International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Volume 3, Issue 3 (May-June 2015), PP. 88-93

ϵ-DIFFERENTIAL PRIVACY MODEL FOR VERTICALLY PARTITIONED DATA TO SECURE THE PRIVATE DATA RELEASE

K.Samhita, M.Sannihitha, G.Sri Sneha Varsha, Y.Rohitha
sannihithamuppidi@gmail.com

Abstract- Protecting the private data from publishing has become a requisite in today's world of data. This solves the problem of divulging the sensitive data when it is mined. Amongst the prevailing privacy models, ϵ-differential privacy outfits one of the strongest privacy guarantees. In this paper, we address the problem of private data publishing on vertically partitioned data, where different attributes exist for same set of individuals. This operation is simulated between two parties. In specific, we present an algorithm with differentially private data release for vertically partitioned data between two parties in the semi-honest adversary model. First step towards achieving this is to present a two party protocol for exponential mechanism. By the same token, a two-party algorithm that releases differentially private data in a secure way following secure multiparty computation is implemented. A set of investigational results on the real-life data indicate that the proposed algorithm can effectively safeguard the information for a data mining task.

Key words: Differential privacy, vertically partitioned data, secure multiparty computation, exponential mechanism, two party algorithm

I. INTRODUCTION

Significant progress in the communication and the storage systems emerged into large data sets. Efficient warehousing and manipulation of these large data sets are of paramount importance today. This made the database a commonplace in all the fields. Each anonymous entity owns a database like student data by an institution, employee data by an organization and medical data by a hospital. Mass appeal of the new paradigms like cloud computing increased the dispensing of data amongst multiple entities. Such data can be integrated to provide a unified view of the data assets, make data more available and to enable better analysis. For instance, enterprise data can be integrated to minimize the inconsistent data, lessen the redundant data and to lessen the interface software. However, none of the participating entities should obtain information more than necessary through data integration. Also, adversaries should not misemploy the new knowledge resulted from the integration of data as this might lead to the divulging of the sensitive information which was not available prior to the data integration. In this paper, we propound an algorithm to securely integrate the data from two parties(two data providers) preserving the personally identifiable sensitive data from publishing, whereby the new data still keep hold of the critical information required in the data mining tasks.

Example:

The National Association of Health Data Organizations (NAHDO) reported that 37 states in the USA have legislative mandates to collect hospital level data and that 17 states have started collecting ambulatory care data from hospitals, physicians offices, clinics, and so forth [2]. The leftmost circle in Figure 1 contains a subset of the fields of information, or attributes, that NAHDO recommends these states collect; these attributes include the patient’s ZIP code, birth date, gender, and ethnicity. In Massachusetts, the Group Insurance Commission (GIC) is responsible for purchasing health insurance for state employees. GIC collected patient-specific data with nearly one hundred attributes per encounter along the lines of the those shown in the leftmost circle of Figure 1 for approximately 135,000 state employees and their families. Because the data were believed to be anonymous, GIC gave a copy of the data to researchers and sold a copy to industry. For twenty dollars I purchased the voter registration list for Cambridge Massachusetts and received the information on two diskettes. The rightmost circle in Figure 1 shows that these data included the name, address, ZIP code, birth date, and gender of each voter. This information can be linked using ZIP code, birth date and gender to the medical information, thereby linking diagnosis, procedures, and medications to particularly named individuals.

For example, William Weld was governor of Massachusetts at that time and his medical records were in the GIC data. Governor Weld lived in Cambridge Massachusetts. According to the Cambridge Voter list, six people had his particular birth date; only three of them were men; and, he was the only one in his 5-digit ZIP code.

