This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2484318, IEEE Transactions on Services Computing

Target-Based, Privacy Preserving, and Incremental Association Rule Mining

Madhu V. Ahluwalia, Aryya Gangopadhyay, Zhiyuan Chen and Yelena Yesha

Abstract— We consider a special case in association rule mining where mining is conducted by a third party over data located at a central location that is updated from several source locations. The data at the central location is at rest while that flowing in through source locations is in motion. We impose some limitations on the source locations, so that the central target location tracks and privatizes changes and a third party mines the data incrementally. Our results show high efficiency, privacy and accuracy of rules for small to moderate updates in large volumes of data. We believe that the framework we develop is therefore applicable and valuable for mining big data.

Index Terms— Association rules, data mining, mining methods and algorithms, security, integrity, and protection

1 Introduction

Association rule mining has been studied extensively in the context of preserving privacy of raw data and sensitive rules. With the single exception of [1], past work on privacy preserving association rule mining has focused on one-time mining. And even though the authors in [1] study private and incremental mining, they do so for non-quantitative data. To date, no work has been published on incrementally mining association rules with privacy protection when the data upon which mining occurs is quantitative and subject to change.

We study this problem against the backdrop of supply chain management where a data owner's operational data is scattered over multiple source sites, but collected at a single target site. Consider for example, large grocery or apparel chains in the United States, such as Giant or Land's End with several retail outlets (source sites/databases), but a single centralized data warehouse (target site/database), that outsource the task of mining to a consulting firm (a data miner).

Four past techniques form the basis of our work. One of these uses ***Discrete Wavelet Transform*** (***DWT***) to mask the original data in such a way that a majority of the original association rules are preserved [2].

Discrete Wavelet Transform is a mathematical transform that attempts to capture the maximum energy (i.e. sum of the squared signal values) in discrete numerical signals. There are many discrete wavelet transforms. The simplest one is the Haar wavelet transform. It lies at the core of our work. To understand the Haar wavelet transform, consider a one-dimensional signal of length $n = 4$ given by a discrete function $f(x) = (10\ 2\ 4\ 5)$ [1]; its 1st and 2nd - level Haar transform computation is shown in Fig. 1. The 1st-level transform, (6 4.5), is obtained by taking the average[2] of (10 2) and (4 5) respectively. The differences of (10 2) and (4 5) divided by two computes to (4 –0.5) respectively. Note that these computations are performed on the signal at full resolution. The averages and differences are designated as wavelet coefficients of the transformed data with averages being referred to as approximations and differences the details. While the details at level 1 are stored, the approximations become inputs in the level 2 transform. The process is applied iteratively on approximations until only one approximation and $n-1$ details over $\log n$ scales or resolutions are left.

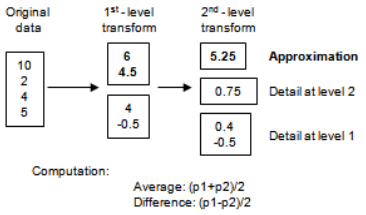


Fig. 1. One-dimenaional Haar DWT transform. We show how Haar transfom is computed for a single attribute. DWT has the property to preserve underlying data patterns even when it suppresses low energy coefficients such as the detail.

The underlying idea in [2] is to sort the original data on each attribute, perform one level of Haar transform on each sorted attribute, and finally retain and replicate the strong (or approximation) coefficients to use them in the mining operation. Ahluwalia, et al. [2] implement this idea through the privatization algorithm. Fig. 2 illustrates this algorithm. Fig. 2(a) shows the original data consisting of 6 tuples and two sensitive attributes, age and salary. Fig. 2(b) shows all rows sorted by the first attribute, age. Fig.

- *M.V. Ahluwalia is with Harris Corporation, 2235 Monroe Street, Herndon, VA 20171. E-mail: madhu.ahluwalia@harris.com.*
- *A. Gangopadhyay is with the Department of Information Systems, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250. E-mail: gangopad@umbc.edu.*
- *Z. Chen is with the Department of Information Systems, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250. E-mail: zhchen@umbc.edu.*
- *Y.Yesha is with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250. E-mail: yeyesha@csee.umbc.edu.*

[1] f(x) represents a one-dimensional vector (x) as opposed to a two-dimensional vector (x, y).

[2] Haar wavelet uses 2, but to preserve energy the standard practice is to use sqrt 2.

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2484318, IEEE Transactions on Services Computing

ID	Age	Slry (K)
1	66	36
2	32	55
3	50	34
4	48	100
5	30	65
6	22	40

(a)

ID	Age	Slry (K)
6	**22**	40
5	**30**	65
2	**32**	55
4	**48**	100
3	**50**	34
1	**66**	36

(b)

ID	Age	Slry (K)
6	**26**	40
5	**26**	65
2	**40**	55
4	**40**	100
3	**58**	34
1	**58**	36

(c)

ID	Age	Slry(K)
3	58	**34**
1	58	**36**
6	26	**40**
2	40	**55**
5	26	**65**
4	40	**100**

(d)

ID	Age	Slry (K)
3	58	**35**
1	58	**35**
6	26	**47.5**
2	40	**47.5**
5	26	**82.5**
4	40	**82.5**

(e)

Fig. 2. Example of executing the privatization algorithm. In our previous work, we take a heuristic approach to show that this algorithm allows preserving both the patterns and the privacy of the original data.

2(c) shows the Haar transformed and duplicated values of age with the detail coefficients discarded. Fig. 2(d) shows all rows sorted by the second attribute, salary. Fig. 2(e) shows the duplicated Haar approximation coefficients of salary with the detail coefficients discarded. The dataset in Fig. 2(e) is fully transformed and ready to be sent to an untrusted server for mining purposes. Note that in [2], the data upon which mining occurs is largely at rest in the sense that it does not change. Since the data does not change, it is transformed (or privatized) once and mined once. This mining model is shown in Fig. 3.

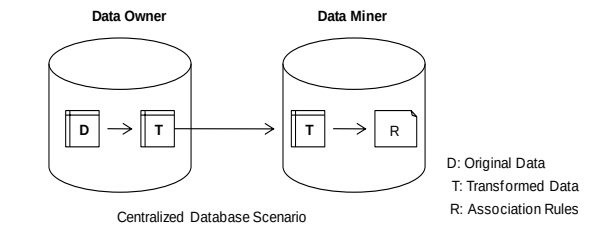


Fig. 3. Static Privacy Preserving Association Rule Mining. This is a one-time mining model because the collection of data does not increase, decrease or change over time.

We extend the privacy preserving model illustrated in [2] to data in transition. This transition features change that occurs across source sites. An efficient way to track changes in very large volumes of transactional data is proposed in [3]. Known as ***Target-Based Database Synchronization*** (***TBDS***), this approach requires a target system to synchronize itself with the source systems by tracking only the differential in large source systems. Hash values for corresponding partitions of the source and the target databases are compared so that if these values differ, the tuples in the target partition can be dropped and reloaded from the source. A table of partition definition guides this process.

In a different work [4], the TBDS algorithm is applied in conjunction with the DWT algorithm to prove that discrete wavelet transformation over changed data is viable because it generates highly accurate association rules and maintains data privacy when few changes occur between the source and the target. The approach used is called ***Wavelet Coefficient Maintenance*** (***WCM***).

Our aim is to use changed data that is transformed with DWT to mine association rules safely and incrementally. We borrow ideas from aforementioned techniques and a technique known as ***Fast Update 2*** (***FUP2***) [5] to generate frequent itemsets and rules incrementally in a large target centric environment.

The essence of this paper is captured in Fig. 4. At the onset, the data owner holds data synchronized with all its

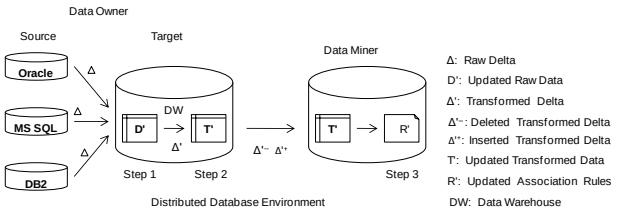


Fig. 4. Dynamic Privacy Preserving Association Rule Mining. This figure shows the steps in incremental privacy preserving association rule mining that is controlled by a target system.

source systems as well as a transformation of this data; the data miner holds a copy of this transformation and the association rules generated on the transformation. A target system now seeks to incrementally capture changes across source systems as they occur. These changes are denoted by the symbol Δ and we call them raw deltas because they occur in the original data. The target is synchronized with the source upon capturing all of the raw deltas. We denote synchronized data in the target by the symbol D' and call it updated raw data. See step 1 in Fig. 4. The target system applies Haar transform on the raw deltas. The transformed deltas are denoted by Δ'. It then updates its first transformation with the transformed deltas. The updated transformed data is denoted by T'. See step 2 in Fig. 4. The target system exchanges only the deleted transformed deltas, Δ'^{-} and inserted transformed deltas, Δ'^{+} with the miner. Finally, the data miner uses these to update his first set of association rules. These are therefore the securely and incrementally updated association rules denoted by R'. See step 3 in Fig. 4.

