conduct privacy-preserving distributed data analysis (PPDA) tasks to learn beneficial data models or analysis results. For example, different credit card companies may try to build better models for credit card fraud detection through PPDA tasks. Similarly, competing companies in the same industry may try to combine their sales data to build models that may predict the future sales. In many of these cases, the competing parties have different incentives. Although certain PPDA techniques guarantee that nothing other than the final analysis result is revealed, it is impossible to verify whether or not participating parties are truthful about their private input data. In other words, unless proper incentives are set, even current PPDA techniques cannot prevent participating parties from modifying their private inputs. This raises the question of how to design incentive compatible privacy-preserving data analysis techniques that motivate participating parties to provide truthful input data. In this paper, we first develop key theorems, then base on these theorem, we analyze what types of privacy-preserving data analysis tasks could be conducted in a way that telling the truth is the best choice for any participating party.**

***Index Terms*— Privacy, Secure multi-party computation, Non-cooperative computation.**

I. Introduction

Privacy and security, particularly maintaining confidentiality of data, have become a challenging issue with advances in information and communication technology. The ability to communicate and share data has many benefits, and the idea of an omniscient data source carries great value to research and building accurate data analysis models. For example, for credit card companies to build more comprehensive and accurate fraud detection system, credit card transaction data from various companies may be needed to generate better data analysis models. Department of Energy supports research on building much more efficient diesel engines [5]. Such an ambitious task requires the collaboration of geographically distributed industries, national laboratories and universities. Those institutions (including the potentially competing industry partners) need to share their private data for building data analysis models that enable them to understand the underlying physical phenomena. Similarly, different pharmaceutical companies may want to combine their private research data to predict the effectiveness of some protein families on certain diseases.

On the other hand, an omniscient data source eases misuse, such as the growing problem of identity theft. To prevent misuse of data, there is a recent surge in laws mandating protection of confidential data, such as the European Community privacy standards [7], U.S. health-care laws [12], and California SB1386. However, this protection comes with a real cost through both added security expenditure and penalties and costs associated with disclosure. For example, CardSystems was terminated by Visa and American Express after having credit card information stolen. ChoicePoint stock lost 20% of its value in the month following their disclosure of information theft. Such public relations costs can be enormous and could potentially kill a company. From lessons learned in practice, what we need is the ability to compute the desired "beneficial outcome" of data sharing for analyzing without having to actually share or disclose data. This would maintain the security provided by separation of control while still obtaining the benefits of a global data source.

Secure multi-party computation (SMC) [9], [24], [25] has recently emerged as an answer to this problem. Informally, if a protocol meets the SMC definitions, the participating parties learn only the final result and whatever can be inferred from the final result and their own inputs. A simple example is Yao's millionaire problem [24]: two millionaires, Alice and Bob, want to learn who is richer without disclosing their actual wealth to each other. Recognizing this, the research community has developed many SMC protocols, for applications as diverse as forecasting [3], decision tree analysis [17] and auctions [19] among others.

Nevertheless, the SMC model does not guarantee that data provided by participating parties are truthful. In many real life situations, data needed for building data

analysis models are distributed among multiple parties with potentially conflicting interests. For instance, a credit card company that has a superior data analysis model for fighting credit card fraud may increase its profits as compared to its peers. An engine design company may want to exclusively learn the data analysis models that may enable it to build much more efficient diesel engines. An exclusive use of better data analysis models for predicting drug effectiveness may reduce the drug development time for a pharmaceutical company, which in return may translate into a huge competitive advantage. Clearly, as described above, building data analysis models is generally performed among parties that have conflicting interests.

In the SMC model, we generally assume that participating parties provide truthful inputs. This assumption is usually justified by the fact that learning the correct data analysis models or results is in the best interest of all participating parties. Since SMC-based protocols require participating parties to perform expensive computations, if any party does not want to learn data models and analysis results, the party should not participate in the protocol. Still, this assumption does not guarantee the truthfulness of the private input data when participating parties want to learn the final result exclusively. For example, a drug company may lie about its private data so that it can exclusively learn the data analysis model. Although SMC protocols guarantee that nothing other than the final data analysis result is revealed, it is impossible to verify whether or not participating parties are truthful about their private input data. In other words, unless proper incentives are set, current SMC techniques cannot prevent input modification by participating parties.

In order to better illustrate this problem, we consider a case from management where competing companies (e.g., Texas Instruments, IBM and Intel) establish a consortium (e.g., Semiconductor Manufacturing Technology[1]). The companies send the consortium their sales data, and key manufacturing costs and times. Then the consortium analyzes the data and statistically summarizes them in a report of industry trends, which is made available back to consortium members. In this case, it is in the interest of companies to learn *true* industry trends while revealing their private data as little as possible. Even though SMC protocols can prevent the revelation of the private data, they do not guarantee that companies send their true sales data and other required information. Assume that n companies would like to learn the sample mean and variance of the sales data for a particular type of product.

Example 1.1: Let x_i be the ith company's sales amount. In order to estimate the sample mean, companies need to calculate $\mu = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$ and similarly $s^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \mu)^2$ for sample variance. Any company may *exclusively* learn the *correct* result by lying about its input. Company i may report x_i' instead of the correct x_i. Given the wrong mean μ' and variance s'^2 (computed based on x_i' and truthful values from the other parties), the company i can calculate the *correct* sample mean μ by setting:

$$\mu = \mu' + \frac{x_i - x_i'}{n}$$

Similarly, it can calculate the *correct* sample variance s^2 by calculating:

$$s^2 = s'^2 + \frac{x_i^2 - x_i'^2}{n-1} + \frac{n\left(\mu'^2 - \mu^2\right)}{n-1}$$

□

As illustrated above, any company may have the incentive to lie about its input in order to learn the result exclusively, and at the same time, the correct result (e.g., μ) can be computed from its original input, modified input and the incorrect final result (e.g., x_i, x_i' and μ'). If this situation always occurred, no company would have the incentive to be truthful. Fortunately, the intrinsic nature of a function determines whether or not the situation (demonstrated by the above example) could occur.

A. Our Contributions

In this paper, we analyze what types of distributed functionalities could be implemented in an incentive compatible fashion. In other words, we explore which functionalities can be implemented in a way that participating parties have the incentive to provide their true private inputs upon engaging in the corresponding SMC protocols. We show how tools from theoretical computer science in general and non-cooperative computation [21] in particular could be used to analyze incentive issues in distributed data analysis framework. This is significant because input modification *cannot* be prevented *before* the execution of any SMC-based protocol. (Input modification could be prevented *during* the execution of some SMC-based protocols, but these protocols are generally very expensive and impractical.) The theorems developed in the paper can be adopted to analyze whether or not input modification could occur for computing a distributed functionality. If the answer is positive, then there is no need to design complicated and generally inefficient SMC-based protocols.

[1] www.sematech.org

NCC	Non-Cooperative Computation
DNCC	Deterministic Non-Cooperative Computation
PPDA	Privacy Preserving (Distributed) Data Analysis
SMC	Secure Multi-party Computation
TTP	Trusted Third Party

TABLE I
NOTATIONS AND TERMINOLOGIES

In this paper, we assume that the number of malicious or dishonest participating parties can be at most $n-1$, where n is the number of parties. This assumption is very general since most existing works in the area of privacy-preserving data analysis assume either all participating parties are honest (or semi-honest) or the majority of participating parties are honest. Thus, we extend the non-cooperative computation definitions to incorporate cases where there are multiple dishonest parties. In addition, we show that from incentive compatibility point of view, most data analysis tasks need to be analyzed only for two party cases. Furthermore, to show the applicability of our developed theorems, we use these theorems to analyze under what conditions, common data analysis tasks, such as mean and covariance matrix estimation, can be executed in an incentive compatible manner.