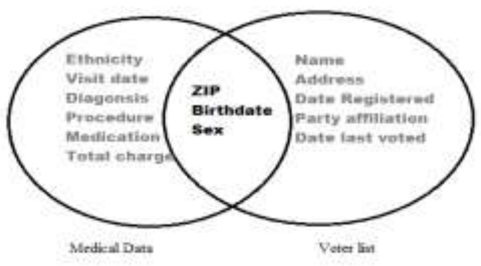

Figure 1 Linking to re-identify data

The example above provides a demonstration of re-identification by directly linking (or “matching”) on shared attributes. The work presented in this paper shows that altering To prevent such linking attacks, Jiang and Clifton and Mohammed et al. have proposed algorithms that enable two parties to integrate their data satisfying the k-anonymity privacy model. The k-anonymity model requires that an individual should not be personally identifiable from a group of size smaller than k based on the quasi-identifier (QID), where QID is a set of attributes that may serve as an identifier in the data set. Later to recover the loop holes of the K-anonymity principal, l-diversity principal was introduced. L-diversity requires that every QID group should contain at least “well-represented” values for the sensitive attribute. Similarly, there are a number of other partition-based privacy models such as (α, k)-anonymity, (c, k)-safety, and t-closeness that differently model the adversary and have different assumptions about her background knowledge. However, recent research has indicated that these privacy models are vulnerable to various privacy attacks and provide insufficient privacy protection. In this paper, we adopt differential privacy, a recently proposed privacy model that provides a provable privacy guarantee. Differential privacy is a rigorous privacy model that makes no assumption about an adversary's background knowledge. A differentially private mechanism ensures that the probability of any output (released data) is equally likely from all nearly identical input data sets and, thus, guarantees that all outputs are insensitive to any individual's data. In other words, an individual's privacy is not at risk because of the participation in the data set. In this paper, we present an algorithm for differentially private data release for vertically partitioned data between two parties. We take the single-party algorithm for differential privacy that has been recently proposed by Mohammed et al as a basis and extend it to the two party setting. Additionally, the proposed algorithm satisfies the security definition of the semi-honest adversary model. In this model, parties follow the algorithm but may try to deduce additional information from the received messages. Therefore, at any time during the execution of the algorithm, no party should learn more information about the other party's data than what is found in the final integrated table, which is differentially private. The main contribution of our paper can be summarized as follows:

We present a two-party protocol for the exponential mechanism. We use this protocol as a sub-protocol of our main algorithm, and it can also be used by any other algorithm that uses the exponential mechanism in a distributed setting. We implement the first two-party data publishing algorithm for vertically partitioned data that generate an integrated data table satisfying differential privacy. The algorithm also satisfies the security definition in the secure multiparty computation (SMC) literature. The rest of the paper is organized as follows: In Section2, we present an overview of ϵ-differential privacy. In section3, we talked about Vertical partitioning of the data in brief. In Section 4, we briefly review the security multiparty computation definition. In Section 5, we describe the two-party protocol for the exponential mechanism. The implementation of two-party data publishing algorithm for vertically partitioned data is presented in Section 6.

II. DIFFERENTIALLY PRIVACY

Differential privacy will take the view that it was not, with the rationale that the impact on the smoker is the same independent of whether or not he was in the study. It is the conclusions reached in the study that affect the smoker, not his presence or absence in the data set. Differential privacy ensures that the same conclusions, for example, smoking causes cancer, will be reached, independent of whether any individual opts into or opts out of the data set. Specifically, it ensures that any sequence of outputs (responses to queries) is "essentially" equally likely to occur, independent of the presence or absence of any individual. Here, the probabilities are taken over random choices made by the privacy mechanism (something controlled by the data curator), and the term "essentially" is captured by a parameter. A smaller will yield better privacy (and less accurate responses). Differential privacy is a definition, not an algorithm. For a given computational task T and a given value of " there will be many differentially private algorithms for achieving T in an "-differentially private manner. Some will have better accuracy than others. When " is small, finding a highly accurate "-differentially private algorithm for T can be difficult, much as finding a numerically stable algorithm for a specific computational task can require effort.