We assume that relatively few changes occur in the source databases and that they allow external systems only read access to data. Further we assume that the schema is identical in all source databases. Finally, we assume that both source and target databases as well as a third party mining engine are accessible for queries throughout the period in which rules are updated. Ideally, the solution is efficient, minimizes its demands on the source and target CPUs, minimizes its demands on the source and target primary and secondary memory, minimizes the data sent across the network that connects the source and target, minimizes the data that is exchanged between the data owner and the data miner and interoperates between diverse database management systems (DBMSs).

A brute-force target-based solution has the target database retrieve and transform all records in each table from the source database (full replication and full transformation). It then has the data miner compute rules over all transformed records (full rule generation) which provides for simplicity but may retrieve, transform and generate rules on much more data than necessary when there are few actual changes between large source and target databases.

We propose a target-based rule maintenance solution while preserving the privacy of raw deltas as well as synchronized data. We assume that after the very first synchronization, the target database secures a table of full

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2484318, IEEE Transactions on Services Computing

partition information (summary of initial synchronization between the source and the target systems) and full transformation (replicated wavelet coefficients on first synchronization) on its own space. The miner (adversary) in turn secures a copy of the first full transformation, and a table with counts of all first frequent itemsets computed over the first full transformation on his space.

In the presence of changes to the source, the target database uses the content of the partition information table to retrieve complete raw tuples from the source database. We aim to decrease the number of retrieved tuples to decrease the number of tuples to transform and exchange with the adversary; for each table in the source database, the target database horizontally partitions the total order of tuples based on the partition start and end markers (row IDs of previous synchronization) of the partition information table, summarizes the tuples in each partition with a hash of all data in the partition, and matches this hash summary with the corresponding hash values in the table of partition definition. If the summaries match then we assume that the source table has not been subject to any change and nothing needs to be exchanged between the owner and the miner. If the summaries do not match then the target database retrieves all tuples in the partition's range from the source database, sorts each continuous-valued attribute in ascending order, applies Discrete Wavelet Transform over it, and stores and duplicates the approximation coefficient; thus providing for column-wise 2-anonymity in the attribute's transformed domain. The transformed partition consisting of deleted and inserted deltas, is then passed to the data miner who combines it with his very first mining result to efficiently compute the new set of frequent itemsets and rules.

We summarize our contributions as follows. We present the first study on privacy preserving association rule mining of fully-dynamic quantitative *source* databases, which can be modified by any sequence of insertions and deletions and which do not use their own resources to identify, extract, transfer and integrate the changes in a *target* database that keeps fully synchronized copies of their datasets. We believe that this problem has significant relevance in many application areas such as retail, health and finance.

Our proposed solution integrates in a single framework, hash-based change detection, wavelet transform and Apriori based techniques to incrementally update association rules while preserving data privacy. We present comparisons of accuracy, privacy and efficiency of the proposed approach with relevant privacy preserving data mining techniques (see section 5.1). Heuristics demonstrate the superiority of our method.

Section 2 refines the problem statement and describes related solutions. Section 3 defines and reviews some basic concepts. Section 4 provides the details of our algorithm and establishes some performance expectations on our solution relative to copying and transforming the entire source database on the target and generating rules from scratch. Section 5 presents empirical tests of our algorithm. Section 6 analyzes the privacy our technique offers before concluding in Section 7.

2 Problem Definition And Related Work

2.1 Problem Definition

We consider the problem of incrementally maintaining quantitative association rules in a dynamic environment while preserving the privacy of original data that is subject to change. In this environment the onus of tracking changes and privatizing changes is upon the target database and the onus of incrementally mining association rules is upon an adversarial database. The target and the adversarial databases need not be empty when rule maintenance starts. The updated rules must match the rules generated on the updated original data. The transformed deltas sent to the miner should not reveal the raw deltas nor should the transformed deltas plugged in the previous transformation reveal synchronized data.

In this instance of the problem, the data owner (the target database) seeks to minimize the load on the source databases. Security concerns prevent source databases to allow access to their transaction logs or to alter their database schema or to add triggers. The target database must therefore initiate and detect the actual change, transform these changes and pass them to a third party mining engine for incremental mining. The source databases simply responds to queries from the target database, much like any other client application that is accessing the source database.

2.2 Related Work

There is a rich body of work on privacy preserving association rule mining for categorical data [6], [7], [8], Boolean data [9], in a distributed mining environment [10], [11], in an environment where data is centralized [12], and where rules instead of data values are hidden [6], [7], [8], [9], [13], [14], [15], [16], [17], [18], [19], [20], [21].

The literature is also rife with work to selectively update frequent itemsets [22], [23], [24], [25], [26], [27]. Most of this work addresses a variation of the problem than the one under consideration in this paper. For instance, in [27], the authors are interested in incremental and interactive high utility pattern mining, where the practical usefulness of frequent itemsets to the user is important. The users' selections on the temporal aspect of the data to mine are explored in [26]. In [24], the problem of mining frequent itemsets on dynamic and distributed databases in different parallel and distributed environments is addressed. One past work deals with the problem of determining when to update the current model of association rules [28], whereas another updates the model after an arbitrary number of updates [25].

Techniques that are more conducive to the model proposed in this work are described in [5], [28], [29]; these are all Apriori based. The problem of maintaining association rules in a centralized environment is studied in [5], [29]. Maintenance of association rules in a distributed environment is studied in [1]. However, incremental techniques to mine association rules with privacy protection in a fully dynamic centralized database have not been researched in the past. Nor does the literature contain work on incrementally mining privacy preserving association rules using target-only resources.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2484318, IEEE Transactions on Services Computing

3 PRELIMINARIES

3.1 Defining Privacy for Data Mining

Typically, privacy is viewed in the context of individual data; Alice must not know that Bob's annual salary is x amount of dollars or that he owns a property in downtown Washington D.C. If Alice cannot trace the salary/property attribute values to Bob, Bob's privacy is protected. Here, privacy is seen as the ability to identify an individual. However, there is another view. Here, the aim is to learn from a body of data. The adversary gets to know the overall patterns and trends from the data collection. Known as corporate privacy in contrast with individual privacy, an adversary must be prevented from breaching this privacy. Both views apply to data mining – the former falls under "data hiding" and the latter under "rule hiding". Our approach to privacy is categorized as a data hiding approach [30]. Our goal is to hide sensitive values in quantitative data.

3.2 Association Rule Mining

We use Apriori, the classic algorithm for learning association rules. The algorithm first finds all the sets of items that appear frequently enough to be considered significant and it then derives from them the association rules that are strong enough to be considered interesting. The user determines the level of interestingness by specifying the minimum support threshold and minimum confidence threshold. Formal definitions of these metrics for an association rule of the form $X \rightarrow Y$ are

$$\frac{|X \cup Y|}{N} \quad \text{and} \quad \frac{|X \cup Y|}{|X|} \quad \text{respectively.}$$

X and Y are disjoint itemsets and N is the number of transactions in a dataset. Thus, while the support is a simple probability or a measure of the frequency of a rule, the confidence is a conditional probability or a measure of the strength of the relation between sets of items.

3.3 Discrete Wavelet Transforms

Discrete wavelet transform is used to analyze data by decomposing it into different frequency components and then studying each component with a resolution matched to its scale. We use the Haar wavelet due to its simplicity. For an input represented by a list of 2^n numbers, the Haar wavelet transform simply pairs up input values, storing the difference and passing the sum. The difference is called the detail coefficients and the sum is called the approximation coefficients[3]. This process can be repeated recursively over approximation coefficients in the previous level. In our experiments, we apply one level of Haar wavelet transform.

We use DWT in our algorithm to preserve patterns and privacy of the original data. Our motivation of using DWT is based on several properties of DWT.

1. DWT allows a multi-resolution representation of discrete data. Important patterns in the high resolution data are usually preserved in the low resolution data (i.e. in approximation coefficients).
2. We can prune detail coefficients to provide privacy protection. Without the detail coefficients, the exact values of the original data cannot be reconstructed from the transformed data because there would be infinitely many solutions towards solving for the original data from the approximation coefficients.
3. DWT can be done very efficiently (in $O(n)$ time for n input data points).

Challenge of Using DWT in Our Context: The main challenge of using DWT in the context of privacy preserving association rule mining is the order dependency of wavelet transforms. In other words, completely different sets of coefficients are generated if the dataset is ordered differently. We have devised a method that optimally groups similar data values together so that association rules can be preserved. The key idea is to sort the data on each attribute so that within-group data homogeneity [2] is maximized.

3.4 Incremental Approaches

Increments pose a major challenge in large systems. One can always resort to naïve brute-force approaches that ignore knowledge previously discovered and replicate work already performed. However, this would result in an explosion in the amount of computational and I/O resources required. Therefore all incremental approaches essentially exploit previous results. While synchronizing databases requires a summary of previous synchronization (the partition information) and updating transforms requires that all previous wavelet coefficients are available, association rule maintenance relies on counts of all previous frequent itemsets. We use these update approaches in conjunction to propose a target-based method to incrementally maintain association rules while preserving the privacy of updated data. Fig. 5 shows an overview of our approach.