The paper is organized as follows: Section II provides an overview of the works closely related to this paper and background regarding the concept of non-cooperative computation. In Section III, we propose several important theorems along with formal proofs. Based on these theorems, Section IV analyzes some common distributed data analysis tasks that are either incentive compatible or not incentive compatible in the context of this paper. Section V concludes the paper with a discussion of possible future research directions.

II. RELATED WORK & BACKGROUND

In this section, we begin with an overview of privacy-preserving distributed data analysis. Then we briefly discuss the concept of non-cooperative computation. Table I provides common notations and terminologies used extensively for the rest of this paper. In addition, the terms *secure* and *privacy-preserving* are interchangeable thereafter.

A. Privacy-Preserving Data Analysis

Many privacy-preserving data analysis protocols have been designed using cryptographic techniques. Data are generally assumed to be either vertically or horizontally partitioned. (Table II shows a trivial example of different data partitioning schemes.) In the case of horizontally partitioned data, different sites collect the same set of information about different entities. For example, different credit card companies may collect credit card transactions of different individuals. Privacy-preserving distributed protocols have been developed for horizontally partitioned data for building decision trees, [16], mining association rules, [14], and generate k-means clusters [15] and k-nn classifiers. (See [23] for a survey of the recent results.)

In the case of vertically partitioned data, we assume that different sites collect information about the same set of entities, but they collect different feature sets. For example, both a university pay roll and the university's student health center may collect information about a student. Again, privacy-preserving protocols for the vertically partitioned case have been developed for mining association rules, [22], building decision trees [6] and k-means clusters [13]. (See [23] for a survey of the recent results.)

To the best of our knowledge, all the previous privacy-preserving data analysis protocols assume that participating parties are truthful about their private input data. Recently, game theoretical techniques have been used to force parties to submit their true inputs [2]. The techniques developed in [2] assume that each party has an internal device that can verify whether they are telling the truth or not. In our work, we do not assume the existence of such a device. Instead, we try to make sure that providing the true input is the best choice for a participating party.

B. Non-Cooperative Computation

Recently, research issues at the intersection of computer science and game theory have been studied extensively. Among those research issues, algorithmic mechanism design and non-cooperative computation are closely related to our work.

The field of algorithmic mechanism design tries to explore how private preferences of many parties could be combined to find a global and socially optimal solution [20]. Usually in algorithmic mechanism design, there exists a function that needs to be maximized based on the private inputs of the parties, and the goal is to devise mechanisms and payment schemes that force individuals to tell their true private values. In our case, since it is hard to measure the monetary value of the data analysis results, devising a payment scheme that is required by many mechanism design models is not viable (e.g., Vickrey-Groves-Clarke mechanisms [20]). Instead, we adopt the non-cooperative computation model [21]

(a) Original dataset

Tr#	a	b	c	d	e
X_1	1	0	1	0	1
X_2	1	0	1	0	0
X_3	0	0	0	0	0
X_4	0	1	0	1	1
X_5	1	1	1	1	1
X_6	1	1	0	1	0
X_7	0	0	0	0	0
X_8	1	1	1	1	1
X_9	1	1	1	1	1

(b) Vertically partitioned

Tr#	a	b
X_1	1	0
X_2	1	0
X_3	0	0
X_4	0	1
X_5	1	1
X_6	1	1
X_7	0	0
X_8	1	1
X_9	1	1

Tr#	c	d	e
X_1	1	0	1
X_2	1	0	0
X_3	0	0	0
X_4	0	1	1
X_5	1	1	1
X_6	0	1	0
X_7	0	0	0
X_8	1	1	1
X_9	1	1	1

(c) Horizontally partitioned

Tr#	a	b	c	d	e
X_1	1	0	1	0	1
X_2	1	0	1	0	0
X_3	0	0	0	0	0

Tr#	a	b	c	d	e
X_6	1	1	0	1	0
X_7	0	0	0	0	0
X_8	1	1	1	1	1
X_9	1	1	1	1	1

TABLE II

A BINARY DATASET ALONG WITH DIFFERENT PARTITIONING SCHEMES

that is designed for parties who want to jointly compute the correct function results on their private inputs. Since data analysis algorithms can be seen as a special case, modifying non-cooperative computation model for our purposes is a natural choice.

The non-cooperative computation (NCC) model can be seen as an example of applying game theoretical ideas in the distributed computation setting [21]. In the NCC model, each party participates in a protocol to learn the output of some given function f over the joint inputs of the parties. First, all participating parties send their private inputs securely to a trusted third party (TTP), then TTP computes f and sends back the result to every participating party. The NCC model makes the following assumptions:

1) ***Correctness***: the first priority for every participating party is to learn the *correct* result;
2) ***Exclusiveness***: if possible, every participating party prefers to learn the *correct* result *exclusively*.

In other words, learning the correct result is the most important objective of every party. Other factors such as privacy and voyeurism could be also considered in the NCC setting. We omit such discussion here. Additional details can be found in [18]. In this paper, we use the NCC setting where each party wants to learn the data mining result *correctly*, if possible prefers to learn it *exclusively*. Also, we assume that revealing only the result does not violate privacy.

Under the *correctness* and *exclusiveness* assumptions, the NCC model is formally defined as follows: Given a set of n parties, for a party i, we denote its private input as $v_i \in D_i$, where D_i is the domain of the possible inputs of party i. For simplicity, we assume that all $D_i = D$ for all i. Parties joint input is represented as $v = (v_1, \ldots, v_n)$, where $v \in D^n$. We use v_{-i} to represent $(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots v_n)$, and (v_i, v_{-i}) to denote the reconstruction of v. It is also assumed that the v values are distributed according to some probability function, and the probability of seeing any $v \in D^n$ is always non-zero. In the NCC model, for calculating any n party function $f : D^n \longmapsto R$ with range R, we use the following simple protocol:

1) Each party i sends v'_i (not necessarily the correct private input) to a TTP;
2) The TTP computes $f(v') = f(v'_1, \ldots, v'_n)$ and sends the results back to the participating parties;
3) Each party i computes $f(v)$ based on $f(v')$ received from TTP and v_i.

Considering the above simple protocol does not limit its generality. Under the literature of SMC, the TTP can be replaced such that the required functionality (represented by f) is still computable without violating privacy regarding each participating party's private input [11]. The next definition states the conditions a function needs to satisfy under the NCC model.

Definition 2.1: [21] Let n, f be as above. Then f is deterministically non-cooperatively computable (*DNCC*), if the following holds: For any party i, every strategy (t_i, g_i), and every $v_i \in D$, it is the case that:

- Either $\exists v_{-i} \in D_{-i}$, $g_i(f(t_i(v_i), v_{-i}), v_i) \neq f(v_i, v_{-i})$
- Or $\forall v_{-i} \in D_{-i}$, $f(t_i(v_i), v_{-i}) = f(v_i, v_{-i})$

The above definition simply states what function could be computed in NCC setting deterministically (i.e., computation result is correct with probability one), and no party could correctly compute the correct result once the party lies about his or her inputs in a way that changes the original function result. In other words, if a party i replaces its true input v_i with v'_i and if $f(v'_i, v_{-i}) \neq f(v_i, v_{-i})$, then party i should not be able to calculate the correct $f(v_i, v_{-i})$ from $f(v'_i, v_{-i})$ and v_i. Note that strategy (t_i, g_i) means that the way the

input is modified, denoted by t_i, and the way the output is calculated, denoted by g_i. In Example 1.1, t_i can be considered as choosing a value different from the actual input, and g_i can be considered as the ways the correct μ and s^2 are computed.

