Securing Mining of Sequence Patterns from Redistributed Data Set Using Association Rule Mining

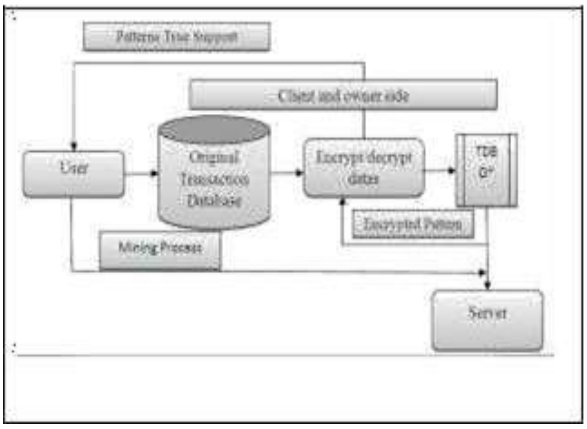
Dr.K.Bhargavi[1], Dr.A.Ramachandra Reddy [2], Dr.D.Kiran kumar[3], Mr.V.NarsingRao[4]
[1,2,3,4] *Sphoorthy Engineering College*

***Abstract*- There has been significant recent interest in the paradigm of data mining as-a-service. A company (data owner) lacking in proficiency or computational resources can outsource its mining needs to a third party service provider (server). In spite of this, both the items and the association rules of the outsourced database are considered private property of the company (data owner). To protect corporate or individuals privacy, the data owner transforms its data and ships it to the server, sends mining queries to the server, and recovers the true patterns from the extracted patterns received from the server. In this paper, experiment evaluation of outsourcing the association rule mining task within a corporate privacy-preserving framework. Proposed an attack model based on background knowledge and devise a approach for privacy preserving outsourced mining.**

Represented approach ensures that each transformed item is indistinguishable with respect to the attacker's background knowledge, from at least k−1 other transformed items. These comprehensive experiments on a very large and real transaction database demonstrate that these techniques are effective, scalable, and protect privacy.

Index Terms- Association Rule Mining, Privacy Preserving Outsourcing.

I. INTRODUCTION



The problem of outsourcing the association rule mining task within a corporate privacy preserving framework is difficult. A construction body of work has been done on privacy preserving data mining (PPDM) in a variety of framework. A common attribute of most of the earlier studied frameworks is that the patterns mined from the data (which may be unclear, encrypted, anonymized or transformed) are intended to be shared with parties other than the data owner. The key peculiarity between such bodies of work and problem is that, in the latter, both the underlying data and the mined results are not intended for sharing and must remain private to the data owner. For this adopted a traditional frequency based attack model in which the server knows the exact set of items in the owner's data and additionally, it also knows the exact support of every item in the original data. Wong et al. was one of the near the beginning works on defending against the frequency based attack in the data mining outsourcing scenario. Wong et al. introduced the idea of using fake items to guard against the frequency based attack; however, it was lacking a formal theoretical analysis of privacy guarantees, and has been shown to be flawed very recently in , where a method for breaking the proposed encryption is given. Therefore, in Giannotti et al. previous and preliminary work, Giannotti et al. proposed to solve this problem by using k-privacy, i.e., each item in the outsourced dataset should be indistinguishable from at least k − 1 items regarding their support. In this paper, the goal is to devise an encryption approach which enables formal privacy guarantees to be proved, and to validate this model over large-scale real-life transaction databases (TDB). The architecture behind this model. The client/owner encrypts its data using an encrypt/decrypt (E/D) module, which can be essentially treated as a black box from its perspective. This model is responsible

