IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 4, Ver. III (Jul.-Aug. 2017), PP 43-49
www.iosrjournals.org

Privacy-Preserving Data Mining with Random decision tree framework

*Ms. Ch.Likitha Sravya, Mrs. G.V Rajya Lakshmi
M. Tech Dept. of computer science and Engineering LBRCE College, Mylavaram, India
Corresponding Author: Ms. Ch.Likitha Sravya

Abstract: *Data mining is the useful tool to discovering the knowledge from large data. Different methods & algorithms are available in data mining. Classification is most common method used for finding the mine rule from the large database. Decision tree method generally used for the Classification, because it is the simple hierarchical structure for the user understanding & decision making. Various data mining algorithms available for classification based on Artificial Neural Network, Nearest Neighbour Rule & Bayesian classifiers but decision tree mining is simple one. ID3 and C4.5 algorithms have been introduced by J.R Quinlan which produce reasonable decision trees. The objective of this paper is to present these algorithms. At first we present the classical algorithm that is ID3, then highlights of this study we will discuss in more detail C4.5 this one is a natural extension of the ID3 algorithm. And we will make a comparison between these two algorithms and others algorithms such as C5.0 and CART.*

Indexterms*: Data mining, Privacy-preserving data mining, classification horizontal partitioning, vertical partitioning*

Date of Submission: 04-07-2017 Date of acceptance: 26-07-2017

I. Introduction

Data mining is the process of discovering interesting knowledge, such as patterns, associations, changes, and anomalies and significant structures, from large amounts of data stored in databases, data warehouses, or other information repositories. Due to the wide availability of huge amounts of data in electronic forms, and the coming up need for turning such data into useful information and knowledge for many broad applications including market analysis, business management, and decision support, data mining has fascinated a great deal of interest in information industry in recent years. Data collected from information providers are important for pattern reorganization and decision making. The data collection process takes time and efforts hence sample datasets are sometime stored for reuse. However attacks are attempted to take these sample datasets and private information may be leaked from these stolen datasets. Therefore privacy preserving data mining algorithms are developed to convert sensitive datasets into sanitized version or altered version in which private or sensitive information is hidden from unauthorized or unofficial retrievers. Privacy Preserving Data Mining (PPDM) refers to the area of data mining that aims to protect sensitive information from illegal or unwanted disclosure.

Privacy Preservation Data Mining was introduced to preserve the privacy during mining process to enable conventional data mining technique. Many privacy preservation approaches were developed to protect private information of sample dataset. Modern research in privacy preserving data mining mainly falls into one of two categories:

1)perturbation and randomization-based approaches, and 2) secure multiparty computation (SMC)-based approaches.SMC approaches employ cryptographic tools for collaborative data mining computation by multiple parties. Samples are distributed among different parties and they take part in the information computation and communication process. SMC research focuses on protocol development for protecting privacy among the involved parties or computation efficiency; however, centralized processing of samples and storage privacy is out of the scope of SMC [1]. However, while perturbation based solutions do not provide strict privacy, cryptographic solutions are too inefficient and infeasible to enable truly large scale analytics to see the era of big data. In this paper, We introduce a new perturbation and randomization based approach that protects centralized sample data sets utilized for decision tree data mining. Privacy preservation is applied to alter or sanitize the samples earlier to their release to third parties in order to moderate the threat of their accidental disclosure or theft. The proposed approach is based on random decision trees (RDT), developed by Fan et al. [2]. In contrast to other sanitization methods, our approach does not affect the accuracy of data mining results. The decision tree can be built directly from the altered or sanitized data sets, such that the originals do not need to be

reconstructed. In addition to this, this approach can be applied at any time during the data collection process so that privacy protection can be in effect even while samples are still being collected [1].

