ISSN:0975-9646

Protecting Frequent Patterns using Distributed Security on M-Clouds

K.H.Shabbeer Basha[1]
PG Scholar, Dept. of Computer Science & Engineering
Madanapalle Institute of Technology & Science
Madanapalle, Andhra Pradesh, India

E .Madhusudhana Reddy[2]
Professor, Dept. of CSE
Madanapalle Institute of Technology & Science
Madanapalle, Andhra pradesh, India

***Abstract*— As the age of big data evolves, outsourcing of data mining tasks to multi-cloud environments has become a popular trend. To ensure the data privacy in outsourcing of mining tasks, the concept of support anonymity was proposed to hide sensitive information about patterns. Existing methods that tackle the privacy issues, however, do not address the related parallel mining techniques. To fill this gap, we refer to a pseudo-taxonomy based technique, called as k-support anonymity, and improve it into multi-cloud environments with secrete sharing scheme. This has several advantages. First, outsourcing to multi-cloud environments can meet the requirement of great computational resources in big data mining, and also parallelize the mining tasks for better efficiency. Second, the data that we send out to a cloud can be partial. An assaulter who gets the data in one cloud can never re-construct the original data. That means it is more difficult for an assailant to violate the privacy in outsourced data. Experimental results also demonstrated that our approaches can achieve good protection and better computation efficiency.**

I. INTRODUCTION

As the age of big data evolves, cloud computing has attracted significant attention in recent years, for its great computing power, storage resources and services. By out-sourcing data and/or computation tasks to the cloud, a user can use the resources and services of cloud without knowing the details of techniques behind. The privacy, however, is always an issue, which stops users enjoying the benefits of cloud techniques. To eliminate the risk of privacy breach, many studies have been proposed in very recent years for secured outsourcing of computation and management. In this paper, we study the secured outsourcing of frequent itemset mining to *multi-cloud* environments for meeting the requirement of great computational resources in big data analysis. Multi-cloud environments in this paper refer to a set of clouds, where each cloud has its own power of computation, storage and framework, and performs tasks independently to the other clouds.For the problem of secured outsourcing of frequent item-set mining, it has been showed in previous works that the supports associated with items and itemsets could be used to re-identify hidden sensitive patterns, called support attacks. in [22] proposed the concept of k-support anonymity to limit the confidence of an item/itemset being re-identified to $1/k$ by adding noise patterns. They also introduced a taxonomy-based approach to reduce the storage overhead for anonymization. However, how to parallelize the mining tasks while protecting the privacy in the outsourcing of frequent itemset mining was not addressed in their study. Concerning the big data, we propose to incorporate k-support anonymity into multi-cloud environments for two-fold advantages. From the aspect of computation efficiency, parallelizing the mining tasks on multi-cloud environments can significantly speed up the computation; from the aspect of security, it is more difficult for an assailant to violate one's privacy since the data outsourced to a cloud can be partial.

Specifically, we aim to solve the problem of secured outsourcing of frequent itemset mining on multi-cloud environments. To parallelize the mining tasks, we first segment the whole data into several overlapping parts by sensitive items. Each part, given a partial set of sensitive items, consists of all the transactions containing its sensitive items, and thus provides complete support information about its sensitive items and partial supports about the other items. As a result, each part can calculate the frequent patterns of items with complete supports. The complete set of frequent patterns can be derived from the union of results in all parts. For satisfying the k-support anonymity, we adopt the taxonomy-based anonymization technique [22] to build a taxonomy tree with the items of complete support and include the items of partial support as noise. Experimental results demonstrated that our approaches can achieve good protection and better computation efficiency.

This paper is organized as follows. Section provides the preliminary to our work. In Section , we introduce the proposed algorithms. Evaluations are shown in Section . Finally, Section concludes this work.

II. PRELIMINARY

A. Frequent pattern mining

Concerning on the co-occurrences between items in a transactional database, the problem of frequent pattern mining is also called frequent itemset mining basically formalized as below.