In this paper, we mainly focus on the DNCC model instead of considering a probabilistic extension due to the following observations. First, as shown in [21], if the v_i values are independent, then a boolean function[2] is in DNCC if and only if it is probabilistically non-cooperatively computable. Second, even if v_i values are not independent, it is shown that if a boolean function is in DNCC, then it is also probabilistically non-cooperatively computable [21]. Because of these observations, DNCC provides a good basis for understanding incentive issues in privacy preserving data analysis. We leave other possible extensions as a future work. Other functions are still open.

III. Characteristics of DNCC Functions

As discussed previously, the term *incentive compatible* means that participating parties have the incentive or motivation to provide their actual inputs when they compute a functionality. Although SMC-based privacy-preserving data analysis protocols (under the malicious adversary model) can prevent participating parties from modifying their inputs once the protocols are initiated, they cannot prevent the parties from modifying their inputs before the execution. On the other hand, parties are expected to provide their true inputs to correctly evaluate a function that satisfies the NCC model. Therefore, any functionality that satisfies the NCC model is inherently *incentive compatible* under the assumption that participating parties prefer to learn the function result *correctly*, and if possible *exclusively*. Now the question is which functionalities or data analysis tasks satisfy the NCC model. For the rest of the paper, we first develop certain key theorems regarding NCC functions. Based on these theorems, we subsequently analyze functionalities that can be implemented under (or satisfy) the NCC model.

Based on definition 2.1, it is difficult to prove whether or not a function f is in DNCC because the proof needs to consider all possible t_i and g_i pairs. In general, the strategy t_i defines a way to change the input, and the strategy g_i defines a method to reconstruct the actual result based on the true input, modified input and the result computed based on the modified input and other parties' input data.

[2]The distinction between probabilistic and deterministic non-boolean functions is still an open problem in the literature of NCC.

Example 3.1: For instance, according to Example 1.1 in Section I, the strategy t_i can be considered as:

$$t_i(x_i) = x_i + c = x_i'$$

where c could be any real number. In addition, to compute the sample mean, the strategy g_i is defined as:

$$g_i(x_i, x_i', \mu') = \mu' + \frac{x_i - x_i'}{n} = \mu$$

To compute the sample variance, the strategy g_i is defined as:

$$g_i(x_i, x_i', s'^2) = s'^2 + \frac{x_i^2 - x_i'^2}{n-1} + \frac{n\left(\mu'^2 - \mu^2\right)}{n-1} = s^2$$

□

For complex functionalities, it is very difficult to enumerate all possible t_i and g_i pairs. To avoid this issue, we instead develop the following theorem that describes a simpler way to prove that a function is in DNCC:

Theorem 3.1: A function $f : D^n \mapsto R$ is in *DNCC* if for any given $v_i \in D$, for every strategy t_i, it is the case that:

- Either $\exists v_{-i} \in D_{-i}$, $f(t_i(v_i), v_{-i}) \neq f(v_i, v_{-i})$ **and** $\exists y_{-i} \in D_{-i}$, $f(t_i(v_i), v_{-i}) = f(t_i(v_i), y_{-i})$ **and** $f(v_i, v_{-i}) \neq f(v_i, y_{-i})$
- Or $\forall v_{-i} \in D_{-i}$ $f(t_i(v_i), v_{-i}) = f(v_i, v_{-i})$

Before we prove the theorem, we need to emphasize the fact that in DNCC setting, a party only lies if it can always compute the *correct* result from the wrong result (based on its modified input and the other parties' inputs) and its original input. Therefore, for any t_i, the corresponding g_i is deterministic. Using this fact, it can be proved that if f satisfies the conditions of the above theorem, then f is in DNCC.

Proof: Please note that Theorem 3.1 is very similar to Definition 2.1. The main difference is in the case where $f(t_i(v_i), v_{-i}) \neq f(v_i, v_{-i})$. This implies that if we can prove there exists *no* g_i for all v_i such that $g_i(f(t_i(v_i), v_{-i}), v_i) = f(v_i, v_{-i})$ then f satisfies definition 2.1. For the case where $f(t_i(v_i), v_{-i}) \neq f(v_i, v_{-i})$, from theorem 3.1, we know that for any given v_{-i}, there exists $y_{-i} \in D_{-i}$ $f(t_i(v_i), v_{-i}) = f(t_i(v_i), y_{-i})$ **and** $f(v_i, v_{-i}) \neq f(v_i, y_{-i})$. Assume there exits $g_i(f(t_i(v_i), v_{-i}), v_i) = f(v_i, v_{-i})$ for all $v_{-i} \in D_{-i}$. Since g_i is deterministic and $f(t_i(v_i), v_{-i}) = f(t_i(v_i), y_{-i})$, we know that

$$\begin{aligned} g_i(f(t_i(v_i), y_{-i}), v_i) &= g_i(f(t_i(v_i), v_{-i}), v_i) \\ &= f(v_i, v_{-i})) \end{aligned}$$

We can conclude that $g_i(f(t_i(v_i), y_{-i}), v_i) \neq f(v_i, y_{-i})$, because $f(v_i, v_{-i}) \neq f(v_i, y_{-i})$. This contradicts with

our initial assumption that $g_i(f(t_i(v_i), v_{-i}), v_i) = f(v_i, v_{-i}))$ for all v_{-i}. ■

We would like to stress that Theorem 3.1 only states a sufficient condition for a function to be in DNCC. In Section IV, we show how Theorem 3.1 provides guidance for proving some common functionalities that satisfy the NCC model.

A. Collusion Regarding the NCC Model

When evaluating certain privacy-preserving protocols in practice, we need to consider the case where an adversary may control a subset of the parties involved in the protocol. Such an adversary may force the parties it controls to submit wrong inputs. In order to analyze functionalities that are incentive compatible when collusion is possible, the current DNCC model needs to be extended to include the possibility of collusion. In other words, we need to understand that given f is in DNCC, whether or not f is still in DNCC when collusion occurs. To analyze this, we continue to follow the *correctness* and *exclusiveness* assumptions under the NCC model. Based on the assumptions, we next define the DNCC functions for the case where an adversary controls at most t fixed parties.

Definition 3.1: Suppose $t < n$, f is (n, t)-deterministically non-cooperatively computable (or (n, t)-*DNCC*) if the following holds: For any set $S \subset \{1, \ldots, n\}$ (where $|S| \leq t$), every strategy (t_S, g_S), and every $v_S = (v_{i_1}, v_{i_2}, \ldots, v_{i_{|S|}})$ (where $i_j \in S$), it is the case that:

- Either $\exists v_{-S} \in D_{-S}$, $g_S(f(t_S(v_S), v_{-S}), v_S) \neq f(v_S, v_{-S})$
- Or $\forall v_{-S} \in D_{-S}$, $f(t_S(v_S), v_{-S}) = f(v_S, v_{-S})$

Intuitively, the above definition indicates that any adversary that controls at most t parties is not able to *exclusively* learn the correct function result. Clearly, if we can prove that a function is $(n, n-1)$-DNCC, then this implies that an adversary that controls at most $n-1$ parties is not able to exclusively learn the correct result by modifying the inputs.

For many distributed data analysis tasks, we need to compute functions that have a special structure. For example, assuming that data sets are horizontally partitioned, any distributed data mining function $f(d_1, \ldots d_n)$ defined over n databases $d_1, \ldots d_n$ could be rewritten as $f(d_i, w(d_{-i}))$, where $w(d_{-i}) = \bigcup_{j \neq i}(d_j)$. In general, for any function $f(v_1, \ldots v_n)$ which can be rewritten as $f(v_i, w(v_{-i}))$ for any i and some function w (we show in Section IV that this is the case for most of the data analysis algorithms), we can show that f is $(n, n-1)$-DNCC if and only if f is $(2, 1)$-DNCC.