One important property of RDT is that the same code can be used for more data mining tasks: classification, regression, ranking and multiple classification [2], [3], [4]. As shown previously, the RDTis an efficient implementationof Bayesoptimalclassifier (BOC) [2], effective non-parametric density approximation[4], and can be explained in a high order statistics such as moments [5]. The use of the multiple RDTs in various learning tasks offers many benefits over other traditional classification/tree building techniques, because its structure and progression lends itself to modification for distributed/ parallel tasks. RDT is also an outstanding candidate for use in secure preserving distributed data mining since:

1. Randomness in structure rather than simple perturbation of input/output is more effective perturbing the input or output from a database to achieve secure works, but the utility of the details garnered from data mining can be diminished if the perturbations are not carefully controlled, or conversely, details can be leaked if the detail is not perturbed enough. Instead, we can exploit the design properties of RDT to generate trees that are random in structure, providing us with a near end effect as perturbation without the associated pitfalls. A random structure provides security against leveraging a priority information to discover the total classification model or instances.
2. Purely cryptographic approaches are often too slow to be practical and can become computationally costly as the size of the data set increases and intercommunications between different parties increase. RDT provides a convenient escape from this paradigm thanks to its structural properties, more specifically, the fact that only specific nodes (the leaves) in the classification tree need to be encrypted/decrypted, and secure token passing stop attackers from utilizing counting techniques to decipher instance classifications, as the branch struc-ture of the tree is hidden from all parties.
3. RDT is common approach in which the same code works for classification, regression, ranking and multilabel classification. Thus the identical techniques, we can answer four typical learning problems in the identical framework.
4. An extra security advantage of RDTs is that they can be simply made differentially private. As shown by Jagannathan et al. [6], the node statistics can be viewed as queries over the training data. Therefore, standard techniques can be used to return differentially secure results, without significant lost of accuracy.

In this paper, we develop methods to securely build RDTs for two horizontally and vertically partitioned data sets. We implement the proposed protocols and examine the computation and communication price, and security. We also compare the performance of the proposed protocols with the existing ID3-based protocols [7].Our main contribution is to realize that RDTs can supply better security with more efficiency.

II. An Example For RANDOM DECISION TREES

The RDTs algorithm builds more(or m) iso-depth RDTs. One important aspect of RDTs is that the structure of a random tree is constructed completely independent of the training data. The RDT algorithm can be break into two stages, training and classification. The training phase consists of building the trees (BuildTreeStructure) and populating the nodes with training sample data (UpdateStatistics). It is suppose that the number of attributes is based on the training data set. The depth of each tree is decided based on a heuristic—Fan et al. [2] show that when the depth of the tree is equal to half of the total number of quality present in the data, the high diversity is achieved, preserving the advantage of random modeling.

The process for generating a tree is as follows. First, start with a list of features (attributes) from the data set. create a tree by randomly choosing one of the quality do not using any training data. The tree end growing once the height limit is reached. Then, use the training data to update the statistics of each node. Note that only the leaf nodes need to store the number of examples of different classes that are classified through the nodes in the tree. The training data is scanned exactly once to store the statistics in multiple random trees. When classifying a new case x, the probability outputs (or regression/ranking values for regression, rank-ing and multi-label classification problems) from more trees are averaged to estimate the a posteriori probability.

III. Analysis

We first derive the computation and communication cost, assuming k sites, n attributes, j instances, p class values, and m random trees, and then go through the security analysis.

3.1 Computation Cost

Since we are particularly interested in comparing against cryptographic algorithms, we only count the number of cryptographic operations (encryption, decryption, exponen-tiation). In general, the non-cryptographic operations do not incur much computation overhead as compared to the cryp-tographic operations, so their overhead can be ignored.

For horizontally partitioned data, only the leaf nodes are encrypted, by each party. Since the depth of a tree is n=2 (assuming binary splits), there are $2^{n=2\ 1}$ leaf nodes. With p classes, m trees, and k parties, there are a total of Oðpmk$2^{n=2\ 1}$Þ encryptions. Classifying a new instance requires OðmpÞ homomorphic multiplications and p threshold decryptions. For vertically partitioned data, all leaf nodes are originally encrypted and then updated. Thus, there are Oðmp$2^{n=2\ 1}$Þ encryptions and OðjpÞ homomorphic multiplications. Classifying a new instance requires p threshold decryptions.