Given a set I of items, a transactional database T_I contains a set of transactions t, $t \subseteq I$. In the problem of frequent itemset mining, a pattern p is a non-empty subset of I, and the support of p, denoted as $sup_{T_I}(p)$, is the number of transactions t in T_I such that $p \subset t$. A pattern p is said frequent if its support $sup_{T_I}(p)$ reaches a specified minimal threshold. The problem of frequent itemset mining discovers all the frequent patterns hidden in a transactional database T_I .

Sometimes the concepts of items, however, are related. For example, a rose can also be regarded as a flower. In the problem of **generalized frequent itemset mining**, a taxonomy tree consisting of items as it nodes is used to describe the relationships between the concepts of items. The leaf nodes represent the most specific concepts of items and the root is the most general concept of all items. The support of an internal item then comes from the occurrences of all its descendant items. The generalized frequent itemset mining discovers frequent patterns not only in the same level but also across levels. Therefore, the pattern {flower, diamond} will also be discovered if the pattern {rose, diamond} are frequent.

In this paper, we will use generalized frequent itemset mining to facilitate the secured outsourcing of frequent pattern mining on multi-cloud environments.

B. k-support anonymity

K-support anonymity [22] is an effective way to protect privacy in the outsourcing of frequent itemset mining. The concept of k-support anonymity is to create $k - 1$ fake items whose support is the same as some sensitive real item. As there are at least k items of the same support, the probability that an attacker can correctly re-identify the real item is then limited to $1/k$. The larger value of k is, the higher security level is for the real items.

In this paper, we refer to the way k-support anonymity with taxonomy tree [22], abbreviated as KAT, to achieve k-support anonymity for privacy protection. The concept of KAT is to generate k copies for each sensitive item so that an attacker does not know which item is (or is not) sensitive. In order to reduce the overhead in storage, KAT hides both the real and fake sensitive items in a pseudo taxonomy tree, in which support dependency exists between child-parent items (because a specific-concept item (such as 'rose') can also be regarded as a type of its parent item (such as 'flower')). Therefore, by utilizing the support dependency between items, KAT can increase the occurrences of a specific-concept item to increase both the supports of the specific-concept item (such as 'rose') and its relatively general-concept item (such as 'flower').

Specifically, KAT has two main steps. The first step constructs a pseudo taxonomy to hide the real items. In this step, the sensitive items are first randomly divided into k groups, and the items in each group are used to produce a sub-taxonomy tree by leaving items in the leaf nodes of the tree. The k sub-taxonomies are then strategically combined to build k-bud tree that facilitates the k-support anonymity in the second step. The second step alters the k-bud tree and uses alteration operations *insert*, *split*, and *increase* to generate fake items of specific supports. The *insert* operation inserts fake items at the internal level of the k-bud tree. The *split* operation raises the level of a leaf item x by adding two fake items y and z as its child nodes in the taxonomy, and replaces the occurrences of x in the transactions with y or z according to the specified supports. Finally, operation *increase* increases the occurrence of a leaf item to make the support of some fake item reach a specific value. Based on KAT, we then extend the secured outsourcing of frequent itemset mining to multi-cloud environments.

III. Distributed k-Support Noise Taxonomy Tree Algorithm

A. DKNT

In this work, we proposed a Distributed k-support Noise Taxonomy tree algorithm, abbreviated as DKNT, to conquer the privacy and the efficiency problems at the same time using the multi-cloud environment. The basic idea of DKNT is to divide the original database into overlapped partitions and send these partitions to different cloud platforms. Each cloud is responsible for a subset of items and DKNT will put transactions containing these items to the partition. Therefore, each cloud will have items which the cloud itself is responsible for and other items in these transactions. Before sending one partition to a cloud, DKNT will build a k-support Noise Taxonomy tree, abbreviated as KNT, for protecting the support privacy of all items being sent to the cloud. To protect sensitive items which the cloud is responsible for, the k-support anonymity tree is built first. To further protect other items being sent to the cloud together, DKNT generates some noise items which have similar supports to these items. After DKNT joins these items and noise items into the k-support anonymity tree, the k-support noise taxonomy tree is constructed. In this way, all items being sent to the cloud are protected. Then, DKNT transforms the original transactions according to the k-support noise taxonomy tree and sends the partition to the cloud. After the cloud gets the partition and the k-support noise taxonomy tree, the cloud can compute the generalized frequent patterns by any distributed algorithm on the cloud. Finally, combing all partial results from all clouds, DKNT can get the final results efficiently.