Fig. 5. Incremental Approaches in Privacy Preserving Association Rule Mining. This figure outlines the process with which we mine rules incrementally and securely.

It is important to note that while data elements that constitute raw deltas, the previous synchronization and the wavelet transform are all at the same grain in the owner's database, this is not true for the data elements that constitute the miner's database. The miner's repository consists of transformed deltas and previously mined frequent items. These are both at different levels of granularity, in the sense that frequent item sets are summarized information while transformed deltas are not. This makes the task of maintaining association rules in large datasets more complex than the task of maintaining raw and transformed data.

[3] We divide the sum and difference by two in this paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2484318, IEEE Transactions on Services Computing

4 METHODOLOGY

We have devised a method to incrementally mine association rules on data that is subject to change. This data resides in several source systems that may be limited by space, time and security constraints. A target system that is designated as a single point of data collection must then rely on its own resources to detect the change and incrementally mask the data for mining purposes. A third party data mining engine in turn must receive the incrementally masked data and use it to incrementally maintain association rules. We show that association rules generated from the incrementally masked data are meaningful, the privacy of the incrementally masked data is high and the process of achieving both is efficient. We describe our Incremental Rule Update algorithm in Section 4.1 and outline factors that affect its performance in Section 4.2. Section 4.3 presents limitations of our algorithm.

4.1 Target-Based Privatized Incremental Rule Update Algorithm

The algorithm combines, in a single framework, techniques for efficiently capturing and hiding modified data and for incrementally preserving patterns in large systems. In our incremental rule update algorithm, the target database poses queries to the source database to retrieve information and tuples. It then transforms sets of updated tuples to exchange them with a third party data mining engine, which uses them to incrementally generate association rules.

We describe the ***Target-Based Privatized Incremental Rule Update*** (***TB-PIRU***) algorithm in the context of a single database table. Multiple tables can be processed using one instantiation of the algorithm for each table, whether executed sequentially or concurrently. We assume that the source and the target databases are already synchronized and the target holds on its space a table of partition information of the current synchronization. The partition information refers to partition start and end markers (row IDs) and hashes of the current synchronization. Before releasing the present synchronization to a miner, the target perturbs it and stores a copy of this perturbation on its space. The miner runs the Apriori algorithm on this perturbation and stores both the perturbation as well as the counts of frequent itemsets on his database. We now seek to directly update these rules in the presence of updates to the source.

The essence of the algorithm, shown in Fig. 6, is to partition the source database table into variable-sized partitions using the partition start and end markers of the partition definition table, and then to compute the hash value of all tuples in each source partition. If the hash of the source partition differs from its counterpart in the partition definition, retrieve the partition from the source, sort it on each dimension in ascending order and apply the Haar wavelet transform on the sorted dimension. Retain only the approximation coefficient which is an average of two closest values. Since there are

Algorithm: TB-PIRU

Input: Source Database - S: Dataset, Δ: Changes in S
Target Database (Data Owner) - r : Number of tuples in a partition, ***p*:** number of partitions, $t_{r(t1-1)+1} \ldots t_{rt1}$, $t_{r(t2-1)+1} \ldots t_{rt2}$, ...: partition ranges and h_{t1}, h_{t2}, ...: their corresponding hashes
Data Miner - T: Copy of previous transformation, Δ^- : transformed deleted delta, Δ^+: transformed added delta, F_k : number of all previous frequent *k*-itemsets, ***s*%:** minimum support, *c*%: minimum confidence

Output: F'_k: Updated frequent *k*-itemsets and **R'**: updated association rules

1. **for** i = 1 to *p* **do**
   1.1 Create an SQL query Q to extract all tuples $\geq t_{r(i-1)+1}$ and $< t_{rt+1}$ on S, catenate tuples into a local buffer B, and return hash(B)
   1.2 Send Q to S and retrieve the query result in h_{st}
   1.3 **if** (h_t != h_{st}) **then** -- *This condition indicates there is* Δ
      1.3.1 Create an SQL query Q' to extract and return all tuples $\geq t_{r(i-1)+1}$ and $< t_{rt+1}$ on S
      1.3.2 Send Q' to S and retrieve the resulting tuples
      1.3.3 **for** each column $C_j \in s_{r(i-1)+1} \ldots s_{rt}$ **do** -- *This step gives* Δ'
         1.3.3.1 Sort $s_{r(i-1)+1} \ldots s_{rt}$ by column C_j
         1.3.3.2 Perform DWT, store and duplicate only approximation coefficients
         1.3.3.3 **if** r is odd in $s_{r(i-1)+1} \ldots s_{rt}$ **then**
            1.3.3.3.1 rotate or eliminate last tuple
         1.3.3.4 **end if**
      1.3.4 **end for**
      1.3.5 Send tuples with updated coefficients from step 1.3.3 to the data miner
   1.4 **end if**
2. **end for**
3. For all tuples from S smaller than t_1 and all tuples from S larger than t_n, apply steps 1.3.3 to 1.3.5 -- *These are rows in S whose PK values are* $< t_1$ *and* $> t_n$
4. Update F_k to F'_k by scanning Δ^- and Δ^+
5. Generate R' on F'_k. Retain rules that meet the *c*% threshold.

Fig. 6. Target-Based Privatized Incremental Rule Update algorithm. This algorithm enumerates steps that were followed to update association rules privately and incrementally using a target-centric approach.

n rows but only $n/2$ coefficients in each partition dimension, copy each coefficient to two consecutive rows. Send the new coefficients to the data miner.

The owner may not update his previous transformation with new coefficients. However, we assume that he does. This allows retrieving from his database transaction logs, sets of all deleted and newly added transformed tuples (Δ'^- and Δ'^+ respectively) since the last transformation. Tuples with modified values for only a few attributes are considered as deletes followed by inserts. New coefficients are sent to the miner in this format. Tuples consisting of deleted and newly added coefficients are shared with the miner without row IDs.

The miner may also choose not to update his previous transformation. Again, we assume that he does. Being malicious, he attempts to find the original values by plugging the transformed deltas in the previous transformation, but is unable to do so because he does not have the row IDs.

Since association rules can be generated efficiently from the support counts of previous frequent itemsets F_k, the miner keeps track of itemsets rather than rules. He generates association rules incrementally by combining previous frequents itemsets with sets of deleted and inserted tuples.

Let $t_1 \ldots t_n$ be the tuples in the target database's synchronized table and $s_1 \ldots s_n$ be the tuples in the source database's table totally ordered by their primary keys.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2484318, IEEE Transactions on Services Computing

Let hash() be a hash function common to both databases, such as SHA-1. Choose r to be the number of tuples in a partition and let $p = \lceil n/r \rceil$ be the number of partitions in the target database. Fig. 6 then describes the algorithm as executed at the target database.

Depending on the implementation, the core of the TB-PIRU algorithm can be divided into six separate stages with each stage of the first three stages within a loop that iterates over all partitions: computing the hashes in the source database (steps 1.1 and 1.2), retrieving the updated records when hashes fail to match (steps 1.3.1 and 1.3.2), computing wavelet coefficients (steps 1.3.3 and 1.3.4), sending the new coefficients to the data miner (steps 1.3.5), updating previous frequent itemsets by scanning the deleted and inserted deltas (step 4) and generating rules that satisfy the minimum confidence on new frequent itemsets (step 5). Our implementation computes and updates rules immediately as differences are detected between source and target partitions. Note that all tuples inserted in S that are smaller than t_1 and larger than t_n will have no corresponding matches in the target. Therefore hash computation is not required for these tuples. Note also that steps 1-3 are processed at the target (owner's) site and steps 4-5 are processed at miner's site.

Fig. 7 illustrates the algorithm shown in Fig. 6. We assume that the source and the target were once synchronized. Let the partition size be four rows per partition. The target database then stores on its space, tables of partition definition (*Partition Information of S*)[4], and transformation (*Transformed Dataset T*) of the initial synchronization (*Source Dataset S*). A copy of the transformation with rows IDs pruned off (*Miner's Transformed Dataset T*) is exchanged with the miner. The source now undergoes some change (*Updated Partition in S denoted by* Δ). Hash values identify that partition 1 has changed, as row 2 is deleted and row 3 is updated. Hence the changed partition is sorted on each attribute and a mean is computed over two consecutive values and replicated. The values in the odd row are rotated. The transformed partition (*Transformed Delta denoted by* Δ') is then plugged in the previous transformation (*Updated Transformation*). This allows extracting from the log files of the target all deleted, Δ'^{-}, and newly computed coefficients, Δ'^{+}, (bottom far right). These coefficients are shared with the data miner to update rules incrementally. Note that all relations in Fig. 7 have the same attributes as the Source Dataset S.