3.2 Communication Cost

For horizontally partitioned data, the encrypted leaf values for the entire trees can be exchanged in one message. Thus, for the tree building phase OðmkÞ messages are exchanged. For classifying a new instance OðkÞ messages are exchanged for the threshold decryption. In the case of vertically partitioned data, for building a tree, in the worst case, a message is exchanged for every node. Since there are n attributes, the depth of each tree is n=2, thus giving us Oð$2^{n=2}$Þ nodes (assuming binary splits). Therefore, in gen-eral, Oðm$2^{n=2}$ Þ messages for the tree building phase. For updating the statistics, each instance travels down to the corresponding leaf node. Thus, we have n=2 messages for each instance for each tree, giving OðnmjÞ messages. Simi-larly, classifying an instance requires OðnmÞ messages.

3.3 Security Analysis

We first look at the horizontal case. In case (1c), the tree structure is known to everyone, though the leaf class distri-bution vector for all trees are encrypted using the threshold encryption. Since semantically secure homomorphic encryption is used, this reveals no information to any of the parties about the values. Furthermore, since threshold encryption is used, none of the encrypted values can be decrypted without interaction from the minimum (thresh-old) number of parties. The threshold is a parameter that can be set (i.e., for ðt; kÞ threshold encryption, it is necessary to have interaction between at least t of the k parties). If the threshold is set to k (the total number of parties), then none of the values can be decrypted even if all of the other parties collude against one party. However, when an instance is classified, the class distribution vector corresponding to the sum is computed and revealed to the party. It is important to note that this does not reveal more information than revealing the average (since the sum can be computed from the average), though it does reveal more information than a normalized class probability distribution (namely, how many other instances in the training data reach the corresponding leaf node). Every classification essentially reveals one linear equation in the global class distribution vector for the m leafs. Given enough linear equations, it may be possible to solve the system of linear equations to find the actual class distribution vectors corresponding to the leaf node. However, this will still be only the global values, and it is impossible to accurately estimate the local values (since infinite solutions are possible), unless every party colludes against one party.

Note that in such a case the information leaked is the number of records of a particular class reaching a particular leaf node. It is possible to protect this information from leak-ing. For example, one possibility is to multiply the encrypted vector by a positive (unknown) random number to hide the information about the number of participating records. This can be done by having each party multiply the global encrypted vector by a random amount. This will still preserve the distribution of class values. Indeed, in this sense model privacy is not preserved since the proportion of classes will be observable to the adversary. However, note that this is unavoidable if the class distribution is to be reported (is part of the output). If the classifier were supposed to only report the predicted class instead of the class distribution, this information leakage could be pre-vented (by carrying out comparisons over the encrypted values, and only reporting the class with the largest value).

In the case of vertical partitioning, even the tree struc-ture is unknown to any of the parties. The process of building a tree does not leak any information except what is leaked through classifying an instance (because this is the basic underlying operation). However, each classification does reveal some limited information about the branch taken to reach the leaf (since the branch must be limited to the values contained within the instance). Even in this case, since each new instance is also distrib-uted, it is impossible to completely reconstruct the entire tree without the help of all parties.

IV. Experimental Evaluation

The experiments were performed in a distributed environ-ment wherein each node was a Linux box with 4 GB mem-ory and 2.8 GHz Intel Xeon processor. The nodes were connected in a star topology over 1 Gbit Ethernet.

4.1 Adapting Weka

Weka [13] developed at the University of Waikato in New Zealand, is a collection of machine learning algorithms for data mining tasks implemented in Java. Apart from providing algorithms, it is a general implementation framework, along with support classes and documenta-tion. It is extensible and convenient for prototyping pur-poses. However, the Weka system is a centralized system meant to be used at a single site.

We have implemented our Privacy Preserving RDT algorithm and integrated it into Weka version 3:6. We fol-low the general model of operation to extend Weka for privacy preserving distributed classification of Vaidya et al. [7]. Fig. 4 demonstrates the basic interaction diagram. Java RMI is used to implement the distributed protocol between all of the sites. Similarly, when split vertically, we assume that each party has an approximately equal number of attrib-utes. Nursery has eight attributes (all categorical) and 12,960 instances; Mushroom has 22 attributes (all categorical) and 8,124 instances; Image Segmentation has 19 attributes (all numeric) and 2,310 instances; Car has six attributes (all cate-gorical) and 1,728 instances; For Nursery the depth of all the RDTs was 4 for both horizontal partitioned and vertical par-titioned case. For Mushroom the depth of RDTs was 8 for the horizontal partitioned case and reduced to 4 for the vertical partitioned case (to ensure sufficient memory for tree con-struction). We also grouped the category values of the attributes for Mushroom so that the domain cardinality of the attributes was no more than 4. For Image Segmentation the depth of RDTs was 10 for both horizontal partitioned and vertical partitioned case. All experiments were repeated three times and results were averaged. Table 2 shows that the time taken to build the classifier scales linearly with the number of trees, as does the time taken to classify a new instance. This is true regardless of