Suppose there are n cloud platforms to be utilized by users. Let C_i be the i_{th} cloud platform, where $1 \le i \le n$. All items in I are randomly partitioned into n groups. The set of items being assigned to C_i is denoted by X_{C_i} , and the set of transactions containing any item in X_{C_i} is denoted by T_{C_i} . Since each item is assigned to one of X_{C_i} and T_{C_i}

Algorithm 1 DKNT Algorithm

Require: n: number of clouds, min_sup: minimum support, k_{anony} : k-support anonymity, k_{noise}: number of $knoise$

Ensure: $C_1 \sim C_n$, $T_{NC1} \sim T_{NCn}$, $M(\cdot)_{C1} \sim M(\cdot)_{Cn}$, R_C : all generalized frequent patterns

1: Randomly partition real items into n clouds X_{Ci} , $i = 1, ..., n$
2: For i from 1 to n:
3: Let T_{Ci} = collect transactions containing items in X_{Ci}
4: Let C_i = ConstructKsupAnonTree(T_{Ci}, C_i ,X_{Ci} ,k_{anony})
5: Let $X_{\geq}$ = items of X_{Ci+} whose supports satisfy min_sup
6: Call CreateNoiseItem($X_{\geq}$, k_{noise})
7: Let $\{T_{NCi}$, C_i , $M(\cdot)_i\}$ = ConstructKNT(T_{Ci} , C_i , $X_{\geq}$)
8: Send T_{NCi} , C_i to C_i and get the sub result R_{Ci}
9: End for
10: Combine all sub-results and use $M(\cdot)_{C1} \sim M(\cdot)_{Cn}$ to get the final output R_C

is sent to C_i, the generalized frequent itemsets containing any items in X_{Ci} are calculated by C_i. Since all sensitive items are divided into one of X_{Ci} and are sent to different clouds, each cloud can only have less sensitive items, which increases the strength of the security.

From C_1 to C_n, every cloud C_i has to construct its own distributed k-support anonymity taxonomy tree C_i , encrypt mapping table $M(\cdot)_n$, and encrypted transactions T_{NCi} as shown by Algorithm 1. After getting the information of T_{Ci} and X_{Ci}, we can construct the corresponding k-support anonymity taxonomy tree using the same method proposed in [22]. In the k-support anonymity taxonomy tree of C_i, all sensitive items in X_{Ci} are protected by k-support anonymity.

In addition to items in X_{Ci} , there are more items in T_{Ci} sent together to C_i. The set of these items, which are not in X_{Ci}, is denoted by X_{Ci+}. Some items in X_{Ci+} are frequent and thus may contribute to form the generalized frequent patterns with items in X_{Ci} . These frequent items are denoted by $X_{\geq}$. Because only parts of transactions containing items in $X_{\geq}$ are sent to C_i, C_i can only know the partial support of items in $X_{\geq}$, which also increase the difficulty of reversing the sensitive items. To have better protection of items in $X_{\geq}$ from the support attack, we propose k-noise taxonomy tree algorithm, abbreviated as KNT, in the next section.