Theorem 3.2: If $f(v_1, v_2, \ldots v_n) = f(v_i, w(v_{-i}))$ for any i, for any v_i and for some function $w : D^{n-1} \mapsto D$ then f is $(n, n-1)$-*DNCC* for any n if and only if f is $(2, 1)$-*DNCC*.

Proof: If f is $(n, n-1)$-DNCC for any n, f is $(2, 1)$-DNCC (by setting $n = 2$). Next we need to show that if f is not $(n, n-1)$-DNCC for some n (say $n = 3$) then f is not $(2, 1)$-DNCC. Suppose f is not $(n, n-1)$-DNCC. Then according to Definition 3.1, $\exists S \subset \{1, \ldots, n\}$, $\exists (t_S, g_S)$ and $\exists v_S$ such that the following holds simultaneously:

- $\forall v_{-S}$: $g_S(f(t_S(v_S), v_{-S}), v_S) = f(v_S, v_{-S})$
- $\exists v_{-S}$: $f(t_S(v_S), v_{-S}) \neq f(v_S, v_{-S})$

Using such t_S, g_S, we can define a cheating strategy for $(2, 1)$-DNCC case. First set $v_i = w(v_S)$, and define a strategy $t_i(v_i)$ for the two party case as follows: Since v_i is equal to $w(v_S)$, set $t_i(v_i) = w(t_S(v_S))$. Also we define the $g_i(f(t_i(v_i), v_{-i}), v_i)$ as $g_i(f(t_i(v_i), v_{-i}), v_i) = g_S(f(t_i(v_i), v_{-i}), v_S)$. The above strategy works if t_S, g_S exist because (without loss of generality, assume $|S| = n-1$):

$$\begin{aligned} g_S(f(t_i(v_i), v_{-i}), v_S) &= g_S(f(w(t_S(v_S)), v_{-i}), v_S) \\ &= g_S(f(t_S(v_S), v_{-i}), v_S) \\ &= f(v_S, v_{-i}) \\ &= f(w(v_S), v_{-i}) \\ &= f(v_i, v_{-i}) \end{aligned}$$

Note that in the above case $v_{-S} = v_{-i}$. Also we know that $f(t_S(v_S), v_{-S}) \neq f(v_S, v_{-S})$ for some v_{-S}. This implies that for some v_{-i} and $v_i = w(v_S)$, $f(w(t_S(v_S)), v_{-i}) \neq f(v_i, v_{-i})$. ■

The effects of an adversary that controls multiple parties have been studied in SMC domain extensively. The general results indicate that any function could be evaluated privately (i.e., nothing other than the function result is revealed) if an adversary is computationally bounded and does not control the majority of the parties (i.e., an adversary controls at most $\lfloor \frac{n-1}{2} \rfloor$ [10]). This result is still valid if the adversary is rational [11]. Using the ideas from [11], we can easily show that every function in DNCC has a $\lfloor \frac{n-1}{2} \rfloor$ private evaluation without requiring a trusted third party.

IV. Analyzing Data Analysis Tasks in the NCC Model

So far, we have developed techniques to prove whether or not a function is in DNCC. Combining the two concepts DNCC and SMC, we can analyze privacy-preserving data analysis tasks (without utilizing a TTP) that are incentive compatible. We next prove several

such important tasks (as function with boolean output, set operations, linear functions, etc) that either satisfy or do not satisfy the DNCC model. Also, note that the data analysis tasks analyzed next have practical SMC-implementations.

A. Function with Boolean Output

From SMC literature, we know that there are few functions that can be evaluated if the adversary controls $n-1$ parties. Here, we prove that functions with boolean outputs that are $n-1$ private are not in DNCC.

Theorem 4.1: [4] A function from $f : D_1 \times D_2 \times \cdots \times D_n \mapsto \{0,1\}$ is $n-1$-private if there exits a protocol f so that no coalition of size $\leq n-1$ can infer any additional information from the execution, other than the function result. Further more, f is $n-1$ private if and only if it can be represented as:

$$f(v_1, v_2, \ldots, v_n) = f_1(v_1) \oplus f_2(v_2) \oplus \cdots \oplus f_n(v_n)$$

where f_is are arbitrary functions with boolean outputs and $\oplus$ is the binary XOR operation.

Theorem 4.2: There does not exit any non-constant $n-1$ private function with boolean output that is in *DNCC*.

Proof: According to Theorem 4.1, we know that any $n-1$ private function is of the form: $f(v_1, v_2, \ldots, v_n) = f_1(v_1) \oplus f_2(v_2) \oplus \cdots \oplus f_n(v_n)$. Clearly, for any t_i, we can define the g_i as:

$$g_i(f(t_i(v_i), v_{-i}), v_i) = f((t_i(v_i), v_{-i})) \oplus f_i(t_i(v_i)) \oplus f_i(v_i)$$

Note that g_i function will always give the correct result for all possible v_{-i} because:

$$\begin{aligned} g_i(f(t_i(v_i), v_{-i}), v_i) &= f((t_i(v_i), v_{-i})) \oplus f_i(t_i(v_i)) \\ &\quad \oplus f_i(v_i) \\ &= (\oplus_{j \neq i} f_j(v_j)) \oplus f_i(t_i(v_i)) \\ &\quad \oplus f_i(t_i(v_i)) \oplus f_i(v_i) \\ &= (\oplus_{j \neq i} f_j(v_j)) \oplus f_i(v_i) \\ &= f(v_i, v_{-i}) \end{aligned}$$

To complete the proof that f is not in DNCC, we also need to show that there exists v_{-i} such that $f(t_i(v_i), v_{-i}) \neq f(v_i, v_{-i})$. Clearly, if f is a non-constant function, there exits v_i, v_i' for some i and for some v_{-i} such that $f(v_i, v_{-i}) \neq f(v_i', v_{-i})$. Then we can define $t_i(v_i) = v_i'$. ■

B. Set Operations

Set operations are commonly used in privacy-preserving data analysis protocols. (See [14], [22] for examples). Here, we show that common set operations like intersection and union are not in DNCC. Let us assume that each party i has a set S_i, a subset of publicly known universal set U. For example, assuming that each party is a financial institution, S_i could be a set of customers' social security numbers of party i and U could be the set of all 9 digit numbers. As before, let S_{-i} denote $(S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n)$.

Theorem 4.3: Let $f(S_1, \ldots, S_n) = S_1 \cup \cdots \cup S_n$ be the union of sets $S_1, \ldots, S_n$. Then f is not in $(n,1)$-*DNCC*.