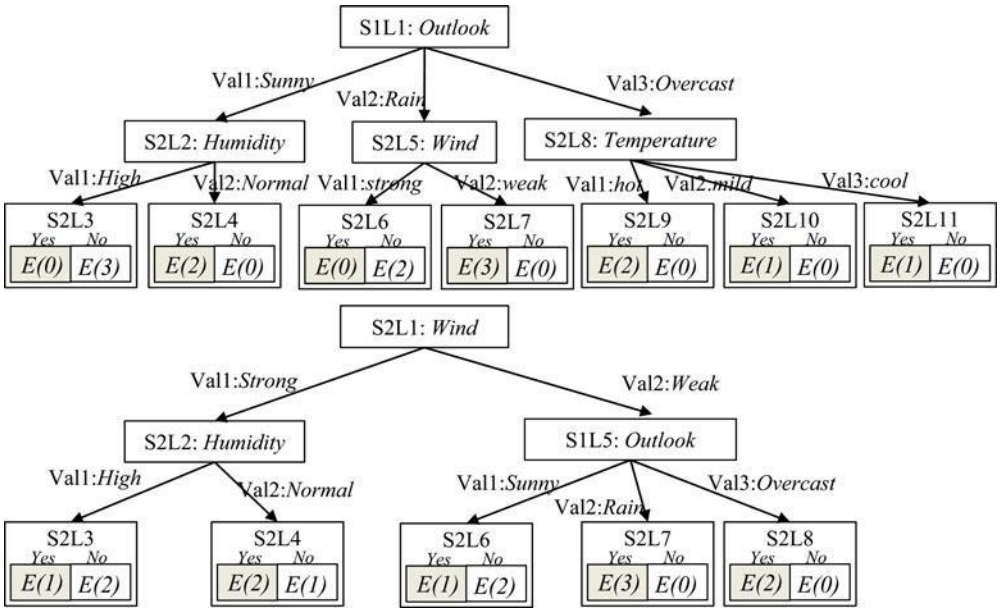

Fig.3. The random decision trees built on vertical partitionin

whether the underlying attributes are numeric or categori-cal. Note though, that the time required for vertically parti-tioned data (in both cases) is orders of magnitude higher than that required for horizontal partitioned data. This is since the time reported includes the time for computation and for communication, and is to be expected since the actual tree is distributed in the vertical partitioned case. Indeed, the degree of communication required for vertically partitioned data is proportional to the number of instances, where as it only depends on the number of parties for horizontally partitioned data. Thus, most of the time taken is due to the additional Java RMI invocations required for the vertical partitioned case. Note that once the relatively expensive protocol to build the RDT classifier has already been executed, it is a computationally trivial task to classify any given instance. Also, the classification accu-racy increases as the number of trees increase from 7-20. For Nursery and Image Segmentation, the accuracy results are similar for both horizontally and vertically partitioned cases since the depth of the RDTs was the same in both cases. The small variation is due to the random choice of testing instances. For Mushroom, the classification accuracy in the horizontal partitioned case (RDT depth 8) is signifi-cantly greater than vertical partitioning (RDT depth 4). This is due to the difference in RDT depth. Table 2 also shows that the time taken for training and classification in vertically partitioned data both increase as the number of sites increases. This is due to the additional communication (and RMI calls) required. The accuracy stays the same. Table 3 compares the time taken by ID3 and RDT for all data sets assuming vertical partitioning among three sites. The secure version of ID3[7] was used in the same distributed environment. From the results, it can be noted that building an RDT model is generally an order of magnitude faster than building the ID3 model. However,instance classification is significantly slower. This is because only one tree is used for classifying an instance using the ID3 model; whereas, multiple RDTs (10 or more) are used for classification. Nevertheless, the classification speed is reasonable. Note that the accuracy of RDT is worse in this case than ID3. However, the accuracy increases with larger number of trees and in general, is comparable to other inductive learning models [2], [3], [4].