B. KNT

We cannot get the complete frequent patterns only by items in X_{Ci} and k-support anonymity taxonomy tree C_i . We have to include items in $X_{\geq}$ to the taxonomy tree. Since the transactions sent to C_i have only partial supports of items in $X_{\geq}$, we do not need to create as many fake items whose

Algorithm 2 CreateNoiseItem Algorithm

Require: X: itemset from C_i, k_{noise}

Ensure: items in X satisfy k-noise anonymity

1: For every unchecked $item_j$ in X:
2: While $Anonymity(item_j) < k_{noise}$:
3: Create a noise item $item_{noise}$ and let $Sup(item_{noise})$ close to $Sup(item_j)$
4: Put $item_{noise}$ into X
5: Mark $item_{noise}$ as checked
6: End while
7: Mark $item_j$ as checked
8: End for

Algorithm 3 ConstructKNT Algorithm

Require: T_{Ci} : transaction form C_i, C_i : k-bud tree of C_i, $X_{\geq}$: itemset from C_i which satisfy the minimum support, k_{noise}: noise support of C_i

Ensure: $T_{NCi} = T_{C_i}$, C_i , $M(\cdot)_{Ci}$

1: Let $T_{N_{Ci}} = T_{C_i}$
2: For every item $item_j$ in $X_{\geq}$:
3: Create $X_{\geq}$ having a $root$
4: Let $root.right = item_j$
5: Let $root.left = C_i$
6: Let $C_i = X_{\geq}$
7: End for
8: Generate an one-to-one mapping function $M(\cdot)_{Ci}$
9: Replace each $item$ in T_{NCi} with $M(item)_{Ci}$
10: Return $\{T_{NCi}$, C_i , $M(\cdot)_{Ci}\}$

supports are equal to the items in $X_{\geq}$ as in the k-support anonymity taxonomy tree. Instead, we relax the definition of k-support anonymity to k-noise anonymity, which means there should be at least k items with similar supports in a set. Therefore, we only generate noise items whose supports are similar to items in $X_{\geq}$. In this way, many items with similar supports could share these noise items and achieve k-noise anonymity. The algorithm is shown in 2. For example, if there is an item i with support 50, we can randomly create $k - 1$ noise items whose supports are between 48~52 and put them into the set. If there are more items in $X_{\geq}$ with similar supports, we can reduce the number of generated noise items significantly.

After we get the modified set of items $X_{\geq}$, which satisfies k-noise anonymity, we have to join these items into the k-support anonymity taxonomy tree C_i . The algorithm is shown in 3. For each item x in $X_{\geq}$, we build a new k-bud tree $X_{\geq}$. Let x be the right leaf of the root and let the current C_i as the left leaf of the root. After all the joins of items in $X_{\geq}$ are completed, we generate an one-to-one mapping function $M(\cdot)_{Ci}$ and encrypt each item x in T_{NCi} . Finally, the algorithm outputs the encrypted transaction set together with the altered $'_{Nci}$ taxonomy tree C_i and the

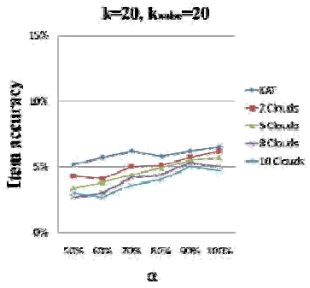


Figure 1. Item accuracy for the cases k = 20 and k_{noise} = 20.

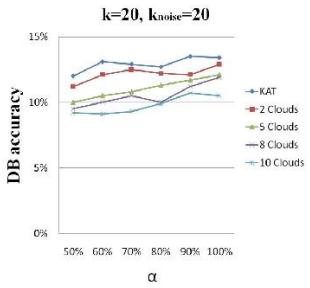


Figure 2. Database (DB) accuracy for the cases k = 20 and k_{noise} = 20.

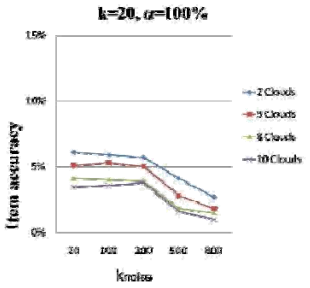


Figure 3. Item accuracy for the cases k = 20 and α = 100%.