Definition: A randomized function K gives ϵ-differential privacy if for all data sets D1 and D2 differing on at most one element, and all S Є Range(K),

$$\Pr[K(D1) \in S] \leq \exp(\epsilon) \times \Pr[K(D2) \in S] \quad (1)$$

A mechanism K satisfying this definition addresses concerns that any participant might have about the leakage of her personal information x: even if the participant removed her data from the data set, no outputs (and thus consequences of outputs) would become significantly more or less likely. For example, if the database were to be consulted by an insurance provider before deciding whether or not to insure Terry Gross, then the presence or absence of Terry Gross in the database will not significantly affect her chance of receiving coverage. This definition extends to group privacy as well. A collection of c participants might be concerned that their collective data might leak information, even when a single participant's does not. Using this definition, we can bound the dilation of any probability by at most exp(ϵc), which may be tolerable for small c. Note that we specifically aim to disclose aggregate information about large groups, so we should expect privacy bounds to disintegrate with increasing group size.

A. Achieving Differential Privacy

We now describe a concrete interactive privacy mechanism achieving ϵ-differential privacy. The mechanism works by adding appropriately chosen random noise to the answer a = f(X), where f is the query function and X is the database; thus the query functions may operate on the entire database at once. It can be simple – eg, "Count the number of rows in the database satisfying a given predicate" – or complex – eg, "Compute the median value for each column; if the Column 1 median exceeds the Column 2 median, then output a histogram of the numbers of points in the set S of orthants, else provide a histogram of the numbers of points in a different set T of orthants."

Note that the complex query above (1) outputs a vector of values and (2) is an adaptively chosen sequence of two vector-valued queries, where the choice of second query depends on the true answer to the first query. Although complex, it is solely a function of the database. We handle such queries in Theorem. For example, suppose the adversary first poses the query "Compute the median of each column," and receives in response noisy versions of the medians. Let M be the reported median for Column 1 (so M is the true median plus noise). The adversary may then pose the query: "If M exceeds the true median for Column 1 (ie, if the added noise was positive), then . . . else . . ." This second query is a function not only of the database but also of the noise added by the privacy mechanism in responding to the first query; hence, it is adaptive to the behaviour of the mechanism.

B. Exponential Noise and the L1-Sensitivity

We will achieve ϵ-differential privacy by the addition of random noise whose magnitude is chosen as a function of the largest change a single participant could have on the output to the query function; we refer to this quantity as the sensitivity of the function.

Definition: For $f : D \rightarrow R^d$, the L1-sensitivity of f is

$$\Delta f = \max_{D1,D2} \| f(D1) - f(D2) \| \qquad (2)$$

for all D1,D2 differing in at most one element.

For many types of queries Δf will be quite small. In particular, the simple counting queries ("How many rows have property P?") have $\Delta f \leq 1$. Our techniques work best – ie, introduce the least noise – when Δf is small. Note that sensitivity is a property of the function alone, and is independent of the database.

The privacy mechanism, denoted K_f for a query function f, computes f(X) and adds noise with a scaled symmetric exponential distribution with variance σ^2 (to be determined in the Theorem below in each component, described by the density function

$$Pr[Kf(X)=a] \propto \exp(-\| f(X) - a \|_1/\sigma) \qquad (3)$$

This distribution has independent coordinates, each of which is an exponentially distributed random variable. The implementation of this mechanism thus simply adds symmetric exponential noise to each coordinate of f(X).

Theorem: For $f : D \rightarrow R^d$, the mechanism K_f gives $(\Delta f/\sigma)$-differential privacy.

Proof: Starting from (3), we apply the triangle inequality within the exponent, yielding for all possible responses

$$Pr[K_f\ (D1)=r] \leq Pr[K_f\ (D2) = r] \times \exp(\| f(D1) - f(D2) \| / \sigma) \qquad (4)$$

The second term in this product is bounded by $\exp(\Delta f/\sigma)$, by the definition of Δf. Thus (1) holds for singleton sets S = {a}, and the theorem follows by a union bound.