4.2 Experimental Conclusion

The distributed RDT algorithms and implementation pre-sented in this paper are a significant step forward in creat-ing usable, distributed, privacy-preserving, data mining algorithms. The running time of the algorithms, is compar-atively much faster than the existing implementations, and is usable on everyday computing hardware. As compared to the standard, non privacy-preserving version, the accu-racy of the privacy-preserving solution is exactly the same, though the computational overhead is significant. How-ever, privacy is not free. In general, privacy-preserving protocols are more expensive than non-privacy-preserving protocols for the same problem. For example, [7] shows that the privacy-preserving ID3 requires two orders of magnitude larger computation time than the non privacy-preserving version. Indeed, this motivated us to build the more efficient solutions proposed in this paper, so that use of PPDM solutions can become a reality.

V. Related Work

There has been extensive work in privacy-preserving data mining. The perturbation approach, pioneered by Agrawal and Srikant [14], relies on adding "noise" to the data before the data mining process. Techniques are then used to some-how remove the noise from the data mining results. Several

TABLE 2 Experimental Results for the Mushroom, Nursery, Image Segmentation Comparing RDT Model Building Time (BT), Instance Classification Time (CT), and Classification

(a) Horizontally partitioned data

Nursery (No. of attributes = 8; No. of instances = 12,960; RDT Depth = 4)									
	3 sites			**4 sites**			**5 sites**		
	BT seconds	**CT** seconds	**Acc** %	**BT** seconds	**CT** seconds	**Acc** %	**BT** seconds	**CT** seconds	**Acc** %
7 trees	29.5	0.044	87.1	27.8	0.049	87.2	23.9	0.055	87.7
10 trees	33.8	0.046	86.6	29.4	0.049	87.3	27.2	0.055	90.8
14 trees	33.4	0.045	89.3	30.3	0.050	89.5	28.1	0.054	88.8
20 trees	37.9	0.046	89.6	33.5	0.051	88.0	32.3	0.056	92.2
Mushroom (No. of attributes = 22; No. of instances = 8124; RDT depth = 8)									
	3 sites			**4 sites**			**5 sites**		
7 trees	125.5	0.032	96.7	112.5	0.036	96.7	146.5	0.041	97.3
10 trees	177.3	0.033	97.6	190.2	0.038	96.9	196.7	0.044	98.2
14 trees	236.9	0.033	98.4	225.9	0.038	97.2	259.8	0.043	99.3
20 trees	349.2	0.035	99.0	379.4	0.044	98.5	342.4	0.045	99.1
Image Segmentation (No. of attributes = 19; No. of instances = 2310; RDT depth = 10)									
	3 sites			**4 sites**			**5 sites**		
7 trees	46.8	0.022	70.5	42.7	0.025	70.3	50.5	0.027	70.9
10 trees	62.2	0.024	74.8	64.4	0.026	75.0	66.5	0.028	75.7
14 trees	87.3	0.024	77.1	82.9	0.026	77.7	83.5	0.029	77.9
20 trees	110.5	0.025	81.4	110.2	0.027	80.9	105.7	0.031	81.3

(b) Vertically partitioned data

Nursery (No. of attributes = 8; No. of instances = 12,960; RDT Depth = 4)									
	3 sites			**4 sites**			**5 sites**		
	BT seconds	**CT** seconds	**Acc** %	**BT** seconds	**CT** seconds	**Acc** %	**BT** seconds	**CT** seconds	**Acc** %
7 trees	19,112	0.230	84.6	24,093	0.320	83.1	21,674	0.296	82.4
10 trees	24,489	0.353	89.1	30,817	0.387	90.1	33,350	0.417	84.0
14 trees	32,750	0.410	93.1	41,675	0.460	90.0	46,320	0.558	87.3
20 trees	52,113	0.626	89.3	56,968	0.653	90.2	66,158	0.758	85.2
Mushroom (No. of attributes = 22; No. of instances = 8124; RDT depth = 4)									
	3 sites			**4 sites**			**5 sites**		
7 trees	8,151	0.218	88.5	7,686	0.214	87.4	8,950	0.231	90.0
10 trees	10,481	0.257	93.1	11,707	0.283	88.6	12,487	0.295	93.6
14 trees	14,342	0.327	94.7	15,470	0.342	88.0	16,617	0.361	87.3
20 trees	19,798	0.426	89.0	22,965	0.457	90.1	25,143	0.498	87.7
Image Segmentation (No. of attributes = 19; No. of instances = 2310; RDT depth = 10)									
	3 sites			**4 sites**			**5 sites**		
7 trees	1410	0.192	70.5	1566	0.259	69.6	1575	0.301	70.6
10 trees	2077	0.222	75.7	2083	0.261	76.2	2250	0.328	76.8
14 trees	2552	0.233	77.5	2823	0.295	77.3	3207	0.364	78.7
20 trees	3849	0.283	81.9	4074	0.335	80.4	4851	0.441	81.2