Source Dataset S

RowID	Quantity	Price	Discount
1	25.00	126	0.16
2	19.00	76.95	0.12
3	8.00	49.99	0.00
4	27.00	119.5	0.17
5	15.00	51.99	0.15
6	6.00	32.45	0.05
7	47.00	150.1	0.21
8	18.00	64.25	0.13
9	35.00	105.9	0.30
10	5.00	15.25	0.10

Partition Information of S

PartID	RowID	Hash
1	1	H1H1
2	5	H2H2
3	9	H3H3

Transformed Dataset T

RowID	Quantity	Price	Discount
3	11.5	50.99	0.03
6	5.5	23.85	0.03
10	5.5	23.85	0.11
2	18.5	70.6	0.11
8	18.5	70.6	0.14
5	11.5	50.99	0.14
1	26	138	0.17
4	26	112.7	0.17
7	41	138	0.26
9	41	112.7	0.26

Miner's Transformed Dataset T

Quantity	Price	Discount
11.5	50.99	0.03
5.5	23.85	0.03
5.5	23.85	0.11
18.5	70.6	0.11
18.5	70.6	0.14
11.5	50.99	0.14
26	138	0.17
26	112.7	0.17
41	138	0.26
41	112.7	0.26

Updated Partition in S (Δ)

1	25.00	126	0.16
3	11.00	69.99	0.10
4	27.00	119.5	0.17

Transformed Delta (Δ')

3	18.00	94.74	0.13
1	18.00	97.99	0.13
4	19.00	94.74	0.14

Updated Transformation T'

3	**18.00**	**94.74**	**0.13**
6	5.5	23.85	0.03
10	5.5	23.85	0.11
8	18.5	70.6	0.14
5	11.5	50.99	0.14
1	**18.00**	**97.99**	**0.13**
4	**19.00**	**94.74**	**0.14**
7	41	138	0.26
9	41	112.7	0.26

Deleted Transformed Delta (Δ'^{-})

18.5	70.6	0.11
11.5	50.99	0.03
26	138	0.17
26	112.7	0.17

Inserted Transformed Delta(Δ'^{+})

18.00	94.74	0.13
18.00	97.99	0.13
19.00	94.74	0.14

Fig. 7. Example of executing the TB-PIRU algorithm with r=4 and p=3. This figure uses quantitative data to illustrate how we exchange data with a third party to mine association rules securely and incrementally.

4.2 Performance Factors

The performance of TB-PIRU will depend on factors related with synchronization, privatization and computational complexity of the Apriori algorithm (since FUP2 computes frequent itemsets on sets of inserted and deleted tuples like Apriori).

Synchronization is affected by the distribution of changes in the source database. In one extreme where the source and target database are already identical, the algorithm will compute and exchange p hash values, which comprises less data to exchange than copying the entire source database if the average length of r tuples is longer than the hash value. In the other extreme where every partition contains some modified tuple in the source database, the algorithm will exchange more data than simply copying the entire source database to the target. Specifically, the overhead consists of the set of p partition hashes (and the corresponding queries to create the hashes) and p SQL queries to retrieve the tuples in each partition. Synchronization therefore favors fewer changes to numerous changes in the source.

Privatization occurs by first sorting each attribute individually and then computing and duplicating coefficients. If intermediate sort results are stored for each attribute in order to correctly compute and duplicate coefficients, the cost of the privatization grows quadratically with the number of attributes. If coefficients are computed and duplicated directly with the help of an array that stores row IDs in sort order after the matrix is sorted on each attribute, the cost of privatization scales linearly with the number of records and attributes. Due to higher space and time overheads of the former, privatization favors the latter approach. We call this approach ***In-Place DWT*** (***IP-DWT***) [2].

Computational complexity of Apriori takes a hit with large dataset size, high dimensionality and the number of passes over the dataset, among other factors. FUP2 optimizes primarily by minimizing the impact of these factors. It prunes the size of datasets to scan. It prunes the number of candidate new frequent itemsets to evaluate. Finally, it prunes the number of total database scans.

4.3 Algorithm Limitations

There are three main limitations to our algorithm. First, our approach is meaningful only for data with continuous attributes. When the data has categorical attributes, existing techniques [7], [8], [35] can be used in conjunction with our proposed approach.

Second, hash collisions, although rare for high bit-length hash functions, are possibilities in every instance where a large space (tuples in a partition) is summarized by a smaller space (hash values). If any risk of a collision is unacceptable for the task then one should resort to exchanging the raw tuples or lossless compressions of the tuples. If

[4] In the table *Partition Information of S*, "H1H1", "H2H2" and "H3H3" represent concatenating hashes of three different partitions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2484318, IEEE Transactions on Services Computing

there is some tolerance for the risk of a hash collision but the probability of collision for a hash function is too high then one can decrease the risk by one or a combination of concatenating hashes from different hash functions over the same data or by using overlapping or multiple partition schemes.

Third, if the hash function can only handle a limited number of characters then multiple insertions into the source database into the middle of a partition could have the source database's partition size exceed the hash capacity. One can mitigate this limitation by first retrieving the number of tuples within the partition at the source database and only asking for the hash of the source's partition if the number of tuples in the partition matches between the target and the source.

5 Experimental Evaluation

Our experiments indicate that our proposed approach not only performs incremental updates on rules efficiently when few changes are made to the raw data, but it also keeps the incrementally maintained association rules very similar to the association rules generated from incrementally updated raw data. In so doing, it keeps the privacy level of the incrementally updated raw data as well as raw deltas sufficiently high. Both the source and target databases are operational, while updates take place on rules. Our method causes no overhead for the source and does not alter it in any way.

Section 5.1 describes the setup. Section 5.2 presents the results for pattern preservation. Section 5.3 reports the degree of privacy offered. Section 5.4 reports the execution time. Results on accuracy, privacy and performance obtained from our approach are compared with results obtained from existing benchmark approaches.

5.1 Setup

Change detection and extraction was conducted using source and target databases installed on two separate machines. The database synchronization algorithm was implemented in JAVA version 1.6. The privatization algorithm was implemented using Matlab R2007a. For association rule mining, we used Oracle Data Miner 10.2.

Datasets: The experiments were run over four real datasets. Datasets "Adult" and "Pima Indians" were obtained from UCI Machine Learning Repository [31]. Datasets "Custom Checking" and "Sales Fact" are from the "Bank" and "Grocery" databases in the Ralph Kimball collection of databases [32]. All experiments were performed using numeric attributes.

Privacy preserving algorithms: Two approaches kd-tree and random projection serve as benchmarks to evaluate some parameters. Kd-tree is a micro-aggregation technique to mask data and therefore bears resemblance to our wavelet based approach. However, it has not been used in privacy preserving association rule mining [33]. We use it to compare accuracy and privacy obtained from our approach. Random projection is a dimensionality reduction technique that has been used to preserve correlations between attributes [34]; however it cannot be used to generate association rules because attributes in the transformed domain do not correspond to attributes in the original domain. Therefore random projection is used as a benchmark only in privacy comparisons.

Data mining parameters: A minimum support of 0.1 and 0.3 and a minimum confidence of 0.5 and 0.7 were used for all datasets.

Rule quality metrics: We compared rules obtained from incrementally updated original data with incrementally mined rules using the metrics recall and precision.

Recall is defined as $\frac{|\{D'_R\}\bigcap\{R'\}|}{|\{D'_R\}|}$

and precision is defined as $\frac{|\{D'_R\}\bigcap\{R'\}|}{|\{R'\}|}$

where $\{D'_R\}$ denotes the set of rules obtained from incrementally updated original data, $\{R'\}$ denotes the set of incrementally mined rules, and $\{D'_R\}\bigcap\{R'\}$ denotes the set of rules common to both.

Bin sizes: To avoid overfitting, association analysis allows discretizing continuous attributes. Discretization replaces each continuous attribute value with its corresponding discrete interval. If the number of intervals is too large, there may be too few records in each interval to satisfy the minimum support and minimum confidence levels. As a result few rules are generated. On the other hand, if the number of intervals is too small, then some intervals may aggregate records which would otherwise generate unique rules. The process of specifying the number of intervals is referred to as binning. We use equiwidth binning with bin sizes 2, 5 and 8 to study the effect of bin size on rule preservation.

Partition sizes: We use a fixed partition size of 100 rows per partition in all datasets due to the limits in the size of our hash function's input parameters. Larger partitions would allow for fewer query exchanges between the target and the source at the expense of a higher probability that each partition would contain an update and need to be exchanged over the network.

Percentage of changes: Partitions in the source table were randomly selected to change for each experiment. We experimented with 10% and 50% changes to partitions in all datasets. Changes were generated randomly between the minimum and maximum values of each attribute. The number of partitions and the number of rows changed for 10% and 50% database changes are summarized in Table 1.