Proof: In order to prove that f is not in $(n,1)$-DNCC, we need to provide a correct (t_i, g_i) pair that works on S_i and S_{-i}. We also need to prove that t_i prevents the correct function evaluation for some S_{-i} (i.e., $\exists S_{-i}$ such that $f(t_i(S_i), S_{-i}) \neq f(S_i, S_{-i})$). We can define $t_i(S_i) = S_i \setminus S'$ where S' is any non-empty subset of S_i. The corresponding g_i could be defined as:

$$\begin{aligned} g_i(f(t_i(S_i), S_{-i}), S_i) &= f(t_i(S_i), S_{-i}) \cup S' \\ &= (\cup_{j \neq i} S_j) \cup (S_i \setminus S') \cup S' \\ &= S_1 \cup \cdots \cup S_n = f(S_i, S_{-i}) \end{aligned}$$

The above g_i works because $S' \subseteq S_i$ implies $S_i = (S_i \setminus S') \cup S'$. To conclude the proof, we need to show that there exists S_{-i} such that $f(t_i(S_i), S_{-i}) \neq f(S_i, S_{-i})$. Note that for any S' and S_{-i} where $S' \cap (\cup_{j \neq i} S_j) = \emptyset$, we know that

$$\begin{aligned} f(t_i(S_i), S_{-i}) &= (\cup_{j \neq i} S_j) \cup (S_i \setminus S') \\ &= f(S_i, S_{-i}) \setminus S' \end{aligned}$$

This implies that $f(t_i(S_i), S_{-i}) \neq f(S_i, S_{-i})$. ■

Theorem 4.4: Let $f(S_1, \ldots, S_n) = S_1 \cap \cdots \cap S_n$ be the intersection of sets $S_1, \ldots, S_n$. Then f is not in $(n,1)$-*DNCC*.

Proof: We need to define (t_i, g_i) that works for a chosen S_i and show that there exists S_{-i} for any S_i such that $f(t_i(S_i), S_{-i}) \neq f(S_i, S_{-i})$. Let $S' \subseteq U$ and $S_i \cap S' = \emptyset$. Define t_i as: $t_i(S_i) = S_i \cup S'$. Then we can define g_i as:

$$\begin{aligned} g_i(f(t_i(S_i), S_{-i}), S_i) &= f(t_i(S_i), S_{-i}) \setminus S' \\ &= (\cap_{j \neq i} S_j) \cap (S_i \cup S') \setminus S' \\ &= S_1 \cap \cdots \cap S_n \\ &= f(S_i, S_{-i}) \end{aligned}$$

The g_i works correctly because $S_i \cap S' = \emptyset$ and $(S_i \cup S') \setminus S' = S_i$. Also we need to show that for any S_i,

there exists S_{-i} such that $f(t_i(S_i), S_{-i}) \neq f(S_i, S_{-i})$. Let us assume $S' \cap (\cap_{j\neq i} S_j) \neq \emptyset$. In that case,

$$\begin{aligned} f(t_i(S_i), S_{-i}) &= (\cap_{j\neq i} S_j) \cap (S_i \cup S') \\ &= f(S_i, S_{-i}) \cup ((\cap_{j\neq i} S_j) \cap S') \end{aligned}$$

Since $S' \cap (\cap_{j\neq i} S_j) \neq \emptyset$, we can conclude that $f(t_i(S_i), S_{-i}) \neq f(S_i, S_{-i})$. ■

C. Multivariate Statistics

In many distributed data mining tasks, we may need to learn the mean and the covariance of the underlying data set. For example, to build a mixture of Gaussian model for classification, we may want to learn the mean and the covariance matrix of each class [8]. Let us assume that $X_1, X_2, \ldots X_N$ are identically and independently distributed $1 \times p$ row vectors (or a tuple in a dataset) where $E(X_k) = \mu$ and $\Sigma = E((X_k - \mu)^T (X_k - \mu))$. Let X_k^v denotes the vth column of the row vector X_k. We can estimate μ and Σ as $\bar{\mu}$ and $\bar{\Sigma}$:

$$\begin{aligned} \bar{\mu} &= \frac{1}{N}\sum_{k=1}^{N} X_k \\ \bar{\Sigma} &= \frac{1}{N}\sum_{k=1}^{N} (X_k - \bar{\mu})^T (X_k - \bar{\mu}) \end{aligned}$$

In our analysis, we consider two different data partition cases: the horizontally partitioned data and vertically partitioned data. In the case of horizontally partitioned data, each party i owns a set S_i where $S_i \cap S_j = \emptyset$ for any i and j, and $\cup_{i=1}^{n} S_i = \{X_1, \ldots, X_N\}$. In the case of vertically partitioned data, for all k, each party i owns all $X_k^{v_1}, \ldots, X_k^{v_i}$, where $1 \leq k \leq N$ and $\{v_1, \ldots, v_i\} \subset \{1, \ldots, p\}$ (p indicates the number of attributes or columns).

Example 4.1: Table II shows an example of different data partitioning schemes. Table II (a) represents a binary dataset, and Table II (b) shows a vertical partition between two parties, where columns a and b belong to one party and columns c, d and e belong to another party. Table II (c) shows a horizontal partition between two parties, where records X_1, X_2 and X_3 belong to one party and records X_6, X_7, X_8 and X_9 belong to another party.

Next we discuss whether $\bar{\mu}$ and $\bar{\Sigma}$ could be evaluated in the DNCC model on horizontally partitioned and vertically partitioned data.

1) Horizontally Partitioned Data: For the horizontally partitioned case, we will show that if the total number of vectors are private (i.e., N is private), then the functions computing $\bar{\mu}$ and $\bar{\Sigma}$ are in $(2, 1)$-DNCC. If N is a public information, then the the functions computing $\bar{\mu}$ and $\bar{\Sigma}$ are not in $(n, 1)$-DNCC. Consider Example 1.1, in this example, the total number of vectors is equal to the total number of parties. Since the total number of parties is a public information, the functions computing the mean and the variance are not in DNCC.

To prove that the functions computing $\bar{\mu}$ and $\bar{\Sigma}$ are not in DNCC when N is public, we first prove that any linear function is not in DNCC, and later on, we will show that functions computing $\bar{\mu}$ and $\bar{\Sigma}$ are linear functions if N is public.

Theorem 4.5: Any linear function $f(v_1, v_2, \ldots v_n) = f_1(v_1) + f_2(v_2) + \cdots + f_n(v_n)$ is not in $(n, 1)$-*DNCC*.

Proof: For any t_i we can define the g_i as follows:

$$g_i(f(t_i(v_i), v_{-i}), v_i) = f(t_i(v_i), v_{-i}) - f_i(t_i(v_i)) + f_i(v_i)$$

Note that g_i function will always give the correct result for all possible v_{-i} because

$$\begin{aligned} g_i(f(t_i(v_i), v_{-i}), v_i) &= f(t_i(v_i), v_{-i}) - f_i(t_i(v_i)) \\ &\quad + f_i(v_i) \\ &= \sum_{j\neq i} f_j(v_j) + f_i(t_i(v_i)) \\ &\quad - f_i(t_i(v_i)) + f_i(v_i) \\ &= f(v_i, v_{-i}) \end{aligned}$$

To conclude the proof, we need to show that there exists v_{-i} such that $f(t_i(v_i), v_{-i}) \neq f(v_i, v_{-i})$. Clearly, if f is a non-constant function, there exits v_i, v_i' for some i and for some v_{-i} such that $f(v_i, v_{-i}) \neq f(v_i', v_{-i})$. Then we can define $t_i(v_i) = v_i'$. ■

Using Theorem 4.5, we can easily prove that functions computing $\bar{\mu}$ and $\bar{\Sigma}$ are not in DNCC.

Theorem 4.6: If N is public, then the functions computing $\bar{\mu}$ and $\bar{\Sigma}$ on the horizontally partitioned data are not in $(n, 1)$-*DNCC*.

Proof: Each party i can calculate $Y_i = \frac{1}{N}\sum_{X_j \in S_i} X_j$ locally. It is easy to see that $\bar{\mu}$ is a linear function of Y_i values (i.e., $\bar{\mu} = \sum_{i=1}^{n} Y_i$). Therefore, function computing $\bar{\mu}$ is not in $(n, 1)$-DNCC due to Theorem 4.5.