privacy-preserving data mining algorithms have since been proposed [1], [15]. One point of concern is that, since the original data is still available (though in perturbed form), noise removal techniques can be used to give estimates of the original values, giving rise to debate about the security properties of such algorithms . The alternative approach to protecting privacy of dis-tributed sources using cryptographic techniques was first

applied in the area of data mining for the construction of decision trees by Lindell and Pinkas. This work falls under the framework of secure multiparty computation [8], achieving "perfect" privacy, i.e., nothing is learned that could not be deduced from one's own data and the resulting tree. The key insight was to trade off computation and communication cost for accuracy, improving efficiency over the generic secure multiparty computation method. However, the proposed solution is still too inefficient for practical usage.

TABLE 3 ID3 [7] and RDT Comparison

Car Dataset: No. of attributes = 6; No. of instances = 1728; No. of instances used for training = 1545; No. of instances used for testing = 183					
ID3			**RDT** (10 trees; depth 4)		
BT seconds	**CT** seconds	**Acc** %	**BT** seconds	**CT** seconds	**Acc** %
3041.26	0.0057	85.55	509.52	0.091	71.6
Nursery Dataset: No. of attributes = 8; No. of instances = 2000; No. of instances used for training = 1791; No. of instances used for testing = 209					
BT	**CT**	**Acc**	**BT**	**CT**	**Acc**
9084.1	0.0244	89	613.64	0.0968	83.2
Mushroom Dataset: No. of attributes = 22; No. of instances = 2060; No. of instances used for training = 1844; No. of instances used for testing = 216					
BT	**CT**	**Acc**	**BT**	**CT**	**Acc**
1600.24	0.0158	98.06	776.21	0.0932	89.4

There has been work in distributed construction of deci-sion trees on vertically partitioned data. Wang et al. present a solution based on passing the transaction identifiers between sites ; while this does not reveal specific attri-bute values, parties learn which transactions follow which path down the tree, enabling (for example) one site to say "these two individuals have the same attribute values." Du and Zhan do suggest a way of building a privacy-preserving decision tree classifier for vertically partitioned data. Their method is limited to two parties, assumes that both parties have the class attribute, and is not imple-mented. Vaidya et al. [7] extend this to the multi-party case, also relaxing the assumption that the class attribute must be known to all parties. This approach has been implemented, though the experimental results suggest that for large scale data, the computational complexity is quite high. These works make trade-offs between efficiency and information disclosure, however all maintain provable bounds on disclosure. Jagannathan et al. [6] propose ways to construct a differentially private RDT classifier from a centralized data set. Since our data is distributed, we cannot directly use. K-means clustering in vertically partitioned data , association rules in vertically partitioned data , and generalized approaches to reducing the number of "on-line" parties . The methodology for proving the correctness of the algo-rithm comes from secure multiparty computation [8], Currently, assem-bling these into efficient privacy-preserving data mining algorithms, and proving them secure, is a challenging task. This paper demonstrates how we can leverage the beneficial properties of both randomization and cryptography to pro-vide a highly efficient yet secure approach to performing classification. We hope that this approach will be pursued in future work to make other privacy-preserving data min-ing tasks more efficient.