TABLE 1
PERCENTAGE OF DATABASE CHANGES AT PARTITION SIZE 100

Datasets	No. of Dataset Rows	No. of Partitions Changed at		No. of Rows Changed at	
		10%	50%	10%	50%
Adult	32561	33	163	3256	16280
Custom Checking	5814	6	29	581	2907
Pima Indians	768	1	4	77	384
Sales Fact	11040	11	55	1104	5520

We measure the effectiveness of our proposed approach on three levels: pattern preservation, privacy preservation and execution time. The details on each follow.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2484318, IEEE Transactions on Services Computing

5.2 Preserving Patterns

To measure the quality of rules obtained from our approach, we strove to find how closely they matched the rules generated from incrementally updated original data. Recall measures the extent of patterns preserved and precision measures the extent of non-extraneous patterns generated. High recall and precision indicate good pattern preservation. Typically, the strength of association rules is determined at varying combinations of minimum support and minimum confidence thresholds. We, therefore, provide results for rule accuracy at low and high thresholds.

Low thresholds: We calculated recall and precision on each dataset for three bin sizes with 10% and 50% changes to partitions at sup_{min} = 10% and $conf_{min}$ = 50%. Table 2(a) shows recall and precision for a bin size of 2. Table 2(b) shows recall and precision for a bin size of 5 and Table 2(c) shows recall and precision for a bin size of 8.

TABLE 2
RULE ACCURACY AND COMPARISON AT LOW THRESHOLDS
SUPPORT: 0.1, CONFIDENCE: 0.5

Application Domain	10% Change				50% Change			
	Recall (%)		Precision (%)		Recall (%)		Precision (%)	
	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree
Adult	100	100	100	100	100	98.2	100	98.2
Custom Checking	99.9	95.5	99.9	95.5	99.5	92.2	99.4	91.7
Pima Indians	98.5	94.1	98.5	91.1	94.2	75.9	97.8	60.2
Sales Fact	100	100	100	100	100	100	100	100
Average	99.6	97.4	99.6	96.7	98.4	91.6	99.3	87.5

(a) We used 2-Bin Discretization in the above table.

Application Domain	10% Change				50% Change			
	Recall (%)		Precision (%)		Recall (%)		Precision (%)	
	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree
Adult	83.3	38.8	88.2	38..8	92.85	85.7	68.4	100
Custom Checking	99.6	68.6	99.6	74.3	98.9	64.2	99.2	66.3
Pima Indians	95.5	58.2	99.1	84.1	85.8	47.1	89.3	80.2
Sales Fact	92.5	97.5	100	92.8	100	100	100	100
Average	92.7	65.8	96.7	83.7	94.4	74.3	89.2	86.6

(b) We used 5-Bin Discretization in the above table.

Application Domain	10% Change				50% Change			
	Recall (%)		Precision (%)		Recall (%)		Precision (%)	
	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree
Adult	100	78.3	100	94.8	100	37.5	100	100
Custom Checking	98.9	60.4	98.9	54.7	100	62.3	99.3	69.4
Pima Indians	86.7	32.5	94.7	64.3	76.5	52.9	92.8	75
Sales Fact	86.7	86.6	100	100	100	100	100	100
Average	93.1	64.5	98.4	78.5	94.1	63.2	98.0	86.1

(c) We used 8-Bin Discretization in the above table.

High thresholds: Recall and precision are also calculated on each dataset for three bin sizes with 10% and 50% changes to partitions at sup_{min} = 30% and $conf_{min}$ = 70%. Table 3(a) shows recall and precision for a bin size of 2. Table 3(b) shows recall and precision for a bin size of 5 and Table 3(c) shows recall and precision for a bin size of 8.

TABLE 3
RULE ACCURACY AND COMPARISON AT HIGH THRESHOLDS
SUPPORT: 0.3, CONFIDENCE: 0.7

Application Domain	10% Change				50% Change			
	Recall (%)		Precision (%)		Recall (%)		Precision (%)	
	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree
Adult	100	100	100	100	100	100	100	100
Custom Checking	100	97.9	100	92.9	100	99.4	100	86.1
Pima Indians	96.5	74.9	96.1	86.7	95.2	78.8	97.2	70.9
Sales Fact	100	88.5	100	100	100	100	100	100
Average	99.1	90.3	99.0	94.9	98.8	94.6	99.3	89.3

(a) We used 2-Bin Discretization in the above table.

Application Domain	10% Change				50% Change			
	Recall (%)		Precision (%)		Recall (%)		Precision (%)	
	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree
Adult	100	33.3	66.66	33.3	0	0	0	0
Custom Checking	99.1	90.4	98.1	94	96.6	93.1	96.6	96.4
Pima Indians	89.5	42.1	100	100	0	0	0	0
Sales Fact	100	100	100	100	100	100	100	100
Average	97.2	66.5	91.2	81.8	98.3	96.6	98.3	98.2

(b) We used 5-Bin Discretization in the above table.

Application Domain	10% Change				50% Change			
	Recall (%)		Precision (%)		Recall (%)		Precision (%)	
	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree	TB-PIRU	Kd-tree
Adult	0	0	0	0	0	0	0	0
Custom Checking	100	95.8	96.1	88.5	100	80.4	100	100
Pima Indians	100	0	100	0	0	0	0	0
Sales Fact	100	100	100	100	0	0	0	0
Average	100.0	97.9	98.7	94.3	100.0	80.4	100.0	100.0

(c) We used 8-Bin Discretization in the above table.

The figures indicate the strength of the proposed TB-PIRU approach in incrementally preserving rules at varying thresholds. We find that in general the average recall and precision values are both very high (above 90%) signifying excellent pattern preservation regardless of the bin sizes, the percentage of partitions changed and minimum support and confidence thresholds. This is because both the incrementally updated original data, as well as the incrementally updated transformed data, on which rules are mined, are discretized before mining and because changes are generated within the domain of original values. Both factors contribute in assigning nearly the same interval to nearly the same number of data points, thus keeping correlations between coefficients intact.

TB-PIRU also shows higher recall and precision values in comparison with the kd-tree approach indicating that it preserves the underlying data patterns to a greater degree of accuracy than kd-tree. This is due to the sort operation performed on each attribute before coefficient calculation. The Haar wavelet transform (component of TB-PIRU) is a bottom-up approach that computes approximations on two closest numeric values of each attribute. Note that this is different from the kd-tree approach where similar values are grouped together only for the splitting attribute. Kd-tree works in a top-down greedy manner. It preserves associations only if all attribute values are correlated. If some attributes are not strongly correlated, their values fall under different groups thereby destroying rules.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2484318, IEEE Transactions on Services Computing

In general, the accuracy of rules drops with an increase in the bin size. This is especially true for the kd-tree. While low recall is unacceptable, low precision can be mitigated by filtering extraneous rules. Lower averages for both TB-PIRU and kd-tree in Tables 3(b) and 3(c) are attributed to the fact that higher bin size and higher minimum support and minimum confidence thresholds generate no rules for some datasets. This is shown by empty set symbols in Tables 3(b) and 3(c).

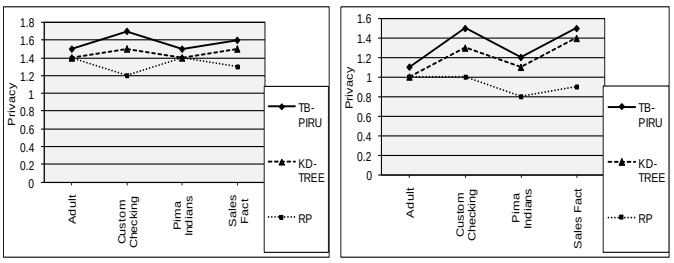
5.3 Preserving Privacy

There are two main instances in our proposed approach when privacy breaches can occur. First, when the data owner passes transformed deltas to the data miner. Second, when the data miner plugs the received transformed delta in his previous transformation to estimate updated original values. In the first case, we must ensure that the transformed delta does not reveal any element of the raw delta on which it is based. We refer to the high level of privacy obtained in such case by authors of [4]. The authors use information theory approach to prove that wavelet transforms do indeed provide higher levels of conditional privacy than random projection and kd-tree. In the second case, care must be taken to prevent the adversary from inferring elements of incrementally updated original (synchronized) data from elements of incrementally updated transformed data. One way our algorithm takes care of this is by eliminating or rotating tuples in a partition with odd number of rows. Additionally, we use a widely accepted measure of privacy that takes as input both incrementally updated raw and transformed data to test the privacy of the incrementally updated raw data. It is based on confidence intervals [35]. It states that if an incrementally updated original attribute x can be estimated with $c\%$ confidence to lie in the interval $[x_1, x_2]$, then privacy is measured by the difference between the incrementally updated transformed data and the incrementally updated original data, normalized by the range of the incrementally updated original data, as follows:

$$\frac{x_2 - x_1}{\max\{x\} - \min\{x\}},$$

where $\max\{x\}$ is the maximum value of attribute x and $\min\{x\}$, its minimum value. We use the confidence interval measure to compare the privacy achieved by our approach with privacies offered by kd-tree and random projection approaches. We use 95% confidence interval in our experiments. Fig. 8(a) shows the levels of privacy attained for 10% changes to source database partitions. Fig. 8(b) shows the levels of privacy attained for 50% changes to source database partitions. Privacy of incrementally updated raw data is preserved due to pruning of detail coefficients (true raw data cannot be reconstructed from approximation coefficients alone).