for transforming the input data into an encrypted database. The server conducts data mining and sends the (encrypted) patterns to the owner. This encryption approach has the property that the returned supports are not true supports. The E/D module recovers the true identity of the returned patterns as well their true supports. It is trivial to show that if the data are encrypted using 1–1 substitution ciphers (without using fake transactions), many ciphers and hence the transactions and patterns can be broken by the server with a high probability by launching the frequency-based attack. Thus, the major focus of this paper is to devise encryption approaches such that formal privacy guarantees can be proven against attacks conducted by the server using background knowledge, while keeping the resource requirements under control. First, formally defining an attack model for the adversary and make the background knowledge the adversary may possess precise. The notion of privacy requires that, for each cipher-text item, there are at least k−1 distinct cipher items that are indistinguishable from the item regarding their supports. Secondly, developed an encryption approach, called Rob Fruga,l that the E/D module can employ to transform client data before it is shipped to the server. Third, to allowing the E/D module to recover the true patterns and their correct support, it is proposed that it creates and keeps a compact structure, called synopsis. For this providing the E/D module with an efficient strategy for incrementally maintaining the synopsis against updates in the form of appends. Also conducted a formal analysis based on this attack model and prove that the probability that an individual item, a transaction, or a pattern can be broken by the server can always be controlled to be below a threshold chosen by the owner, by setting the anonymity threshold k. This result holds unconditionally for the Rob Frugal approach. Then conduct experimental analysis of this schema using a large real dataset from the Coop store chain in Italy. The result shows that these encryption schemas is effective, scalable, and achieve the desired level of privacy. Providing here the key theoretical results which concern the complexity and privacy guarantees the privacy. Also discusses the results of a comprehensive set of experiments conducted using real and synthetic datasets.

II. MODEL PRIVACY

Consider denote the original TDB that the owner has. to protect individual identification of items, when the owner applies an encryption function to and transforms it to ,the encrypted database. Refer items in as plain items and items in as cipher items. The term item shall mean plain item by default. The notions of plain item sets, plain transactions, plain patterns, and their cipher counterparts are defined in the obvious way. use to denote the set of plain items and to refer to the set of cipher items.

A) Adversary Knowledge: The server or an opponent might gains access to it may possess some prior or background knowledge using which opponent can conduct attacks on the encrypted database. Generically refer to any of these agents as an attacker. Here adopting a conventional model and assuming that the attacker knows exactly the set of (plain) items in the original and their true supports in, i.e., , . The attacker may have access to similar data from a competing organization, may read published reports, etc. In reality, the opponent may be having approximate knowledge of the supports or may know the exact/approximate supports of a subset of items in D. However, to make the analysis robust, one can adopt the conventional assumption that he knows the exact support of every item. Remember, that as the opponent has access to the encrypted database D*, he also knows the supports, where is the set of cipher items in the encrypted database D*.

1)Replacing each plain item in D by a 1–1 substitution cipher and

2)Adding fake transactions to the database.

Consider, that the opponent knows this and thus he knows that | | = |I|. Essentially, compared to, adversary knowledge model corresponds to a (100%, 0%) knowledge model, confined to single items. Assume that the opponent neither has the knowledge of plaintext transactions nor the frequency of item sets and the distribution of transaction lengths in the original database.

b)Attack Model: By assuming that the service provider (who can be an opponent) is semi-honest in the sense that although he does not know the details of encryption algorithm, opponent can be curious and he can use his prior knowledge to make assumption on the encrypted transactions. Also considered, that the opponent always returns (encrypted) item sets

together with their exact support. The data owner considers the true identity of:

1) Every cipher item;
2) Every cipher transaction;
3) Every cipher frequent pattern;

As intellectual property which should be protected. The following attack model will considers all these.

Refer to prob(e) and prob(E) as crack probabilities. From the point of view of the owner, minimizing the probabilities of crack is desirable. Intuitively, Cand(e) and Cand(E) should be as large as possible. Ideally, Cand(e) should be the whole set of plaintext items. This can be achieved if one can bring each cipher item to the same level of support, e.g., to the support of the most frequent item in D. Unfortunately, this option is impractical, as it will lead to a large size of the fake transactions, which in turn leads to a dramatic explosion of the frequent patterns and making pattern mining at the server side computationally prohibitive.

This motivates us of relaxing the equal-support constraint and introducing item k-anonymity as a compromise.

c) Problem Statement: To compute the privacy guarantees of an encrypted database, Here define the following notion.