Similar argument is valid for the function computing $\bar{\Sigma}$. This time, each party i can compute the $Y_i = \frac{1}{N}\sum_{X_j \in S_i} (X_j - \bar{\mu})^T (X_j - \bar{\mu})$ locally. Similarly, $\bar{\Sigma}$ is a linear function of Y_i values (i.e., $\bar{\Sigma} = \sum_{i=1}^{n} Y_i$). Therefore, function computing $\bar{\Sigma}$ is not in $(n, 1)$-DNCC due to Theorem 4.5. ■

If we assume that $|S_i|$ values are private (i.e., the total number of vectors denoted by N is a private information), then we can prove that functions computing $\bar{\mu}$ and $\bar{\Sigma}$ on the horizontally partitioned data are in $(n, n-1)$-DNCC. To prove the above statement, we first prove that

$f : (\mathcal{R}, \mathcal{Z}^+)^n \mapsto R$ defined as below is in $(n, n-1)$-DNCC:

$$f((Y_1, C_1), \ldots, (Y_n, C_n)) = \frac{\sum_i^n Y_i}{\sum_i^n C_i}$$

Note that if we set $Y_i = \sum_{X_j \in S_i} X_j$ and $C_i = |S_i|$, then $\bar{\mu} = f((Y_1, C_1), \ldots, (Y_n, C_n))$. Likely, if we set $Y_i = \sum_{X_j \in S_i} (X_j - \bar{\mu})^T (X_j - \bar{\mu})$, then we can compute $\bar{\Sigma} = f((Y_1, C_1), \ldots, (Y_n, C_n))$.

To prove that $f((Y_1, C_1), \ldots, (Y_n, C_n))$ is in $(n, n-1)$-DNCC, we use Theorem 3.2. Let $v_i = (Y_i, C_i)$. First note that f satisfies the requirements of Theorem 3.2, due to the fact that $f(v_i, v_{-i}) = f(v_i, w(v_{-i}))$ for $w(v_{-i}) = (\sum_{j \neq i} Y_j, \sum_{j \neq i} C_j)$. Therefore, showing that f is in $(2, 1)$-DNCC will automatically imply that f is in $(n, n-1)$-DNCC. Below, we prove that f is in $(2, 1)$-DNCC.

Theorem 4.7: Let Y_i be a real number ($Y_i \in \mathcal{R}$) and let C_i a positive integer ($C_i \in \mathcal{Z}^+$). Then $f((Y_1, C_1), (Y_2, C_2)) = (Y_1 + Y_2)/(C_1 + C_2)$ is in $(2, 1)$-*DNCC*.

Proof: If we can show that the conditions stated in Theorem 3.1 holds for any t_i, we can conclude that f is in $(2, 1)$-DNCC. Assume that party 1 (without loss of generality) uses a t_1 to modify its Y_1 and C_1 input. Let Y_1' and C_1' be those modified inputs. Now consider the case where $f(t_1(Y_1, C_1), (Y_2, C_2)) = u'$, $f((Y_1, C_1), (Y_2, C_2)) = u$, and $u' \neq u$ for some u' and u. Note that $f(t_1(Y_1, C_1), (Y_2, C_2)) = f(t_1(Y_1, C_1), (Y_2 + ku', C_2 + k)) = u'$ for any positive integer k. This implies that given $t_1(v_1) = (Y_1', C_1')$, we can find $v_2 = (Y_2, C_2)$ and $y_2 = (Y_2 + ku', C_2 + k))$ that satisfies the $f(t_i(v_i), v_{-i}) = f(t_i(v_i), y_{-i})$ requirement of Theorem 3.1. Next we consider whether or not $f(v_i, v_{-i}) \neq f(v_i, y_{-i})$ requirement is satisfied. Without loss of generality and assuming $k = 1$, $f(v_1, v_2) = f(v_1, y_2)$ iff $(Y_1 + Y_2)/(C_1 + C_2) = (Y_1 + Y_2 + u')/(C_1 + (C_2 + 1))$. This implies that $f(v_1, v_2) = f(v_1, y_2)$ iff $u' = (Y_1 + Y_2)/(C_1 + C_2) = u$. This contradicts the initial assumption that $u' \neq u$. Therefore, we can conclude that $f(v_1, v_2) \neq f(v_1, y_2)$. ■

Although, in the above proof, we assumed that Y_i is a real number, the above proof could be directly applied if $Y_i \in \mathcal{R}^{p \times p}$.

2) Vertically Partitioned Data: In the case of vertically partitioned data, each party i owns all $X_k^{v_1}, \ldots, X_k^{v_i}$, where $1 \leq k \leq N$ and $\{v_1, \ldots, v_i\} \subset \{1, \ldots, p\}$. Note that the vth column of the $\bar{\mu}$ can be calculated by the party who owns all X_k^v values, for $1 \leq k \leq N$ (i.e., $\bar{\mu}^v = \frac{1}{N} \sum_{k=1}^N X_k^v$. Thus, in order to estimate $\bar{\mu}$, each party could calculate the $\bar{\mu}^v$ for v values it owns and announce the results. Clearly, each party can lie about the $\bar{\mu}^v$ value as in the example given in Section I. Therefore, for the vertically partitioned data case, the function computing $\bar{\mu}$ is not in $(n, 1)$-DNCC.

Also, we can show that computing $\bar{\Sigma}$ is not in $(n, 1)$-DNCC. First note that $\bar{\Sigma}_{uv}$ could be computed as

$$\bar{\Sigma}_{uv} = \frac{1}{N} \sum_{k=1}^{N} (X_k^u - \bar{\mu}^u)(X_k^v - \bar{\mu}^v)$$

If party i knows all the X_k^u and X_k^v values, it can calculate the $\bar{\Sigma}_{uv}$ locally. If party i knows all the X_k^u values and party j knows all the X_k^v values, they may jointly compute $\bar{\Sigma}_{uv}$. Unfortunately, even in that case, the function that computes $\bar{\Sigma}_{uv}$ is not in $(2, 1)$-DNCC.

Since computing $\bar{\Sigma}_{uv}$ can be seen as a dot product of two vectors, we can show that computing $\bar{\Sigma}_{uv}$ is not in $(2, 1)$-DNCC by showing that computing the dot product of vectors of real numbers are not in $(2, 1)$-DNCC. As before, we show that computing the dot product of vectors of real numbers are not in $(2, 1)$-DNCC by specifying t_i and g_i functions.

Theorem 4.8: Let $f(Y_1, Y_2) = \sum_{k=1}^N (Y_1^k \cdot Y_2^k)$ where Y_i is a $1 \times p$ row vector with real number entries and Y_i^k be the kth column of the Y_i. Then f is not in $(2, 1)$-*DNCC*.

Proof: Without loss of generality and assuming $t_1(Y_i) = \alpha \cdot Y_1$ where α is a non-zero scalar number. Since $f(t_1(Y_1), Y_2) = \alpha \cdot f(Y_1, Y_2)$, we can define g_1 as follows:

$$g_i(f(t_1(Y_1), Y_2), Y_1) = f(t_1(Y_1), Y_2)/\alpha$$

Clearly, (t_i, g_i) works for any non-zero scalar number α. Also, note that $f(Y_1, Y_2) \neq f(t_1(Y_1), Y_2)$ for any $f(Y_1, Y_2) \neq 0$. ■

D. Privacy Preserving Association Rule Mining

In this section, we first summarize the association rule mining and analyze whether the association rule mining can be done in an incentive compatible manner over horizontally and vertically partitioned databases.