The plots indicate that irrespective of the percentage of changes, our approach offers higher privacy than kd-tree and random projection. In figures 8(a) and 8(b), the confidence interval for all datasets transformed with TB-PIRU is above 1. A confidence interval of 1.1 for adult dataset in Fig. 8(a) means that 95% of the times, the incrementally transformed data deviates from the original adult data in the range -55% to +55%. A confidence interval of 1.5 for custom checking dataset in Fig. 8(a) means that 95% of the times, the incrementally transformed data deviate from the original custom checking data in the range -75% to +75% and so forth. Intuitively, privacy should be higher when more data is changed. We see that in Fig. 8(b).



(a) 10% Changes (b) 50% Changes

Fig. 8. Privacy comparison with 10% and 50% database changes

5.4 Overhead of the Proposed Method

There are three main overheads that interest us in the present scenario. The time it takes to detect and retrieve changes in the source, the time it takes to transform the changes and the time it takes to update association rules. We measured these times for our approach with 10% and 50% changes to partitions in the source database. We compared the total time to the time it takes a brute-force approach to complete these operations. Fig. 9 shows this comparison at low data mining thresholds: $sup_{min} = 10\%$ and $conf_{min} = 50\%$.

In the absence of our approach, one must resort to full replication with table-level locking, full transformation of the fully replicated data with IP-DWT and re-executing Apriori algorithm on the whole transformed data. There is no information on the last synchronization (partition start and end markers and hashes) that can be exploited for the current synchronization, transformation and rule generation. So the fastest approach to transform the changed table would be to truncate the whole table and reload it from the source, then apply full transformation on the fully reloaded table and finally run Apriori on entire transformation. This would mean that the entire table is locked during the full replication and full transformation period as well as while the entire transformation is transmitted to the mining engine, thereby disabling queries to the source, target and the mining engine during that period.

Our experiments plotted in Fig. 9 show that our approach consistently outperforms full replication, transformation and rule generation (full RTRG) when the source is subject to 10% changes, strengthening the efficacy of our approach for small changes to the source database. For 50% changes our approach shows time savings for all except one dataset. To analyze the graph in Fig. 9, we break down the execution time into the three main overheads incurred by TB-PIRU and full RTRG. Table 4 shows this breakdown at low data mining thresholds.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2484318, IEEE Transactions on Services Computing

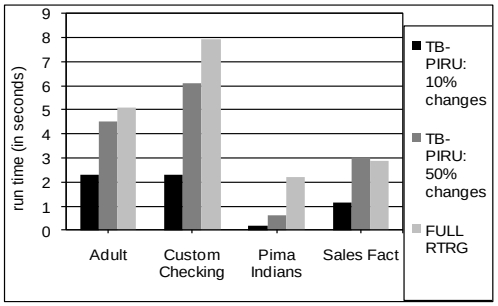


Fig. 9. Execution time with 10% and 50% database changes at low data mining thresholds. The graph shows that the TB-PIRU algorithm is an efficient approach when there are few changes in the source databases.

While changes up to 10% result in increased efficiency for all algorithms that comprise TB-PIRU, 50% changes do not work well for some datasets and applications. We find that the time to detect 50% changes exceeds full replication time for datasets over 10,000 rows (adult and sales fact in Table 4). This is the effect of our algorithm exchanging more data than simply copying the source database to the target. For such datasets, our algorithm should be used for fewer than 50% changes and one should resort to full replication for changes over that threshold. We also find an unusually high cost to incrementally mine 50% changes to the custom checking dataset. This high cost is also incurred for generating rules from scratch. This is due to the high dimensionality of the custom checking dataset. We found that in custom checking, the number of frequent items grew with the dimensionality of the data which caused an increase in the computation and I/O costs because a large number of candidate itemsets were generated.

TABLE 4
BREAKDOWN OF EXECUTION TIME INTO MAIN OVERHEADS
SUPPORT: 0.1, CONFIDENCE: 0.5

	10% Changes			50% Changes			full replication: table-level lock	full transformation: IP-DWT	full rule generation : APRIORI
Datasets	Change Detection	Transformation: IP-DWT	Incremental Mining: FUP_2	Change Detection	Transformation: IP-DWT	Incremental Mining: FUP_2			
Adult	2.03	0.004	0.29	3.2	0.029	1.3	2.4	0.09	2.6
Custom Checking	0.7	0.0035	1.6	1.5	0.012	4.6	2	0.03	5.9
Pima Indians	0.09	0.0004	0.08	0.2	0.002	0.4	1.7	0.002	0.5
Sales Fact	1.03	0.002	0.12	2.6	0.006	0.43	2.07	0.09	0.73

Time is shown in seconds.

Note that data mining parameters at low thresholds and high thresholds affect the execution time. Fewer rules are generated at high thresholds thereby lowering the execution time. A comparison of the time it takes to mine association rules at different thresholds is shown in Table 5.

As evident from this analysis, there are undoubtedly operations in TB-PIRU that cause bottlenecks and delays individually, however implemented together, they prove to be efficient for all datasets when a small percentage of the datasets change and indicate higher efficiency for a majority of the datasets – Adult, Custom Checking and Pima Indians when half of the original datasets change.

TABLE 5
EXECUTION TIME TO INCREMENTALLY MINE RULES AT BOTH LOW AND HIGH DATA MINING THRESHOLDS

	10% Changes		50% Changes		Full Rule Generation: APRIORI	
Datasets	Incremental Mining: FUP_2 s=0.1; c=0.5	Incremental Mining: FUP_2 s=0.3; c=0.7	Incremental Mining: FUP_2 s=0.1; c=0.5	Incremental Mining: FUP_2 s=0.3; c=0.7	s=0.1; c=0.5	s=0.3; c=0.7
Adult	0.29	0.25	1.3	1.2	2.6	1.3
Custom Checking	1.6	1.5	4.6	4.5	5.9	5.06
Pima Indians	0.08	0.07	0.4	0.3	0.5	0.28
Sales Fact	0.12	0.1	0.43	0.42	0.73	0.68

Time is shown in seconds.

6 PRIVACY ANALYSIS

Past work in data mining and data publishing has revealed threats to privacy resulting from an adversary's background knowledge. Since it is hard to know what background knowledge an adversary might possess, scientific research assumes a worst-case scenario when analyzing privacy. One worst-case scenario can be seen in differential privacy [36], [37], in which the assumption is that the attackers know every row in the database except one. The differential privacy model is the strongest and the most preferred model of privacy analysis today.

In addition to background knowledge, multiple sequential releases of anonymized datasets may expose sensitive data. This problem has been studied in data publishing [38], [39]. However, there is no such study in data mining. Chen et al. [37] measure the performance of association rule mining after applying differential privacy, but their method applies to a static dataset.

Since updates take place continuously, sanitized data is published in succession in our study as well. The privacy protection we impose with discrete wavelet transform can also be easily circumvented by inference and by comparing sequential publications of sanitized data. We, therefore, study an inference and a background knowledge attack on the data sanitized with DWT to understand its implications for our research.

Our goal is to analyze privacy breaches with background knowledge and by inference from sequential releases. So we bound the attacker's knowledge to a few pieces of critical information such as knowledge of the encryption scheme and knowledge of the original data.

Armed with this knowledge, we simulate a privacy attack using payroll data. Fig. 10 shows salaries of employees of a hypothetical company (see top left *Original Data D*). We partition this data using partition size 4 rows per partition (top center left *Partition Information of D*). The transformation of the sensitive "salaries" data as per our technique is shown top center right (*D Transformed to T*). This data is shared with an adversary without row ids (top right *Shared T*).