Definition 2: Given a database D and its encrypted version D*, say D* is k-private if:

Problem studied: Given a plain database D, construct a k-private cipher database D* by using substitution ciphers and adding fake transactions such that from the set of frequent cipher patterns and their support in D* sent to the owner by the server, the owner can reconstruct the true frequent patterns of D and their exact support. Additionally, like to minimize the space and time incurred by the owner in the process and the mining overhead incurred by the server.

III. ENCRYPTION/DECRYPTION APPROACH

A. Encryption: In this section, introducing the encryption approach, called Rob Frugal, which transforms a TDB D into its encrypted version D*. This approach is parametric with respect to $k > 0$ and consists of three main steps:

1)Using 1–1 substitution ciphers for each plain item;
2)Using a specific item k-grouping method; and
3)Using a method for adding new fake transactions for achieving k-privacy.

The constructed fake transactions are added to D (once items are replaced by cipher items) to form D *, and transmitted to the server. A record of the fake transactions, i.e., , is stored by the E/D module in the form of a compact synopsis, as discussed in previous sections.

B. Decryption: When the client requests the execution of a pattern mining query to the server, specifying a minimum support threshold σ, the server returns the computed frequent patterns from D*.

Clearly, for every item set S and its corresponding cipher item set E, have that $\leq$. For each cipher pattern E returned by the server together with, the E/D module recovers the corresponding plain pattern S. It needs to reconstruct the exact support of S in D and decide on this basis if S is a frequent pattern. To achieve this goal, the E/D module adjusts the support of E by removing the effect of the fake transactions (E)., this follows from the fact that support of an item set is additive over a disjoint union of transaction sets. Finally, the pattern S with adjusted support is kept in the output if $\geq \sigma$. The calculation of (E) is performed by the E/D module using the synopsis of the fake transactions in D * \D.

The proposed encryption/decryption approach is a viable solution for privacy-preserving pattern mining over outsourced TDB, provided that a correct and efficient implementation exists. On the efficiency side, it is not practical to store the support (E) for every cipher pattern. In order to realize the encryption approach efficiently, need to address the following technical issues.

1) How do cluster items into groups of ?
2) How do create the needed fake transactions?
3) How is the synopsis represented and stored?

C. Grouping Items for k-Privacy

Given the items support table, several strategies can be adopted to cluster the items into groups of size k. Start from a simple grouping method called Frugal. By assuming the item support table is sorted in descending order of support and refer to cipher items in this order as , etc.

Definition 3: The Frugal method consists of grouping together cipher items into groups of k adjacent items

in the item support table in decreasing order of support, starting from the most frequent item . Assume . . . is the list of cipher items in descending order of support (with respect to D), the groups created by Frugal are {e1, . . . , }, {, . . . , }, and so on. The last group, is less than k in size, is merged with its previous group. By denoted the grouping obtained using the above definition as G frug. For example, consider the example TDB and its associated (cipher) item support shown in Fig. 2. For k = 2, has two groups: {} and { , , }. This corresponds to the partitioning groups shown in Table I(a). Thus, in D *,the support of will be brought to that of ; and the support of and brought to that of . Given the fact that the support of the items strictly decreases monotonically,

Frugal grouping is optimal among all the groupings with the item support table sorted in descending order of support. This means, it minimizes ‖G‖, the size of the fake transactions added, and hence the size ‖D*‖. But is Frugal a robust grouping, i.e., will it guarantee that itemsets (or transactions) cannot be cracked with a probability higher than k ? The answer is no, in general. To see this point, consider the item support table in Table I: the first group created by Frugal for k = 2, {} [see Table I(a)] is supported in D, because occur together in a transaction of D. Therefore, there only exists one itemset candidate of {e2, e4}, i.e., the privacy guarantee is 1-privacy.To fix the privacy vulnerabilities of Frugal, introduce the RobFrugal grouping method, which modifies Frugal by requiring that no group is a supported itemset in D.

Definition 4: Given a TDB D and its Frugal grouping = , the grouping method RobFrugal consists in modifying the groups of Gfrug by repeating the following operations, until no group of items is supported in D:

1)select the smallest such that find the most frequent item such that, for the least frequent item i of have: swap with in the grouping.