Theorem describes a relationship between Δf, σ, and the privacy differential. To achieve ϵ-differential privacy, one must choose $\sigma \geq \epsilon / \Delta f$.

The importance of choosing the noise as a function of the sensitivity of the entire complex query is made clear by the important case of histogram queries, in which the domain of data elements is partitioned into some number k of classes, such as the cells of a contingency table of gross shoe sales versus geographic regions, and the true answer to the query is the k-tuple of the exact number of database points in each class. Viewed naively, this is a set of k queries, each of sensitivity 1, so to ensure ϵ-differential privacy it follows from k applications of Theorem (each with d = 1) that it suffices to use noise distributed according to a symmetric exponential with variance k/ϵ in each component. However, for any two databases D1 and D2 differing in only one element, $\|f(D1) - f(D2)\| = 1$, since only one cell of the histogram changes, and that cell only by 1. Thus, we may apply the theorem once, with d = k and Δf = 1, and find that it suffices to add noise with variance 1/ϵ rather than d/ϵ.

III. VERTICALLY PARTITIONED DATA

Data is said to be vertically partitioned when several organizations own different attributes of information for the same set of entities. Thus, vertical partitioning of data can formally be defined as follows: First, define a dataset D in terms of the entities for which the data are collected and the information that is collected for each entity. Thus, D ≡ (E, I), where E is the entity set for whom information is collected and I is the feature set that is collected. Assume that there are k different sites, P1,...,Pk collecting datasets D1 ≡ (E1, I1),...,Dk ≡ (Ek,Ik) respectively. Therefore, data is said to be vertically partitioned if $E = \cap_i E_i = E1 \cap ... \cap Ek$, and $I = U_i I_i = I_1 U ... UI_k$. In general, distributed data can be arbitrarily partitioned. Vertical partitioning can also be defined as a special case of arbitrary partitioning, where all of the partitions consist of information about the same set of entities.

A. Vertically Partitioning

A common form of vertical partitioning is to split dynamic data (slow to find) from static data (fast to find) in a table where the dynamic data is not used as often as the static. Creating a view across the two newly created tables restores the original table with a performance penalty, however performance will increase when accessing the static data e.g. for statistical analysis.

Like horizontal partitioning, vertical partitioning lets queries scan less data. This increases query performance. For example, a table that contains seven columns of which only the first four are generally referenced may benefit from splitting the last three columns into a separate table.

Vertical partitioning should be considered carefully, because analyzing data from multiple partitions requires queries that join the tables. Vertical partitioning also could affect performance if partitions are very large.

Partitioning is important for the following reasons:

- For easy management
- To assist backup/recovery
- To enhance performance.

➢ **For Easy Management**

The fact table in a data warehouse can grow up to hundreds of gigabytes in size. This huge size of fact table is very hard to manage as a single entity. Therefore it needs partitioning.

➢ **To Assist Backup/Recovery**

If we do not partition the fact table, then we have to load the complete fact table with all the data. Partitioning allows us to load only as much data as is required on a regular basis. It reduces the time to load and also enhances the performance of the system.

Note: To cut down on the backup size, all partitions other than the current partition can be marked as read-only. We can then put these partitions into a state where they cannot be modified. Then they can be backed up. It means only the current partition is to be backed up.

➢ **To Enhance Performance**

By partitioning the fact table into sets of data, the query procedures can be enhanced. Query performance is enhanced because now the query scans only those partitions that are relevant. It does not have to scan the whole data.

Vertical partitioning can be performed in the following two ways:

- Normalization
- Row Splitting

➢ **Normalization**

Normalization is the standard relational method of database organization. It is a process of removing redundant columns from a table and putting them in secondary tables that are linked to the primary table by primary key and foreign key

relationships. It also involves this splitting of columns across tables, but vertical partitioning goes beyond that and partitions columns even when already normalized.
Take a look at the following tables that show how normalization is performed.