VI. Conclusion

We have demonstrated that general and efficient distributed privacy preserving knowledge discovery is truly feasible. We have considered the security and privacy implications when dealing with distributed data that is partitioned either horizontally or vertically across multiple sites, and the chal-lenges of performing data mining tasks on such data. Since RDTs can be used to generate equivalent, accurate and sometimes better models with much smaller cost, we have proposed distributed privacy-preserving RDTs. Our approach leverages the fact that randomness in structure can provide strong privacy with less computation. The experiments show that the privacy preserving version of the RDT algorithm scales linearly with data set size, and requires significantly less time than alternative crypto-graphic approaches. In the future, we plan to develop gen-eral solutions that can work for arbitrarily partitioned data.

Acknowledgments

The work of Danish Mehmood and Basit Shafiq was sup-ported in part by the LUMS Departmental Research Grant. The work of Jaideep Vaidya and David Lorenzi was sup-ported in part by a Rutgers Business School—Newark and New Brunswick, Research Resources Committee Grant.

References

[1] J. Vaidya, C. Clifton, and M. Zhu, Privacy-Preserving Data Mining.ser. Advances in Information Security first ed., vol. 19, Springer-Verlag, 2005.

[2] W. Fan, H. Wang, P.S. Yu, and S. Ma, "Is Random Model Better? On Its Accuracy and Efficiency," Proc. Third IEEE Int'l Conf. Data Mining (ICDM '03), pp. 51-58, 2003.

[3] W. Fan, J. McCloskey, and P. S. Yu, "A General Framework for Accurate and Fast Regression by Data Summarization in Random Decision Trees," Proc. 12th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '06), pp. 136-146, 2006.

[4] X. Zhang, Q. Yuan, S. Zhao, W. Fan, W. Zheng, and Z. Wang, "Multi-Label Classification without the Multi-Label Cost," Proc. SIAM Int'l Conf. Data Mining (SDM '10), pp. 778-789, 2010.

[5] A. Dhurandhar and A. Dobra, "Probabilistic Characterization of Random Decision Trees," J. Machine Learning Research, vol. 9, 2321-2348, 2008.

[6] G. Jagannathan, K. Pillaipakkamnatt, and R.N. Wright, "A Practi-cal Differentially Private Random Decision Tree Classifier," Proc. IEEE Int'l Conf. Data Mining Workshops (ICDMW '09), pp. 114-121, 2009.

[7] J. Vaidya, C. Clifton, M. Kantarcioglu, and A.S. Patterson, "Privacy-Preserving Decision Trees over Vertically Partitioned Data," ACM Trans. Knowledge Discovery from Data, vol. 2, no. 3,1-27, 2008.

[8] O. Goldreich, "General Cryptographic Protocols," The Foundations of Cryptography, vol. 2, pp. 599-764, Cambridge Univ. Press, 2004.

[9] D. Gritzalis, Secure Electronic Voting.ser. Advances in Information Security first ed., vol. 7, Springer-Verlag, 2003.

[10] B. Schneier, Applied Cryptography. second ed., John Wiley & Sons, 1995.

[11] R. Cramer, I. Damgard, and J.B. Nielsen, "Multiparty Compu-tation from Threshold Homomorphic Encryption," Proc. Int'l Conf. Theory and Application of Cryptographic Techniques (EURO-CRYPT '01), pp. 280-299, May 2001.

[12] [12] P. Paillier, "Public Key Cryptosystems Based on Composite

[13] Degree Residuosity Classes," Proc. 17th Int'l Conf. Theory and

[14] Application of Cryptographic Techniques (EUROCRYPT '99), pp. 223- 238, 1999.

[15] [13] I.H. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann,Oct. 1999.

[16] [14] R. Agrawal and R. Srikant, "Privacy-Preserving Data Mining,"Proc. ACM SIGMOD Conf. Management of Data, pp. 439-450, May 2000.

[17] [15] D. Agrawal and C.C. Aggarwal, "On the Design and Quantification of Privacy Preserving Data Mining Algorithms," Proc. 20th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems, pp. 247-255, May 2001.

IOSR Journal of Computer Engineering (IOSR-JCE) is UGC approved Journal with Sl. No. 5019, Journal no. 49102.

Ms. Ch.Likitha Sravya. "Privacy-Preserving Data Mining with Random decision tree framework." IOSR Journal of Computer Engineering (IOSR-JCE) 19.4 (2017): 43-49.