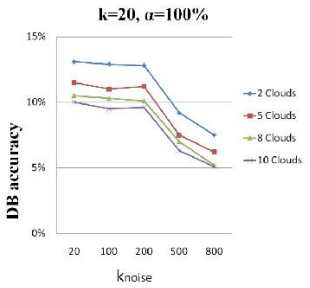


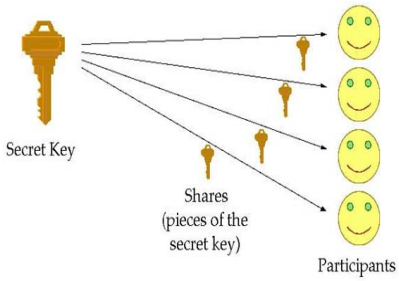
Figure 4. Database (DB) accuracy for the cases k = 20 and α = 100%.

mapping function $M(\cdot)_{Ci}$. The time complexity of 3 is $O(|\chi_{\geq}|)$.

When we have T_{NCi}, C_i, and $M(\cdot)_{Ci}$, we can simply send T_{NCi}, C_i, and the minimum support to C_i. When C_i get the all encrypted transactions containing all items in χ_{Ci}, C_i could apply any generalized frequent pattern mining method to find all generalized frequent patterns containing any items in χ_{Ci} as subresult R_{Ci}. After we receive R_{Ci}, by using $M(\cdot)_{Ci}$, we can transform these patterns back and drop the frequent patterns with the fake items. Then we can combine all sub-results and delete the duplicated frequent patterns to get the final result R_C.

Shamir's Secret Sharing Scheme:

Shamir's secret sharing scheme is a threshold scheme based on polynomial interpolation.It allows a dealer D to distribute a secret value s to n players, such that at least $t < n$ players are required to reconstruct the secret. The protocol is **information theoretically secure**, i.e., any fewer than t players cannot gain **any** information about the secret by themselves. To share the secret s among players $P_1, P_2, \ldots, P_n$ such that t players are required to reconstruct the secret.



Properties:

1. **Perfect Security –** information theoretic security. Given any t shares, the polynomial is **uniquely determined** and hence the secret a_0 can be computed. However, given t-1 or fewer shares, the secret can be any element in the field and thus those shares do not supply any further information regarding the secret.
2. **Ideal –** Each share is exactly the same size as the secret.
3. **Extendable –** additional shares may easily be created, simply by calculating the polynomial in additional points.
4. **Flexible –** can assign different weights (by the number of shares) to different authorities.

Sharing Protocol: To share the secret *s* among players $P_1, P_2, ..., P_n$ such that *t* players are required to reconstruct the secret

1. Dealer *D* creates a random polynomial *f*(*x*) of degree *t*-1 and constant term *s*.

$$f(x) = a_0 + a_1 x + \square + a_{t-1} x^{t-1}$$

The polynomial is developed over a limited field, such that the coefficient a0 is the mystery s and all different coefficients are arbitrary components in the field; the field is known to all members. Dealer's *D* publicly chooses *n* random distinct evaluation points: $X_j \neq \square 0$, and secretly distributes to each player P_j the share $share_j(s) = (X_j, f(X_j))$, *j=1...n*. (Remark: The evaluation point X_j could be any publicly known value, therefore for our convenience, we assume $X_j = j$, hence the shares are denoted as $f(1),..., f(j),..., f(n)$)

Reconstruction Protocol:
To reconstruct the secret from each subset of t shares out of n shares. Without loss of generality we will mark this subset: $f(1),..., f(t)$

1. Use Lagrange interpolation to find the unique polynomial *f*(*x*) such that $\deg f(X) < t$ and $f(j) = share_j(s)$ for *j=1,2,..t*
2. Reconstruct the secret to be *f(0)*.

Interpolation Property: Given t pairs of $(i, f(i))$, with *i*'s all distinct, there is a unique polynomial *f(X)* of degree *t-1*, passing through all the points. This polynomial can be effectively computed from the pairs $(i, f(i))$.