P_id	Qty	Sales date	Store id	Store name	Location
30	5	3-8-13	16	Sunny	Bangalore
35	4	3-9-13	16	Sunny	Bangalore
40	5	3-9-13	64	San	Mumbai
45	7	3-9-13	16	Sunny	Bangalore

Table 1: Table before Normalization

Store_id	Store_name	Location
16	Sunny	Bangalore
64	San	Mumbai

P_id	Qty	Sales_date	Store_id
30	5	3-8-13	16
35	4	3-9-13	16
40	5	3-9-13	64
45	7	3-9-13	16

Table 2: Table after Normalization

- **Row Splitting**

Row splitting tends to leave a one-to-one map between partitions. The motive of row splitting is to speed up the access to large table by reducing its size. Row splitting divides the original table vertically into tables with fewer columns. Each logical row in a split table matches the same logical row in the other tables as identified by a UNIQUE KEY column that is identical in all of the partitioned tables. For example, joining the row with ID 712 from each split table re-creates the original row.

IV. SECURE MULTIPARTY COMPUTATION

The aim of secure multiparty computation is to enable parties to carry out such distributed computing tasks in a secure manner. The setting of secure multiparty computation encompasses tasks as simple as coin-tossing and broadcast, and as complex as electronic voting, electronic auctions, electronic cash schemes, contract signing, anonymous transactions, and private information retrieval schemes. Consider for a moment the tasks of voting and auctions. The privacy requirement for an election protocol ensures that no parties learn anything about the individual votes of other parties; the correctness requirement ensures that no coalition of parties has the ability to influence the outcome of the election beyond simply voting for their preferred candidate. Likewise, in an auction protocol, the privacy requirement ensures that only the winning bid is revealed (if this is desired); the correctness requirement ensures that the highest bidder is indeed the winning party (and so the auctioneer, or any other party, cannot bias the outcome). Due to its generality, the setting of secure multiparty computation can model almost every cryptographic problem. A number of different definitions have been proposed and these definitions aim to ensure a number of important security properties that are general enough to capture most (if not all) multiparty computation tasks. We now describe the most central of these properties:

- **Privacy:** No party should learn anything more than its prescribed output. In particular, the only information that should be learned about other parties' inputs is what can be derived from the output itself. For example, in an auction where the only bid revealed is that of the highest bidder, it is clearly possible to derive that all other bids were lower than the winning bid. However, this should be the only information revealed about the losing bids.
- **Correctness:** Each party is guaranteed that the output that it receives is correct. To continue with the example of an auction, this implies that the party with the highest bid is guaranteed to win, and no party including the auctioneer can alter this.
- **Independence of Inputs:** Corrupted parties must choose their inputs independently of the honest parties' inputs. This property is crucial in a sealed auction, where bids are kept secret and parties must ¯x their bids independently of others. We note that independence of inputs is not implied by privacy. For example, it may be possible to generate a higher bid without knowing the value of the original one. Such an attack can actually be carried out on some encryption schemes (i.e., given an encryption of $100, it is possible to generate a valid encryption of $101, without knowing the original encrypted value).
- **Guaranteed Output Delivery:** Corrupted parties should not be able to prevent honest parties from receiving their output. In other words, the adversary should not be able to disrupt the computation by carrying out a \denial of service" attack.
- **Fairness:** Corrupted parties should receive their outputs if and only if the honest parties also receive their outputs. The scenario where a corrupted party obtains output and an honest party should not be allowed to occur. This property can be crucial, for example, in the case of contract signing. Specifically, it would be very problematic if the corrupted party received the signed contract and the honest party did not.