1) Overview of Association Rule Mining: The association rules mining problem can be defined as follows [1]: Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items. Let DB be a set of transactions, where each transaction T is an itemset such that $T \subseteq I$. Given an itemset $X \subseteq I$, a transaction T *contains* X if and only if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$ where $X \subseteq I, Y \subseteq I$ and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ has *support* s in the transaction database DB if $s\%$ of transactions in DB contain $X \cup Y$. The association rule holds in the transaction database DB with *confidence* c if $c\%$ of transactions in DB that contain X also contain

Y. An itemset X with k items called k-itemset. The problem of mining association rules is to find all rules whose support and confidence are higher than certain user specified minimum support and confidence.

In this simplified definition of the association rules that we use in this paper, missing items and negative quantities are not considered. In this respect, transaction database DB can be seen as $0/1$ matrix where each column is an item and each row is a transaction.

2) Horizontally Partitioned Data: The above problem of mining association rules can be extended to distributed environments. Let us assume that a transaction database DB is horizontally partitioned among n parties where $DB = DB_1 \cup DB_2 \cup \cdots \cup DB_n$, and DB_i resides at party i's site ($1 \leq i \leq n$). The itemset X has *local* support count of $X.sup_i$ at party i if $X.sup_i$ of the transactions contain X. The *global* support count of X is given as $X.sup = \sum_{i=1}^{n} X.sup_i$. An itemset X is *globally supported* if $X.sup \geq s \cdot \sum_{i=1}^{n} |DB_i|$. Global confidence of a rule $X \Rightarrow Y$ can be given as $\{X \cup Y\}.sup/X.sup$.

The set of large itemsets $L_{(k)}$ consists of all k-itemsets that are globally supported. The aim of distributed association rule mining is to find the sets $L_{(k)}$ for all $k > 1$ and the support counts for these itemsets, and from this, association rules with the specified minimum support and confidence can be computed.

In [14], authors discuss how to convert a fast distributed association rule mining algorithm to a privacy-preserving association rule mining algorithm. As discussed in [14], to enable privacy-preserving version, it is enough to privately check whether or not a candidate large itemset is globally supported. Therefore, for our purposes, it is sufficient to show that checking if a candidate itemset is globally supported is in DNCC. First note that, to check if a candidate itemset X is supported globally, all we need to know is whether $X.sup \geq s \cdot |DB|$. The following allows us to reduce this to a comparison against the sum of local values (the *excess support* at each party):

$$X.sup \quad \geq \quad s * |DB| = s \cdot \sum_{i=1}^{n} |DB_i|$$

$$\sum_{i=1}^{n} (X.sup_i - s \cdot |DB_i|) \quad \geq \quad 0$$

Let Y_i be the local excess support at party i (i.e., $Y_i = (X.sup_i - s \cdot |DB_i|)$). We next show the predicate $f(Y_1, \ldots, Y_n)$: $(\sum_i^n (Y_i) \geq 0)$ is in $(n, n-1)$-DNCC,

Theorem 4.9: Given a candidate itemset X, checking $X.sup \geq s \cdot |DB|$ is in $(n, n-1)$-*DNCC*.

Proof: As noted above checking $X.sup \geq s \cdot |DB|$ is equivalent to evaluate the predicate $f(Y_1, \ldots, Y_n)$: $(\sum_i^n (Y_i) \geq 0)$ for $Y_i = (X.sup_i - s \cdot |DB_i|)$. Note that for $w(Y_{-i}) = \sum_{j \neq i} Y_j$, we get $f(Y_i, Y_{-i}) = f(Y_i, w(Y_{-i}))$. Based on Theorem 3.2, to show f is $(n, n-1)$-DNCC, it is sufficient to show that f is in $(2, 1)$-DNCC. For the two party case, we can define the predicate as $f(Y_1, Y_2)$: $(Y_1 \geq -Y_2)$. As a result, we merely need to show that the comparison function is in $(2, 1)$-DNCC for arbitrary real number inputs.

If we can show that the conditions stated in Theorem 3.1 hold, then we can conclude that f is in $(2, 1)$-DNCC for any t_i. Suppose $v_1 = Y_1$ ($v_2 = -Y_2$) is the true input of the first (second) user. We define $t_1(v_1)$ as the modified input of party 1 and let $t_1(v_1) - v_1 = r$, for some value r. When r is 0, we say t_i is the identity function, and for the identity function, the condition "Or $\forall v_{-i} \in D_{-i}$, $f(t_i(v_i), v_{-i}) = f(v_i, v_{-i})$" holds automatically. For the rest of the proof, we consider the situation where $r \neq 0$. Based on the r value, we consider two cases:

1) Assume $r > 0$, consider the case where $v_1 + r > v_2 > v_1 > y_2$. Note that $f(t_1(v_1), v_2) = f(v_1 + r, v_2) = f(v_1 + r, y_2)$, but $f(v_1, v_2) \neq f(v_1, y_2)$. Also $f(t_1(v_1), v_2) \neq f(v_1, v_2)$. This implies that the first condition of the theorem 3.1 is satisfied.
2) Assume $r < 0$, consider the case where $y_2 > v_1 > v_2 > v_1 + r$. Note that $f(t_1(v_1), v_2) = f(v_1 + r, v_2) = f(v_1 + r, y_2)$, but $f(v_1, v_2) \neq f(v_1, y_2)$. Also $f(t_1(v_1), v_2) \neq f(v_1, v_2)$. This implies that the first condition of the the theorem 3.1 is satisfied.

As a result, we conclude that f is in $(2, 1)$-DNCC. ■

Again as shown in [14], we can use the f described above to check whether the confidence threshold is satisfied for any candidate rule of the form $X \Rightarrow Y$.

3) Vertically Partitioned Data: Given the transaction database DB as $0/1$ matrix, where each column is an item and each row is a transaction, the DB is considered vertically partitioned if different parties know different columns of the DB. To mine association rules over vertically partitioned data, it has been shown that you need to calculate a dot product with $0/1$ vectors, where each vector represents whether a certain set of items are present in a transaction or not [22]. Therefore, if we can show that functions calculating dot product with binary vectors is in DNCC, then using the results from [22], we can conclude that calculating an support count of an itemset is also in DNCC.

Below we show that calculating the dot product of non-zero binary vectors is in $(2, 1)$-DNCC. (In this context, we call a vector, non-zero vector if at least one of the entries is non-zero.) Before we proceed with the proof, we stress that the following theorem is not a

contradiction with our earlier result that shows that the function computing dot product of real valued vectors is not in DNCC. Note that t_i described for the dot product of real valued vectors will no longer work in the context of binary vectors since you can only multiply each entry with zero or one. Another important detail to note is we assume that the binary vectors are non-zero vectors. This assumption is important because if we allow zero vectors, the dot product result could be exactly determined even without any computation by the owner of the zero vector. Thus, the owner of the zero vector could lie about its input easily.

We believe that the assumption of non-zero binary vectors is realistic because with fairly large databases, it is highly likely that there exists at least one transaction that supports the required item.

Theorem 4.10: Let f be denoted as $f(v_1, v_2) = \sum_{j=1}^{k} v_1^j \cdot v_2^j$, where v_i is a non-zero binary vector with k rows and v_i^j is the value of the jth row of vector v_i, then f is in $(2,1)$-*DNCC*.