Lagrange interpolation:

$$f(x) = \sum_{i=1}^{t} f(i) * L_i(X)$$

$$L_i(X) = \frac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} (x_i - x_j)}$$

where $L_i(X)$ is the Lagrange polynomial:
which has value 1 at X_i, and 0 at every other X_j.

Detection of Corrupted Shares:

In an ace dynamic mystery offering framework, taking an interest shareholders must have the capacity to verify whether shares of different shareholders have not been ruined or lost, and restore the right impart if important. Something else, a foe could result in the loss of the mystery (by crushing n-(t-1) shares). The objective in this segment is to present an instrument for location of tainted shares.There are clear circumstances in which there is a high likelihood that the offer is demolished, e.g. a circle crash, yet how would anybody figure out that a programmer infiltrated his/her machine, uncovered his/her impart and transformed it? The thought is to spare some unique mark for each one impart that is regular to all the shareholders, so occasionally, shareholders can hope to measure up shares (utilizing secure telecast).

With a specific end goal to actualize the appropriated irrefutability of shares, a fundamental peculiarity is added to the past convention. In each one time period, every shareholder stores the encryptions of every last one of shares he/she got from alternate shareholders. This is accomplished as takes after:

- Perform the non-interactive VSS, so the encryption of the initial shares will be stored at each shareholder.
- Using the homomorphic property, each *i*'th shareholder updates his/her set of encrypted shares by computing for every j: $E(h(i)) = E(f(i)) * \prod_{m=1}^{n} E(P_m(j))$. Actually, this product is computed using only update shares corresponding to well behaved shareholders.

IV. EXPERIMENT

This section evaluates the security and cost effective-ness of our approaches. The programs are implemented in *Python*. All experiments are performed on an Arch GNU/Linux server with Intel(R) Core(TM) i7 CPU 860 @ 2.8 GHz Opteron processors and 32*GB* RAM.

The testing data, *T*10*I*1*kD*100*k* dataset, is syntheti-cally generated by the IBM data generator. Specifically, T10I1kD100k dataset contains 100k transactions, 1000 dif-ferent items, and about 400 frequent itemsets. The average transaction length is 10. In the experiments, we assume all
the items are sensitive.

A. Security Analysis

To evaluate the security of our approaches, we implemented genetic algorithm [33] to simulate attackers knowing item support information as his background knowledge. The risk of privacy leakage is studied from both item and database aspects defined below: (1) item accuracy is defined as the ratio of the items being correctly re-identified according to the attacker's knowledge; (2) database accuracy is the average ratio of the content of an transaction being correctly revealed. As the impact of *k*-support anonymity has been explored in [22],

First, we evaluate the privacy risk when data are par-titioned and outsourced to different numbers of clouds. Figures 1 and 2 explore the item accuracy and database accuracy, respectively, where α denotes the ratio of frequent items together with the corresponding support information known to the assaulter. The experimental results show that, from the both item and database aspect of privacy protection, partitioning and outsourcing the data to multi-cloud environ-ments are always better than outsourcing the whole data to a single party, and the more partitions the data is divided into, the better the privacy protection is. This is because the

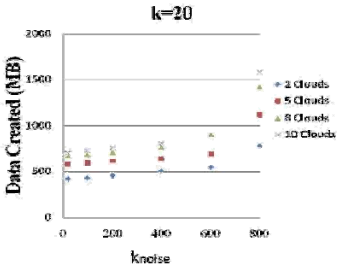


Figure 5. The overall data size of encrypted database when $k = 20$ and k_{noise} varies.

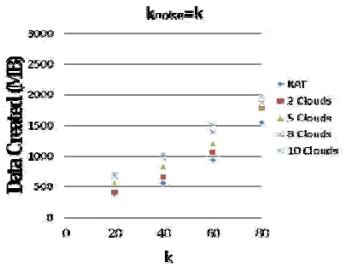


Figure 6. The overall data size of encrypted database when $k_{noise} = k$.

data is partial on each cloud and an assaulter who gets the data in one cloud can never re-construct the original data. Therefore, it is more difficult for an assailant to violate the privacy in outsourced data. Second, we study the impact of k_{noise} for privacy protection. Figures 3 and 4 shows the item and database accuracy under different values of k_{noise}. Generally, the privacy protection will become stronger when more noise is introduced.