Symbolically, Secured Multiparty Computation is defined as Let f : {0,1}* × {0,1}*→{0,1}*×{0,1}* be a probabilistic polynomial-time functionality, where f1(x,y) (f2(x; y), respectively) denotes the first (second, respectively) element of f(x,y). Let Π be a two-party protocol for computing f. Let the view of the first (second, respectively) party during an execution of protocol Π on (x, y) denoted $view_1^{\pi}$ ($view_2^{\pi}$, respectively) be (x,r1,m1, . . .,mt) ((y,r2,m1, . . .,mt),respectively), where r1 represents the outcome of the first (r2 the second, respectively) party's internal coin tosses and mi represents the ith message the first (second, respectively) party has received. The output of the first (second, respectively) party during an execution of Π on (x,y) denoted $output_1^{\pi}$ (x, y) ($output_2^{\pi}$ (x, y), respectively) is implicit in the party's view of the execution. We say that Π securely computes f if there exist probabilistic polynomial time algorithms denoted S1 and S2 such that

{(S1(x,f1(x,y)), f2(x; y))}x,yϵ{0,1}
$\overset{c}{\equiv}${($view_1^{\pi}$(x,y),$ouput_2^{\pi}$(x,y))}x;y2ϵ{0,1}*
{(f1(x,y),S2(x, f1(x; y)))}x;yϵ{0,1}*
$\overset{c}{\equiv}${($output_1^{\pi}$ (x,y), $view_2^{\pi}$(x,y))} x;yϵ{0,1}* ,

where $\overset{c}{\equiv}$ denotes computational indistinguishability.

Two probability distributions are computationally indistinguishable if no efficient algorithm can tell them apart. Namely, the output distribution of every efficient algorithm is oblivious whether the input is taken from the first distribution or from the second distribution [20]. Many of the protocols, as in the case of the proposed algorithm in this paper, involve the composition of secure subprotocols in which all intermediate outputs from one subprotocol are inputs to the next subprotocol. These intermediate outputs are either simulated given the final output and the local input for each party or computed as random shares. Random shares are meaningless information by themselves. However, shares can be combined to reconstruct the result. Using the composition theorem [21], it can be shown that if each subprotocol is secure, then the resulting composition is also secure.

V. TWO-PARTY PROTOCOL FOR EXPONENTIAL MECHANISM

In this section, we present a two-party protocol for the exponential mechanism together with a detailed analysis. The exponential mechanism chooses a candidate that is close to optimum with respect to a utility function while preserving differential privacy. In the distributed setting, the candidates are owned by two parties and, therefore, a secure mechanism is required to compute the same output while ensuring that no extra information is leaked to any party.

A. Distributed Exponential Mechanism (DistExp)

The distributed exponential mechanism presented in Algorithm1 takes the following items as input:

1.Finite discrete alternatives $\{(v_1, u_1), \ldots, (v_n, u_n)\}$, where a pair (v_i, u_i) is composed of the candidate vi and its score ui. Parties P1 and P2 own $(v_1, u_1), \ldots, (v_j, u_j)$ and $(v_{j+1}, u_{j+1}) \ldots (v_n, u_n)$, respectively.

2.Privacy budget ϵ.

Algorithm 1 Distributed Exponential Mechanism

Input: Candidate-score pairs $((v_1; u_1), \ldots, (v_n, u_n))$ owned by the parties, and the privacy budget ϵ

Output: Winner w

```
1: P1 evaluate $s_1 \leftarrow \sum_{i=1}^{j} exp(\frac{\epsilon u_i}{2\Delta u})$
2: P2 evaluates $s_2 \leftarrow \sum_{i=j+1}^{n} exp(\frac{\epsilon u_i}{2\Delta u})$
3: P1 and P2 execute RVP to compute random shares
R1 and R2, where (R1 + R2) $\epsilon$ (S1+S2);
4: for k=1 to n do
5: if k $\leq$ j then6: P1 evaluates L1$\leftarrow \sum_{j=1}^{k} exp(\frac{\epsilon u_i}{2\Delta u})$ ;
7: P2 evaluates L2 $\leftarrow$0;
8: else
9: P1 evaluates L1$\leftarrow \sum_{i=1}^{j} exp(\frac{\epsilon u_i}{2\Delta u})$ ;
10: P2 evaluates L2$\leftarrow \sum_{i=j+1}^{k} exp(\frac{\epsilon u_i}{2\Delta u})$ ;
11: end if
12: P1 and P2 execute COMPARISON(R1;R2; L1; L2);
13: if b= 0 then
14: w $\leftarrow$vk;
15: return w;
16: end if
17: end for
```