For example, given the item support table in Fig. 2, the grouping illustrated in Table I(b), obtained by exchanging e4 and e5 in the two groups of Frugal, is now robust: none of the two groups, considered as itemsets, is supported by any transaction in D. The aim of Step 2 in Definition 4 is to obtain a robust grouping while maintaining as small as possible the number of fake transactions that are added to achieve k-privacy. In particular, It will show the information about fake transactions can be maintained by the data owner using a compact synopsis. This step is used to ensure the synopsis is as small as possible.

The key property of Rob Frugal is that, by construction, it is a robust grouping for any input TDB D. It is immediate to note that if the support in D of each group of the initial grouping Gfrug is 0, then RobFrugal produces a robust and optimal grouping, where optimal means that it minimizes the number of the fake transactions that are created by this encryption approach. On the other hand, it should be noted that a grouping according to Rob Frugal may not exist, depending on the extent of density in the TDB. For example, in a TDB where each pair of items occurs at least once together, Rob Frugal will not find a grouping for k = 2. In this case, a simple solution is to keep increasing the value of k until a Rob Frugal grouping approach exists. The intuition is that as k gets larger it is less likely that there is a real transaction containing all items in a group. However, with a large k, the number of fake transactions increases. This affects storage and processing at the server side although the data owner can always maintain information about fake transactions using a compact synopsis of size n being the number of items. In practice, It has been found that even for small values of k = 10 to 50, a Rob Frugal grouping approach does exist. This was the case in all these experiments with real transaction data. In the Rob Frugal encryption approach, the output of grouping can be represented as the noise table. It extends the item support table with an extra column “Noise” indicating, for each cipher item e, the difference among the support of the most frequent cipher item in e’s group and the support of e itself, as reported in the item support table. Denoted the noise of a cipher item e as N(e). Continuing the example, the noise table obtained with Rob Frugal is reported in Table II(a). The noise table represents the tool for generating the fake transactions to be added to D to obtain D*.

IV. CONSTRUCTING FAKE TRANSACTIONS

Given a noise table specifying the noise N(e) needed for each cipher item e, for this the fake transactions generated as follows. First, it drop the rows with zero

noise, corresponding to the most frequent items of each group or to other items with support equal to the maximum support of a group. Then the remaining rows are sorted in descending order of noise. Let , . . . , be the obtained ordering of (remaining) cipher items, with associated noise N(), . . . ,N(). The following fake transactions are generated:

1) N() − N() instances of the transaction {};
2) N() − N() instances of the transaction {, };
3) . . . ;
4) N (−1) − N () instances of the transaction {, . . . , };
5) N() instances of the transaction {, . . . , }.

Suppose cipher items of nonzero noise in Table II(a). The following two fake transactions are generated: two instances of the transaction {} and one instance of the transaction {}. Note that even though the attacker may know the details of the construction method, he/she is not able to distinguish these fake transactions from the true ones, since the attacker does not have any background knowledge of frequency of item sets or of original transaction length distribution. It can be shown that this method yields a minimum number of different types of fake transactions that equal the number of cipher items with distinct noise. This observation yields a compact synopsis for the client of the introduced fake transactions. The purpose of using a compact synopsis is to reduce the storage overhead at the side of the data owner who may not be equipped with sufficient computational resources and storage, which is common in the outsourcing data model. In order to implement the synopsis efficiently, need to use a hash table generated with a minimal perfect hash function [16]. Minimal perfect hash functions are widely used for memory efficient storage and fast retrieval of items from static sets. A minimal perfect hash function is a perfect hash function that maps n keys to n consecutive integers, usually. Hence, h is a minimal perfect hash function over a set S if and only if implies and there exists an integer p such that the range of h is . A minimal perfect hash function h is order-preserving if for any keys j and i, j < i implies h(j) < h(i). In this approach, the items of the noise table with N() > 0 are the keys of the minimal perfect hash function. Given function h computes an integer in denoting the position of the hash table storing the triple of values _, timesi, occi, where timesi represents the number of times that the fake transaction {e1, e2, . . . , }occurs in the set of fake transactions, and occi is the number of times that ei occurs altogether in the future fake transactions after the transaction {, , . . . , }. Given a noise table with m items with non null noise, this approach generates hash tables for the group of items. In general, the ith entry of a hash table HT containing the item ei has = N() − N(), =, where g is the number of items in the current group. Note that each hash table HT represents concisely the fake transactions involving all and only the items in a group of g ≤ items. The hash tables for the items of nonzero noise in Table II(a) are shown in Table II(b). Finally, use a (second-level) ordinary hash function H to map each item e to the hash table HT containing e. Note that after the data owner outsources the encrypted database (including the fake transactions), he/she does not need to maintain the fake transactions in its own storage .Instead the data owner only has to maintain a compact synopsis, which stores all the information needed on the fake transactions, for later recovery of real supports of item sets. The size of the synopsis is linear in the number of items and is much smaller than that of the fake transactions.