B. Cost of Encryption

In this subsection, we study the storage overhead of our methods on the T10I1kD1000k datasets.

Figure 5 shows the storage overhead as a function of k_noise. The results show that the storage overhead increases when k_{noise} increases. In addition, when the number of clouds increases, the overall data size of the encrypted database also grows for a specific k and k_{noise}. This is because a transaction containing items in different clouds will have additional copies for the clouds, i.e., one additional copy for each cloud. Note that, however, the burden in storage can be a relatively small cost compared to the benefit from the computation efficiency. For example, the computation efficiency in a 10 multi-cloud environment could be 5 times of that in a 2 multi-cloud environment while the data size grows only a bit.

In Figure 6, we compare the storage efficiency of our methods to that of KAT method by setting k_{noise} equal to k. As expected, the more the number of clouds is, the larger the storage overhead will be. However, as what we explained in the experiment of Figure 5, we just earn much more computation resources by scarifying a little storage space.

on a pseudo-taxonomy based anonymization technique [22], called KAT, we proposed DKNT to ensure the privacy security for each partial data outsourced to different clouds. Experimental results demonstrated that our approaches can achieve good protection and better computation efficiency, compared to the computation efficiency on single machine.

V. CONCLUSION

In this paper, we studied the problem of secured out-sourcing of frequent itemset mining on the multi-cloud en-vironments. Concerning the challenges in big data analysis, we suggested to partition the data into several parts, and outsourced each part independently to different cloud. Based on a pseudo-taxonomy based anonymization technique [22], called KAT, we proposed DKNT to ensure the privacy security for each partial data outsourced to different clouds. Experimental results demonstrated that our approaches can achieve good protection and better computation efficiency, compared to the computation efficiency on single machine.

REFERENCES

[1] C. C. Aggarwal and P. S. Yu, *On static and dynamic methods for condensation-based privacy-preserving data mining*, ACM Transactions on Database Systems, 2008.

[2] R. Agrawal and R. Srikant, *Privacy-preserving data mining*, In Proc. of ACM Special Interest Group on Management Of Data, 2000.

[3] R. Buyya, C. S. Yeo, and S. Venugopal, *Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities*, In Proc. of Canadian Society for the Study of Education, pages 10-1016, 2008.

[4] L. Cao, P. S. Yu, C. Zhang, and H. Zhang, *Data Mining for Business Applications*, Springer, 2008.

[5] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, *rivacy preserving mining of association rules*, In Proc. of ACM Special Interest Group on Knowledge Discovery and Data Mining, 2002.

[6] F. Giannotti, L. V. Lakshmanan, A. Monreale,D. Pedreschi, and H. Wang, *Privacy-preserving mining of association rules from outsourced transaction databases*, In Workshop on Security and Privacy in Cloud Computing, 2010.

[7] J. Han and Y. Fu, *Discovery of multiple-level association rules from large databases.*, In Proc. of Very Large Data Base, 1995.

[8] M. Kamber and J. Han, *Data Mining: Concepts and Tech-niques*, Morgan Kaufmann; 2nd Edition, 2005.

[9] M. Kantarcioglu, R. Nix, and J. Vaidya, *An efficient approxi-mate protocol for privacy-preserving association rule mining*, In Proc. of Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2009.

[10] N. Li, T. Li, and S. Venkatasubramanian, *t-closeness:Privacy beyond k-anonymity and l-diversity*, In Proc. of International Conference on Data Engineering, 2007.

[11] K. Liu and E. Terzi, *Towards indentity anonymization on graphs*, In Proc. of ACM Special Interest Group on Manage-ment Of Data, 2008.

[12] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkita-subramaniam, *l-diversity: Privacy beyond k-anonymity*, ACM Transactions on Knowledge Discovery from Data, 1(1), 2007.