Algorithm 2 Comparision

Input: Random shares R1 and R2, and values L1 and L2

Output: b

```
1: R = add(R1,R2);
2: L = add(L1, L2);
3: b = compare(R,L);
4: return b;
```

The protocol outputs a winner candidate depending on its score using the exponential mechanism. The scores of the candidates can be calculated using different utility functions. Given the scores of all the candidates, exponential mechanism selects the candidate v_j with the following probability, where Δu is the sensitivity of the chosen utility function:

$$\frac{exp(\frac{\epsilon u_i}{2\Delta u})}{\sum_{i=1}^{n} exp(\frac{\epsilon u_i}{2\Delta u})} \qquad (5)$$

The distributed exponential mechanism can be summarized as follows:

Computing(3): Consider an interval [0,1] partitioned into segments as per the probability mass defined in (5) for the candidates. Now, we sample a random number uniformly in the same range and the partition in which the random number falls determines the winner candidate. Since we are not aware of any secure division scheme that fits the situation, we solve this in a different way. We partition the interval $[0, \sum_{i=1}^{n} exp(\frac{\epsilon u_i}{2\Delta u})]$ into n segments, where each segment corresponds to a candidate v_i and has a subinterval of length exp $(\frac{\epsilon u_i}{2\Delta u})$and the sampling procedure is carried out like the earlier which determines the winner candidate.

Selecting the random number: Sampling the random number is carried out using Random Value protocol(RVP). Each party first computes individually exp$(\frac{\epsilon u_i}{2\Delta u})$ for its candidates. Then both P_1 and P_2 compute $s_1 = \sum_{i=1}^{j} exp(\frac{\epsilon u_i}{2\Delta u})$ and $s_2 = \sum_{i=j+1}^{n} exp(\frac{\epsilon u_i}{2\Delta u})$, respectively. P_1 and P_2 need to pick up a random number in the range $[0, s_1+s_2]$ where $s_1+s_2 = \sum_{k=1}^{n} exp(\frac{\epsilon u_i}{2\Delta u})$. This can be achieved by RVP. It takes s_1 and s_2 from the parties as inputs and outputs $R_1$and R_2 to the parties respectively, where $R=R_1+R_2$

VI. IMPLEMENTATION OF TWO-PARTY DIFFERENTIALLY PRIVATE DATA RELEASE ALGORITHM

The basic idea is to anonymize the data by a sequence of specializations starting from the root or the topmost general state. The specialization process is splitting the taxonomy tree downwards where the child values replace that of the parents'. It is an iterative process. Each iteration of the specialization process selects a winner candidate using the distributed exponential mechanism (Algorithm 1) which is owned by either of the parties. Winner candidates are chosen based on their score values which are determined using different utility functions. Once the winner candidate is chosen, both parties perform specialization process by splitting their records into child partitions with respect to the taxonomy trees provided. If the winner candidate belongs to P1, it specializes and instructs P2 to specialize. If winner candidate doesn't belong to P1, it waits to hear from P2 for specialization. This process is repeated as per the number of specializations which has to be given as input. The taxonomy is fixed, so splitting the records according to the taxonomy doesn't violate the differential privacy. Finally a true count and a noisy count is added to the leaves of the taxonomy tree to ensure overall ϵ-differentially private output.