With the above data structure, one can define the function RS that allows an efficient computation of the real support of a pattern E = {, , . . . , }with fake support s as follows: where: i) is the item in E such that for 1 ≤ j ≤ n, have h(ej) ≤ h(), and ii) HT = H() is the hash table associated by H to any item ei of E. For example, in Table I(b), for = {}, , whereas for = { , }, , where is the fake support of . This is exactly right since is fakely added three times while is fakely added two times.

V. EXPERIMENTS

In this section, presenting report on experimented empirical evaluation to assess the encryption/decryption overhead and the overhead at the server side incurred by the proposed schema.

A. Datasets: For analysis purpose experimented on a large real-world database. The real world database is donated to us by Coop, a cooperative of consumers that is today the largest supermarket chain in India. Selected the transactions occurring during four periods of time in a subset of Coop stores, creating in this way four different databases with varying number of transactions: from 100k to 300k

transactions. In all the datasets the transactions involve 15713 different products grouped into 366 marketing categories. Transactions are itemsets, i.e., no product occurs twice in the same transaction. Assume two distinct kinds of TDBs:

1) Product-level Coop TDBs, denoted by Coop-Prod, where items correspond to products, and

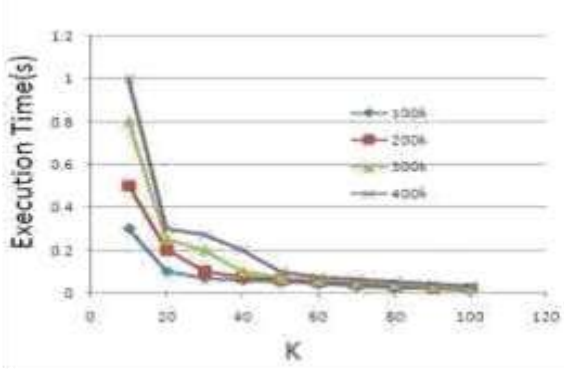
2) Category level Coop TDBs that is denote by CoopCat, where items correspond to the category of the products in the original transactions. In these datasets, = 188 for CoopProd, while = 90 for CoopCat. Also, the two kind of TDBs exhibit very different density properties. The number of frequent patterns found in CoopCat tends to explode for higher support thresholds, compared to CoopProd. Also experimented with this algorithms for both CoopProd and CoopCat.

B. Experimental Evaluation: Implemented Rob Frugal encryption & decryption approach in Java. All experiments were performed on an intel Core2 Duo processor with a 2.66 GHz CPU and 3GB RAM over a Win 7 platform. Adopted the a priori implementation by Christian Borgelt,2 written in C and one of the most highly optimized implementations.