[13] A. Maurizio, B. Francesco, G. Fosca and P. Dino, *Anonymity preserving pattern discovery*, Very Large Data Base, 2008.

[14] T. Mielik inen, *Privacy problems with anonymized a trans-action databases*, In Proc. of Discovery Science, 2004.

[15] J. Pei and B. Zhou, *Preserving privacy in social networks against neighborhood attacks*, In Proc. of International Con-ference on Data Mining series, 2008.

[16] L. Qiu, Y. Li, and X. Wu, *Preserving privacy in association rule mining with bloom filters*, J. Intell. Information Systems, 29(3):253278, 2007.

[17] M. D. Singh, P. R. Krishna and A. Saxena, *A cryptography based privacy preserving solution to mine cloud data*, COM-PUTE, 2010.

[18] R. Srikant and R. Agrawal, *Mining generalized association rules*, In Proc. of Very Large Data Base, 1995.

[19] X. Sun and P. S. Yu, *A border-based approach for hiding sensitive frequent itemsets*, In Proc. of International Conference on Data Mining series, 2005.

[20] C. H. Tai, P. S. Yu, and M. S. Chen, *k-Support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining*, Knowledge Discovery and Data Mining, 2010.

[21] W. K. Wong, D. W. Cheung, E. Hung, B. Kao, and N. Mamoulis, *Security in outsourcing of association rule mining*, In Proc. of Very Large Data Base, 2007.

[22] F. Yu, C. Zhiyuan, K. Gunes and G. Aryya, *A privacy protection technique for publishing data mining models and research data*, In Proc. of ACM Special Interest Group on Management Of Data, 2008.

[23] F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng, *Mining colossal frequent patterns by core pattern fusion*, In Proc. of International Conference on Data Engineering, 2007.

[24] G. Wang, Q. Liu, F. Li, S. Yang, and J. Wu, *Outsourcing Privacy-Preserving Social Networks to a Cloud*, In Proc. of IEEE INFOCOM, 2013.

[25] F. Kerschbaum and J. Vayssiere, *Privacy-Preserving Data Analytics as an Outsourced Service*, In Proc. of ACM Secure Web Services, 2008.

[26] C. Wang, S. Chow, Q. Wang, K. Ren, and W. Lou, *Privacy-Preserving Public Auditing for Secure Cloud Storage*, In Proc. of IEEE INFOCOM, 2010.

[27] M. Bentounsi, S. Benbernou, and M. J. Atallah, *Privacy-Preserving Business Process Outsourcing*, In Proc. of IEEE on Web Services, 2012.

[28] K.-P. Lin and M.-S. Chen, *Privacy-preserving outsourcing support vector machines with random transformation*, In Proc. of KDD, 2010.

[29] I. Molloy, N. Li, and T. Li. On the (in)security and (im)practicality of outsourcing precise association rule mining. In *Proc. of ICDM*, 2009.

[30] W. K. Wong, D. W. Cheung, E. Hung, B. Kao, and N. Mamoulis. Security in outsourcing of association rule mining. In *Proc. of VLDB*, 2007.

[31] R. Spillman, M. Janssen, B. Nelson, and M. Kepner, *Use of a genetic algorithm in he crytanalysis of simple substitution ciphers*, Cryptologia, XVII(1), pp. 31-44, 1993.

[32] K.Kishore, E. Madhusudhana Reddy, (2012), "Outsourcing In Cloud Computing Using Homomorphic Encryption Potentials" International journal of Advanced Scientific and technical Research (ISSN: 2249-9954), August 2012, Issue 2, Volume 4, PP 57-64.

[33] *P. Suneel Kumar, E.Madhusudhana Reddy,* (2012), International Conference on Science and Information Technology (ICSIT-2012), Organized by A STAR india, Deharadun, (June 3, 2012) PP.7-12.ISBN:978-93-81693-92-6.Title – Dynamic Pricing Scheme for Cloud Cache to Maximize the Service Provider.