Consider the data in the tree below. Initially, D contains one root node showing all the generalized records (Branch, Sex and percentage of students). That means a particular record in the table can contain the details of student of any branch out of the branches available in the college, of any sex, with any percentage. Now, to find the winner candidate both the parties perform DISTEXP. Say branch is the winner candidate, Party P1 first creates two child nodes as shown in the figure. P1 sends instruction to P2. P2 then creates two child nodes under the root D. Suppose the next winning candidate is Sex, two parties cooperate to create further specialized partitions resulting in the generalized table.

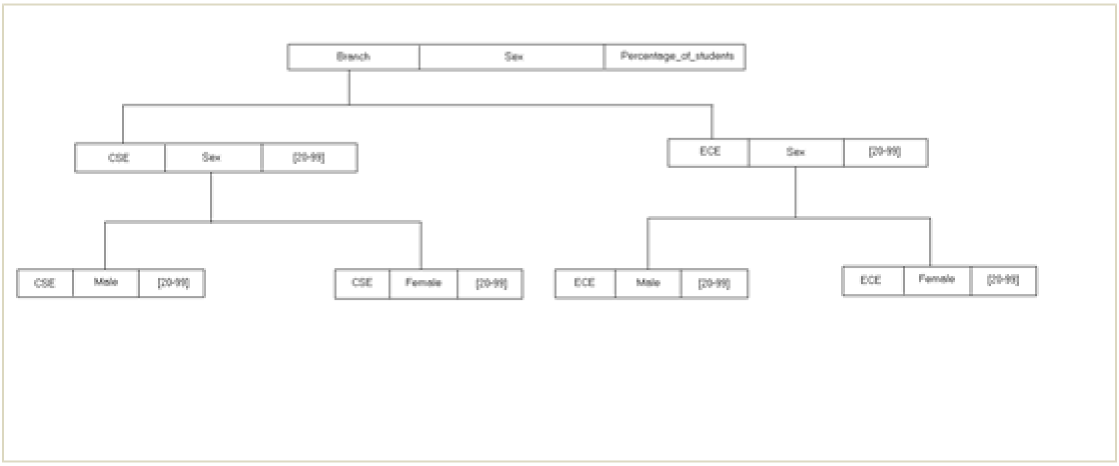

Fig 2.Generalized Data Table (D). Distributed exponential mechanism is used for specializing the predictor attributes in a top-down manner.

VII. CONCLUSION

In this paper, we have presented the differential privacy model which secures the private data shared between two parties. We proposed a two party protocol for exponential mechanism and implemented and algorithm for private data release. We have shown that algorithm is secure under as per the secured multiparty computation. It proves to be better when compared to the single party algorithm and better data utility than the distributed k-anonymity algorithm.

VIII. OVERALL IDEA OF THE PAPER

Data between two parties where integrated by using shared identifier such as ssn, name, employee id. Integrated data is pre-processed ie.. removing all the explicit identifiers such as name, age, etc.. but there may be a existence of pseudo identifiers which may lead to link attack. Integrated data gets generalized to hide the sensitive details. Owner of the data generalizes the details by assuming some of the field as sensitive. Hence security is satisfied statistically. A method is proposed to provide dynamic security called differential privacy which does not assume about adversaries background knowledge.

IX. ACKNOWLEDGMENT

Authors wish to express their deepest gratitude to their supervisor, Assistant Prof. Mrs. Y. Rohita who gave her support, guidance, encouragement and insights throughout the preparation of this paper. The authors are also grateful to their parents, for their support and engagement.

REFERENCES

[1] Latanya Sweeney, "*k*-Anonymity: A Model for Protecting Privacy", School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.
[2] Differential Privacy, Cynthia Dwork, Microsoft Research.
[3] The Algorithmic Foundations of Differential Privacy, Cynthia Dwork Microsoft Research, USA and Aaron Roth, University of Pennsylvania, USA.
[4] Yehuda Lindell and Benny Pinkas, "Secure Multiparty Computation for Privacy-Preserving Data Mining".