Let us assume that in the first round of update to *D*, row 1 is deleted. Per our method, partition 1 with row IDs greater than and equal to *R1* and less than *R5* has changed (see *First Update U1*). This partition is transformed (see *Transformed U1*) and plugged in the previous transform *T* with the result shown in *T'1*. *T'1* is retained by the owner. Deleted and inserted deltas, $\Delta'^{-}1$ and $\Delta'^{+}1$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2484318, IEEE Transactions on Services Computing

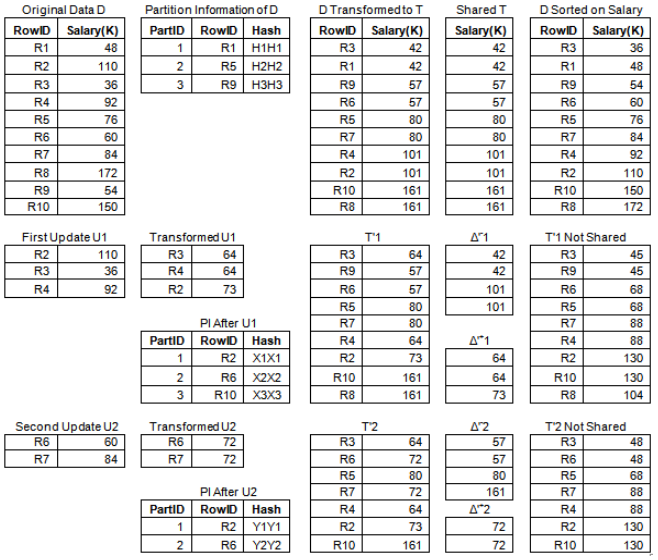


Fig. 10. Privacy Attacks due to background knowledge and multiple transforms. This figure shows how privacy breaches can occur if individuals have background knowledge, sanitized data is published sequentially and there is no approach like TB-PIRU. It serves as a comparison between what gets revealed and what doesn't with and without our approach.

respectively, are sent to the miner. Partition Information after the first update is shown in *PI After U1.*

Next let us assume that in the second round of updates to *D*, rows *R8* and *R9* are deleted. Per our method, partition 2 with all rows greater than and equal to *R6* and less than *R10* has undergone a change (see *Second Update U2*). Note that partition information changes with each round of updates. Partition Information after the second update is shown in *PI After U2.* The transformation of partition 2 (see *Transformed U2*), the update of *T'1* to *T'2* and sharing of Δ'-2 and $\Delta'^{+}2$ take place exactly as indicated for the first round of updates.

The three tables (far right) illustrate how privacy is affected if entire datasets are transformed and the exchange of data between the data owner and the data miner takes place without an approach such as ours that focuses on identifying and transforming the deltas. They serve as a comparison with the data that it is sanitized using our proposed TB-PIRU approach. Note that *T'1 Not Shared* is obtained by applying DWT to *D Sorted on Salary* after removing *R1* and its corresponding salary value. Removing *R1* leaves an odd number of rows, i.e. 9 in *D Sorted on Salary,* therefore the value 172 in the odd row *R8* is rotated by computing an average with the value 36 in the first row *R3*. This results in the value in *R8* becoming 104 in *T'1 Not Shared*.

T'2 Not Shared is obtained by applying DWT on salary values that correspond to *R3* and *R6* of *D Sorted on Salary* and plugging these values in *T'1 Not Shared*. Since this is the second update both *R8* and *R9* and their corresponding salary values are removed from *D Sorted on Salary* as well as *T'1 Not Shared.*

We now inspect two cases of compromise in sensitive information - one with a lower degree of compromise and another with a higher degree of compromise.

- Lower degree of compromise: we assume that the attacker knows the encryption scheme – in this case, the wavelet basis as well as the level of transform. We also assume that the two operations performed on the data - sort and duplicate – have become common knowledge.
- Higher degree of compromise: We assume that in addition to the above, the attacker has managed to obtain complete information about some data elements such as names of individuals and their sensitive data in plaintext.

Lower degree of compromise: Let us assume that the data miner (the adversary in this case) already has a copy of the initial transformation with the row IDs (*D Transformed to T*). In the absence of our approach, he receives a copy of *T'1 Not Shared* after the first update and a copy of *T'2 Not Shared* after the second update.

From *D Transformed to T*, he knows:

$$R3 + R1 = 84 \quad (1)^5$$
$$R9 + R6 = 114 \quad (2)$$

From *T'1 Not Shared*, he knows:

$$R3 + R9 = 90 \quad (3)$$

From *T'2 Not Shared*, he knows:

$$R3 + R6 = 96 \quad (4)$$

This allows him to find the true values:

$$R3 = 36, R1 = 48, R9 = 54, R6 = 60 \quad (5)$$

from four equations and four unknowns, if he has complete knowledge of the encryption scheme.

However, if the data owner shares only the transformed partitions, *Transformed U1* and *Transformed U2* as we propose, then the data miner's sanitized data looks like *T'1* and *T'2* after plugging the transformed partitions *U1* and *U2* in the previous transforms. The four equations now known to the data miner from sequential shared transforms are:

$$R3 + R1 = 84 \quad \text{(see } D \text{ Transformed to } T\text{)} \quad (6)$$
$$R9 + R6 = 114 \quad \text{(see } D \text{ Transformed to } T\text{)} \quad (7)$$
$$R3 + R9 = 121 \quad \text{(see } T'1\text{)} \quad (8)$$
$$R3 + R6 = 136 \quad \text{(see } T'2\text{)} \quad (9)$$

and these prevent him from solving for the true values of *R1, R3, R6* and *R9*. The values he would now calculate are:

$$R3 = 40.5, R1 = 43.5, R9 = 49.5, R6 = 64.5 \quad (10)$$

These discrepancies arise because partitions are created on sorted row IDs and transformations are created on sorted attribute values. To prevent even this disclosure, we propose pruning row identities.

Higher degree of compromise: Let us now assume that in addition to the encryption scheme, the adversary has complete knowledge about two data samples (i.e. sample size is $n > 1$); he knows that Bob and Alice, identified by *R2* and *R8* earn six figure salaries, *$110K* and *$172K* respectively. From the very first transformation, he knows that *R2* and *R8* are *101* and *161* respectively. This then allows him to compute two true contiguous values of *R2*

[5] We add values that correspond to rows R3, R1, R9 and R6.

and *R8*, namely *92* and *150* (extreme right), based on the formula,

$$R_{i+1} = 2R'_{i+1} - R_i \quad (11)$$

Where R denotes the true value
R' denotes the transformed value
i denotes the row id
$i+1$ denotes the contiguous row id after sorting
R_i denotes the true value that corresponds to i
R_{i+1} denotes the true value that corresponds to $i+1$. It is the contiguous true value of R_i
R'_{i+1} denotes the transformed value that corresponds to $i+1$. It is the transform of two true contiguous values R_i and R_{i+1}.

This demonstrates that background knowledge with an initial release of sanitized data poses a disclosure threat when the number of true known samples progressively increases. However, if the rows IDs are not shared, risks and disclosures can be mitigated in that the identity of the individuals receiving these salaries is not revealed.

Background knowledge with sequential releases of sanitized data reduces disclosure risk slightly as evident from *T'1*, where the value of *R2* is *73* instead of *101*. Since the value of *R8* remains the same, only one true contiguous value can be calculated. Therefore, to mitigate privacy breaches due to known plaintext (original data) and cipher text (privatized data) in sequential transforms, we propose updating rules only when the estimated difference between them before and after they are updated is large [35].

The privacy analysis conducted thus far applies to any quantitative data and is not specific to only payroll data. Since quantitative data is the most commonly used data in banking, finance, marketing and other related areas, obscuring quantitative data has numerous benefits. We emphasize that in section 5.3, we apply two different established and proven techniques to measure the level of privacy obtained in quantitative data. These techniques serve to measure and ensure data confidentiality in general. Our analysis of privacy is therefore heuristic in nature. A more comprehensive analysis of privacy that formally quantifies the kind of private information that gets revealed to an adversary for a given set of parameter values can be pursued in a future study.

7 Conclusion

Previous work on privacy preserving quantitative association rule mining using discrete wavelet transform has been set in an environment where the collection of data is not transient and not subject to change. Since data is seldom static and at rest, that approach has little practical significance. We propose a solution to incrementally mine quantitative association rules securely in a dynamic environment where the detection of change in the original data is initiated and controlled by a database other than the one in which changes originate. Tests show 90% -100% accuracy of the rules for most datasets even when 50% of the original data undergoes a change. Comparisons indicate that the proposed TB-PIRU algorithm outperforms other approaches in preserving both data privacy and rules and is very efficient for 10% of changes in the original data, when this data is large in size. We provide a heuristic analysis of privacy for numeric data. A future study is required to conduct a formal analysis of privacy.

References

[1] W. K. Wong, D. W. Cheung, E. Hung, and H. Liu, "Protecting privacy in incremental maintenance for distributed association rule mining," PAKDD'08: Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining, 2008.

[2] M. Ahluwalia, A. Gangopadhyay, and Z. Chen, "Preserving Privacy in Mining Quantitative Association Rules," International Journal of Information Security and Privacy, 2010.

[3] M. Ahluwalia, R. Gupta, A. Gangopadhyay, Y. Yesha, and M. McAllister, "Target-Based Database Synchronization," presented at the 25th ACM Symposium on Applied Computing, Sierre, Switzerland, 2010

[4] M. Ahluwalia, A. Gangopadhyay, Z. Chen, and Y. Yesha, "Target-Based Privacy Preserving Association Rule Mining," presented at 26th ACM Symposium on Applied Computing, Tai-Chung, Taiwan, 2011.

[5] D. W. Cheung, S. D. Lee, and B. Kao, "A general incremental technique for maintaining discovered association rules," Proc. 5th Int. Conf. Database Systems Advanced Applications, pp. 1-4, 1997.

[6] A. Evfimevski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02), pp. 217 - 228, 2002.