Proof: Let U be the set of indexes from $\{1, \ldots, k\}$. Let S_1 be any subset of U such that $\forall j \in S_1, v_1^j = 1$. Also we know that $|S_1| > 0$ since v_1 is a non-zero vector. Without loss of generality, we assume that t_1 modifies some subset of the v_1's rows. (Note that any t_1 that transforms v_1 could be represented as the set of indexes that are negated.) Let C be the set of indexes negated by t_1. Suppose $C \neq \emptyset$, and because $C = \emptyset$ implies $t_1(v_1) = v_1$, we can rewrite $f(v_1, v_2)$ as:

$$f(v_1, v_2) = \sum_{j=1}^{k} (v_1^j \cdot v_2^j) = \sum_{j \in S_1} v_2^j$$

Note that all v_1^js, where $j \notin S_1$, are equal to zero, and all v_1^js, where $j \in S_1$, are equal to one. Similarly, given t_1 and the associated set C, we can represent $f(t_1(v_1), v_2)$ as:

$$f(t_1(v_1), v_2) = \sum_{j \in S_1 \setminus C} v_2^j + \sum_{j \in C \setminus S_1} v_2^j$$

Note that all v_1^j values, where $j \in S_1 \cap C$, are converted to zero by t_1. Also the v_1^j values, where $j \in C \setminus S_1$, that are zero are converted to one by t_1. We next show that for any t_1, we can find y_2 and v_2 such that $f(t_1(v_1), v_2) = f(t_1(v_1), y_2)$ and $f(v_1, v_2) \neq f(v_1, y_2)$. In addition, if $f(t_1(v_1), v_2) \neq f(v_1, v_2)$, then the first condition of Theorem 3.1 would be satisfied. Otherwise, it means that $f(t_1(v_1), v_2) = f(v_1, v_2)$ for all v_2. To prove the existence of such y_2 and v_2 for any t_1, we consider two different cases:

Case 1 $S_1 \cap C = \emptyset$: Implies that some zero values in v_1 are converted to one by t_1:

$$f(t_1(v_1), v_2) = \sum_{j \in S_1} v_2^j + \sum_{j \in C} v_2^j$$

Now consider y_2 and v_2 such that $\sum_{j \in S_1} (v_2^j - y_2^j) = 1$ (this is possible because $S_1 \neq \emptyset$) and $\sum_{j \in C} (v_2^j - y_2^j) = -1$. It is easy to see that

$$\begin{aligned} f(t_1(v_1), v_2)) &= \sum_{j \in S_1} v_2^j + \sum_{j \in C} v_2^j \\ &= \sum_{j \in S_1} (y_2^j) + 1 + \sum_{j \in C} (y_2^j) - 1 \\ &= f(t_1(v_1), y_2) \end{aligned}$$

Clearly, $f(v_1, v_2) \neq f(v_1, y_2)$ since $\sum_{j \in S_1} (v_2^j - y_2^j) = 1$.

Case 2 $S_1 \cap C \neq \emptyset$: Now consider any y_2 and v_2 such that $y_2^j = v_2^j$ for $j \notin S \cap C$, and $v_2^j = 1$ and $y_2^j = 0$ for $j \in S \cap C$. Note that $f(t_1(v_1), v_2) = f(t_1(v_1), y_2)$ because v_2^j and y_2^j are exactly the same except $j \in S \cap C$. Also note that $f(v_1, v_2) \neq f(v_1, y_2)$

$$\begin{aligned} \sum_{j \in S_1 \cap C} v_2^j &\neq \sum_{j \in S_1 \cap C} y_2^j \\ \sum_{j \in S_1 \setminus C} v_2^j + \sum_{j \in S_1 \cap C} v_2^j &\neq \sum_{j \in S_1 \setminus C} v_2^j + \sum_{j \in S_1 \cap C} y_2^j \\ f(v_1, v_2) &\neq f(v_1, y_2) \end{aligned}$$

Therefore, we can conclude that f is in $(2,1)$-DNCC. ■

V. Future Work

Even though privacy-preserving data analysis techniques guarantee that nothing other than the final result is disclosed, whether or not participating parties provide truthful input data cannot be verified. In this paper, we have investigated what kinds of PPDA tasks are incentive compatible under the NCC model. Based on our findings, there are several important PPDA tasks that are incentive driven. As a future work, we will investigate incentive issues in other data analysis tasks, and extend the proposed theorems under the probabilistic NCC model.

The PPDA tasks analyzed in the paper can be reduced to evaluation of a single function. Now, the question is how to analyze whether a PPDA task is in DNCC if it is reduced to a set of functions. In other words, is the composition of a set of DNCC functions still in DNCC? We will formally answer this question in the future.

Another important direction that we would like to pursue is to create more efficient SMC techniques tailored towards implementing the data analysis tasks that are in DNCC.

REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *VLDB '94*, pages 487–499, Santiago, Chile, September 12-15 1994. VLDB.

[2] Rakesh Agrawal and Evimaria Terzi. On honesty in sovereign information sharing. In *EDBT*, pages 240–256, 2006.

[3] Mikhail J. Atallah, Marina Bykova, Jiangtao Li, and Mercan Karahan. Private collaborative forecasting and benchmarking. In *Proc. 2d. ACM Workshop on Privacy in the Electronic Society (WPES)*, Washington, DC, October 28 2004.

[4] B. Chor and E. Kushilevitz. A zero-one law for boolean privacy. In *STOC '89*, pages 62–72, New York, NY, USA, 1989. ACM Press.

[5] www.doe.gov, doe news, feb. 16 2005.

[6] Wenliang Du and Zhijun Zhan. Building decision tree classifier on private data. In Chris Clifton and Vladimir Estivill-Castro, editors, *IEEE International Conference on Data Mining Workshop on Privacy, Security, and Data Mining*, volume 14, pages 1–8, Maebashi City, Japan, December 9 2002. Australian Computer Society.

[7] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Communities*, No I.(281):31–50, October 24 1995.

[8] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, CA, 1990.

[9] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In *19th ACM Symposium on the Theory of Computing*, pages 218–229, 1987.

[10] Oded Goldreich. *The Foundations of Cryptography*, volume 2, chapter General Cryptographic Protocols. Cambridge University Press, 2004.

[11] Joseph Halpern and Vanessa Teague. Rational secret sharing and multiparty computation: extended abstract. In *STOC '04*, pages 623–632, New York, NY, USA, 2004. ACM Press.

[12] Standard for privacy of individually identifiable health information. *Federal Register*, 67(157):53181–53273, August 14 2002.

[13] Geetha Jagannathan and Rebecca N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the 2005 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 593–599, Chicago, IL, August 21-24 2005.

[14] Murat Kantarcıoğlu and Chris Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE TKDE*, 16(9):1026–1037, September 2004.

[15] Xiaodong Lin, Chris Clifton, and Michael Zhu. Privacy preserving clustering with distributed EM mixture modeling. *Knowledge and Information Systems*, 8(1):68–81, July 2005.

[16] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Advances in Cryptology – CRYPTO 2000*, pages 36–54. Springer-Verlag, August 20-24 2000.

[17] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.

[18] Robert McGrew, Ryan Porter, and Yoav Shoham. Towards a general theory of non-cooperative computation (extended abstract). In *TARK IX*, 2003.

[19] Moni Naor, Benny Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*. ACM Press, 1999.

[20] Noam Nisan and Amir Ronen. Algorithmic mechanism design (extended abstract). In *STOC' 99*, pages 129–140, New York, NY, USA, 1999. ACM Press.

[21] Yoav Shoham and Moshe Tennenholtz. Non-cooperative computation: boolean functions with correctness and exclusivity. *Theor. Comput. Sci.*, 343(1-2):97–113, 2005.

[22] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *ACM SIGKDD'02*, pages 639–644, Edmonton, Alberta, Canada, July 23-26 2002.

[23] Vassilios S. Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin, and Yannis Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Rec.*, 33(1):50–57, 2004.

[24] Andrew C. Yao. Protocols for secure computation. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 160–164. IEEE, 1982.

[25] Andrew C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167. IEEE, 1986.