VI. ENCRYPTION OVERHEAD

First assessed the total time needed by the ED module to encrypt the database (grouping, synopsis construction, creation of fake transactions): timings are reported in Fig. 3 for CoopProd and CoopCat, for different values of k and different number of transactions. The results show that the encryption time is always small; it is under 1 s for the biggest CoopProd TDB, and below 0.8 s for the biggest CoopCat TDB. Indeed, it is always less than the time of a single mining query, which is at least 1 s by Apriori, as shown in Fig. 4(d). Therefore, when there are multiple mining queries, which is always the case for the outsourcing system; the encryption overhead of this approach is negligible compared with the cost of mining. It is worth noting that these experiments provide empirical evidence that the theoretical complexity upper bound of O(n2) is indeed over pessimistic. To see this point, counted the number of queries (to check that each group is unsupported) performed by the ED module (Rob Frugal), over the two TDBs for the different values of k, and discovered that such number always coincides with n k , except for CoopCat TDBs in the cases k = 10 and k = 20: for example, for k = 10 and number of transactions 400K (the biggest TDB), an additional 3790 item swaps are needed to find a robust grouping and only 10 for k = 20. This is a strong empirical evidence that in real life databases Rob Frugal reaches a solution very fast, with complexity far below the O(n2) worst case: e.g., for CoopCat with k = 10 and 400 transactions, Rob Frugal only needs to check a total of 3826 queries, while 3662 = 133, 956! Second assessed the size of fake transactions added to the databases after encryption. It is observed that the size of fake transactions increases linearly with k. It is observe that density affects the generation of fake transactions: e.g., have that CoopProd∗, for k = 30, is only 8% larger than CoopProd while, for the same k, CoopCat∗ is 80% larger than CoopCat. The size of the fake transactions on synthetic databases is assessed. The overhead of incremental encryption is also assessed , which occurs when a new TDB is appended; to this end, then split CoopProd with 500k transactions into two-halves and , and treat as the original TDB and as the appended one.

Encryption Overhead on CoopProd



VIII. CONCLUSION AND FUTURE WORK

In this paper, presenting a conservative model result , where the adversary knows the domain of items and their exact frequency and can use this knowledge to identify cipher items and cipher itemsets. An encryption approach is proposed, called *Rob Frugal* that is based on 1–1 substitution ciphers for items and adding fake transactions to make each cipher item share the same frequency as $\geq k-1$ others. It makes use of a compact synopsis of the fake transactions

from which the true support of mined patterns from the server can be efficiently recovered. A strategy for incremental maintenance of the synopsis against updates consisting of appends and dropping of old transaction batches is also proposed. Unlike previous works, such as and ,this formally proved that this method is robust against an adversarial attack based on the original items and their exact support. Our experiments based on both large real and synthetic datasets yield strong evidence in favor of the practical applicability of this approach. Currently, this privacy analysis is based on the assumption of equal likelihood of candidates. It would be interesting to enhance the framework and the analysis by appealing to cryptographic notions such as perfect secrecy.

REFERENCES

[1] W. K. Wong, D. W. Cheung, E. Hung, B. Kao, and N. Mamoulis, "Security in outsourcing of association rule mining," in Proc. Int. Conf.Very Large Data Bases, 2007, pp. 111–122.

[2] L. Qiu, Y. Li, and X. Wu, "Protecting business intelligence and customer privacy while outsourcing data mining tasks," Knowledge Inform. Syst.,vol. 17, no. 1, pp. 99–120, 2008.

[3] C. Clifton, M. Kantarcioglu, and J. Vaidya, "Defining privacy for data mining," in Proc. Nat. Sci. Found.

[4] Workshop Next Generation Data Mining, 2002, pp. 126–133.

[5] Fosca Giannotti, Laks V. S. Lakshmanan, Anna Monreale, Dino Pedreschi, and Hui (Wendy) Wang," Privacy-Preserving Mining of Association Rules From Outsourced Transaction Databases", IEEE

[6] SYSTEMS JOURNAL, VOL. 7, NO. 3, SEPTEMBER 2013,pg385-395.

[7] I. Molloy, N. Li, and T. Li, "On the (in)security and (im)practicality of outsourcing precise association rule mining," in Proc. IEEE Int. Conf Data Mining, Dec. 2009, pp. 872–877.

[8] F. Giannotti, L. V. Lakshmanan, A. Monreale, D. Pedreschi, and H. Wang, "Privacy-preserving data