[7] J.-L. Lin and J. Y.-C. Liu, "Privacy preserving itemset mining through fake transactions," in Proceedings of the 2007 ACM symposium on Applied computing. Seoul, Korea: ACM Press, 2007, pp. 375-379.

[8] S. Rizvi and J. R. Haritsa, "Maintaining Data Privacy in Association Rule Mining," VLDB, pp. 682-693, 2002.

[9] Z.-Y. Chen and G.-H. Liu, "Quantitative Association Rules Mining Methods with Privacy-preserving," Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2005. PDCAT 2005., pp. 910-912, 2005.

[10] J. S. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 639 - 644, 2002.

[11] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," IEEE Transactions on Knowledge and Data Engineering, vol. 16, pp. 1026-1037, 2004.

[12] C. C. Aggarwal and P. S. Yu, "A Condensation Approach to Privacy Preserving Data Mining," 9th International Conference on Extending Database Technology, pp. 183-199, 2004.

[13] S. R. M. Oliveira and O. R. Zaïane, "Protecting Sensitive Knowledge by Data Sanitization," In proc. of the 3rd IEEE International Conference on Data Mining, pp. 613-616, 2003.

[14] M. J. Atallah, A. K. Elmagarmid, M. Ibrahim, and V. S.

Verykios, "Disclosure Limitation of Sensitive Rules," presented at IEEE Knowledge and Data Engineering Workshop, 1999.

[15] S. Menon, Sarkar, S., "Minimizing Information Loss and Preserving Privacy," Management Science, vol. 53, pp. 101-116, 2007.

[16] Y. Saygin, V. S. Verykios, and C. Clifton, "Using Unknowns to Prevent Discovery of Association Rules," ACM SIGMOD Record, vol. 30, pp. 45-54, 2001.

[17] V. S. Verykios, A. K. Elmagarmid, B. Elisa, Y. Saygin, and D. Elena, "Association Rule Hiding," IEEE Transactions on Knowledge and Data Engineering, vol. 16, pp. 434-447, 2004.

[18] E. Dasseni, Verykios, V., Elmagarmid, A. and Bertino, E., "Hiding Association Rules by Using Confidence and Support," Proc. of 4th Intl. Information Hiding Workshop (IHW), 2001.

[19] Y. Saygin, Verykios, V. and Elmagarmid, A., "Privacy Preserving Association Rule Mining," Proc. of 12th Intl. Workshop on Research Issues in Data Engineering (RIDE), 2002.

[20] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," IEEE Transactions on Knowledge and Data Engineering, vol. 16, pp. 1026-1037, September 2004.

[21] J. S. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," presented at 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, July 2002.

[22] B. Kalpana, R. Nadarajan, and J. S. Babu, "A Galois Lattice framework to handle updates in the mining of closed itemsets in dynamic databases," in Proceedings of the 1st Bangalore annual Computer conference. Bangalore, India: ACM, 2008, pp. 1-6.

[23] W. Cheung and O. R. Zaïane, "Incremental Mining of Frequent Patterns without Candidate Generation or Support Constraint," In Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03), pp. 111-116, 2003.

[24] M. E. Otey, S. Parthasarathy, C. Wang, A. Veloso, and W. Meira, Jr., "Parallel and distributed methods for incremental frequent itemset mining," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 34, pp. 2439-2450, 2004.

[25] A. Veloso, W. M. Jr, M. B. d. Carvalho, B. Pôssas, S. Parthasarathy, and M. Zaki, "Mining frequent itemsets in evolving databases," Proc. 2nd SIAM Int. Conf. Data Mining Arlington, TX, pp. 494-510, 2002.

[26] V. Ganti, J. Gehrke, and R. Ramakrishnan, "DEMON: Mining and monitoring evolving data," Proc. 16th Int. Conf. Data Engineering, San Diego, CA, pp. 439-448, 2000.

[27] F. A. Chowdhury, K. T. Syed, J. Byeong-Soo, and L. Young-Koo, "Mining high utility patterns in incremental databases," Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, pp. 656-663, 2009.

[28] S. Lee and D. Cheung, "Maintenance of discovered association rules: When to update?," SIGMOD Workshop Research Issues Data Mining Knowledge Discovery, 1997.

[29] D. Cheung, J. Han, V. Ng, and C. Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating technique," Proc. 12th Int. Conf. Data Engineering, pp. 106-114, 1996.

[30] M. Ahluwalia and A. Gangopadhyay, "Privacy Preserving Data Mining –Taxonomy of Existing Techniques," in *Computer Security, Privacy and Politics: Current Issues, Challenges and Solutions*, R. Subramanian, Ed.: IGI Global, 2008.

[31] S. Hettich, C. L. Blake, and C. J. Merz, "UCI Repository of machine learning databases," 1998.

[32] R. Kimball and M. Ross, The Data warehouse Toolkit: Wiley, 2002.

[33] X. Li and S. Sarkar, "A Tree-Based Data Perturbation Approach for Privacy-Preserving Data Mining," Knowledge and Data Engineering, IEEE Transactions on, vol. 18, pp. 1278-1283, 2006.

[34] L. Kun, H. Kargupta, and J. Ryan, "Random projection-based multiplicative data perturbation for privacy preserving distributed data mining," Knowledge and Data Engineering, IEEE Transactions on, vol. 18, pp. 92-106, 2006.

[35] R. Agrawal and R. Srikant, "Privacy preserving data mining," 2000 ACM SIGMOD Conference on Management of Data, pp. 439-450, 2000.

[36] Y. Yang, Z. Zhang, G. Miklau, M. Winslett, and X. Xiao, "Differential Privacy in Data Publication and Analysis," *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 601-606, 2012.

[37] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, L. Xiong, "Publishing Set-Valued Data via Differential Privacy," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 1087-1098, August 29th - September 3rd, 2011.

[38] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li, "Secure anonymization for incremental datasets," *In SDM*, pp. 48-63, 2006.

[39] X. Xiao and Y. Tao, "m-Invariance: Towards Privacy Preserving Re-publication of Dynamic Datasets," *Proceedings of ACM International Conference on Management of Data (SIGMOD)*, pp. 689-700, 2007.

Madhu V. Ahluwalia obtained her B. SC in Information Systems in 2004 and her Ph.D. in the same program in 2011. In 2009, she completed a research fellowship with SAP Business Objects. In 2011, Madhu joined Harris Corporation and works currently with that organization as a database administrator. Madhu has published parts of her research on privacy preserving data mining in several Refereed Conferences, in a journal, and in a book chapter. She holds a patent in the area of change data capture. She is interested in privacy protecting techniques applicable in association rule mining and the practical significance of this area in health informatics. Her current focus is on supporting the implementation of an Awards based functionality within the Veterans Benefits Management System (VBMS) enterprise framework. The awards are granted in part based on the health ratings of a veteran. Madhu's other research interests include data warehousing and massively parallel processing (MPP) databases. She has attended workshops organized by EMC Corporation on big data analytics and on technologies that integrate MPP databases with Hadoop Distributed File System.

Aryya Gangopadhyay is a professor and the chair of Information Systems Department at the University of Maryland Baltimore County (UMBC). He holds a PhD in Computer Information Systems from Rutgers University. Dr. Gangopadhyay's research interests are in the areas of databases and data mining. Currently, he is focused on privacy preserving data mining, spatio-temporal data mining, and data mining for health informatics. His research has been funded by grants from NSF, NIST, US Department of Education, Maryland Department of Transportation, and other agencies. Dr. Gangopadhyay has published five books and nearly 100 research articles.

Zhiyuan Chen has obtained a BS and a MS from Fudan University, China, and a PhD in Computer Science from Cornell University. He is currently an associate professor at Department of Information Systems, University of Maryland Baltimore County. He serves on the editorial board of International Journal of Knowledge-Based Organizations and International Journal of Embedded Software and Open Source Systems. He has published over forty peer reviewed journal articles and conference papers. His research interests include priva-

cy preserving data mining, data navigation and visualization, health IT, big data, XML, automatic database tuning, and database compression.

Yelena Yesha is a tenured Professor at the Department of Computer Science and Electrical Engineering at the University of Maryland, Baltimore County. She received her B.Sc. degrees in Computer Science and in Applied Mathematics from York University, Toronto, Canada in1984, and her M.Sc. degree and Ph.D. degrees from the Ohio State University in1986 and 1989 respectively. She joined the Department of Computer Science at the University of Maryland, Baltimore County as an Assistant Professor in 1989. In 1994, she was promoted to the rank of tenured Associate Professor, and in 1995, she was promoted to the rank of tenured Professor. In 1994-1995, while on leave from the University of Maryland, Baltimore County, she was the Director of the Center for Applied Information Technology (CAIT) at the National Institute of Standards and Technology as a United States federal government employee (civil servant). In1995, she became the Director of the Center of Excellence in Space Data and Information Sciences (CESDIS) at the NASA Goddard Space Flight Center. She has published 12 books as author or editor, and more than 200 papers in prestigious refereed journals and refereed conference proceedings, and has been awarded external funding in a total amount exceeding 30 million dollars.