A Scalable and Efficient Privacy Preserving Global Itemset Support Approximation Using Bloom Filters

Vikas G. Ashok, Ravi Mukkamala

A Scalable and Efficient Privacy Preserving Global Itemset Support Approximation using Bloom Filters

V. G. Ashok[1] and R. Mukkamala[2]

[1] State University of New York, Stony Brook, NY 11794, USA,
vganjiguntea@cs.stonybrook.edu
[2] Old Dominion University, Norfolk, VA 23529, USA,
mukka@cs.odu.edu

Abstract. Several secure distributed data mining methods have been proposed in the literature that are based on privacy preserving set operation mechanisms. However, they are limited in the scalability of both the size and the number of data owners (sources). Most of these techniques are primarily designed to work with two data owners and extensions to handle multiple owners are either expensive or infeasible. In addition, for large datasets, they incur substantial communication/computation overhead due to the use of cryptographic techniques. In this paper, we propose a scalable privacy-preserving protocol that approximates global itemset support, without employing any cryptographic mechanism. We also present some emperical results to demonstrate the effectiveness of our approach.

Keywords: Privacy Preserving Set Union Protocol, Privacy Preserving Data Mining, Secure Multiparty Computation

1 Introduction

With the increasing need for global data mining, mining of data dispersed among a wide variety of data owners, several schemes have been developed in literature. In addition, the ever increasing awareness for preserving privacy of individuals and organizations has prompted the development of several privacy-preserving global data mining schemes. In this paper, we focus our attention on secure global set operation algorithms. Current techniques in this area are primarily based on data perturbation and secure multiparty computation (SMC) [1–3].

In the case of data perturbation based techniques, each data owner deliberately distorts its private data before sharing it with a third party data miner. The onus now is on the data miner to use special techniques to reconstruct the original distribution of the received perturbed data for data mining purposes. In the SMC approach of [2], data owners securely perform global data mining without revealing individual private data. While data perturbation techniques (e.g. [4]) trade utility for privacy and suffer from security problems, SMC

techniques (e.g. [3]) are in general expensive in terms of computation and communication overhead since they may depend on encryption mechanisms.

The goal of our work is to present an efficient scalable and privacy preserving protocol for secure set union computation that does not rely on any cryptographic techniques and requires minimum cooperation between the participating data sites. In our approach, we strategically decompose a single bloom filter representing an itemset at each data source into several *partial* bloom filters that can be shared with other sites without violating privacy. Our cooperative protocol ensures that the final result obtained from operations on these *partial* bloom filters is the same as that obtained from a naive protocol employing the original undecomposed bloom filters without any privacy concerns. We also show that our protocol performs better than some of the well known approaches with respect to the overall communication cost. We present a brief discussion of the related work next.

2 Background

A standard Bloom filter [5] uses an m bit array to represent a set S where all m bits are initially set to 0. Every element $x \in S$ is hashed using k independent uniform hash functions $h_1(), \ldots, h_k()$ each with the range $\{1, \ldots, m\}$ and the corresponding bits $h_i(x)$ in the array are set to 1. Any membership query can therefore be addressed by simply hashing the query element y using the same k hash functions and checking if the corresponding bits in the Bloom filter array are set to 1.

It has been shown in [9] that the number z of 0 bits in a Bloom filter for a set S is strongly concentrated around its expectation $m(1-1/m)^{k|S|}$. Therefore, given z, m and k, we can approximate the size of S using (1).

$$|S| = \frac{\ln(z/m)}{k \ln(1 - 1/m)} \tag{1}$$

The secure computation performed by previous work (e.g. [7, 8]) typically refer to the cooperative determination of set intersection/union cardinality for itemsets distributed across several sites. To the best of our knowledge, all existing approaches (e.g. [7,8]) rely on some encryption mechanism for their effective operation, thereby making them either unscalable or cost intensive. Our approach overcomes these deficiencies by employing Bloom filters.

3 Problem Statement

We assume that there are $n \geq 2$ data sites which independently collect similar data for a sample of individuals belonging to some population $\mathcal{P}$. Each site (or data owner) $S_i, 1 \leq i \leq n$, has data for a sample $\mathcal{D}_i$ drawn from population $\mathcal{P}$. Each transaction in $\mathcal{D}_i$ is assumed to have a unique identifier *TID* (e.g., SSN, driver's license number, etc.). Overlaps between samples at different sites

may exist; the data corresponding to the same individual may exist at multiple sites. The goal is to determine the overall global support of an itemset, without violating privacy of individual private data. In other words, for a given itemset X and a site S_i, if $L_i(X)$ denotes the set of *TID*s of transactions in $\mathcal{D}_i$ that contain X, the goal is to preserve individual privacy while estimating the global support ($s_g(X)$) of X given by $|\bigcup_{1\leq i\leq n} L_i(X)|$.

With respect to privacy, we assume an *Honest-but-Curious* (HBC) adversary model [10]. Informally, in the HBC model, the participating parties correctly observe the protocol but they may utilize any information exchanged in the intermediate stages of the protocol to learn certain private details about individuals. Therefore, the privacy requirement in this model states that any party should not be able to learn anything more than what can be deduced from the final output of the multiparty computation.

4 Privacy Preserving Global Itemset Support Computation Protocol

This protocol involves direct communication among the sites (data owners) without any third party. The main idea behind the protocol is based on the following observation: determining global support $s_g(X) = |\bigcup_{1\leq i\leq n} L_i(X)|$ for an itemset X is approximately equivalent to determining the number of elements hashed into the global Bloom filter $BF_G(X)$ using (1), where the set of hashed elements in BF_G equals $\bigcup_{1\leq i\leq n} L_i(X)$. Therefore, the task boils down to computing $BF_G(X)$ in a privacy preserving manner. The protocol is shown in Algorithm 1. It involves two phases, decomposition and reconstruction, described below.

4.1 Decomposition

In the decomposition phase, each site S_i generates n Bloom filters BF_j^i, $1 \leq j \leq n$, one for each of the n sites, which are individually incomprehensible/incomplete but collectively represent $L_i(X)$. Therefore, each of these component Bloom filters can be *safely* shared with another participating site without any concern of privacy violation. To generate these BF_j^i, $1 \leq j \leq n$, we propose a *partial Bloom filter construction* technique that leverages the services of *GenerateLocalKeys* routine presented in Algorithm 1.

In this technique, given an itemset X, a site S_i uses only subsets of publicly known k hash functions to generate any Bloom filters for $L_i(X)$. Both the size and composition of these subset of hash functions are private to S_i. In our protocol, since we generate n component Bloom filters, we need n such subsets of hash functions. In addition, each hash function has to appear in at least one of these subsets so that the Bloom filters collectively represent $L_i(X)$.

The *GenerateLocalKeys* function (Algorithm 1) produces n random subsets (Keys) of **indices** of k Bloom filter hash functions such that each of the k indices appear in at least one subset or Key. As seen in Algorithm 1, in the initialization phase (lines 22–26 in Algorithm 1), n keys are initialized with random subsets

Algorithm 1 Protocol for global support approximation of itemset X

Input: $n \geq 2$, $L_i(X)$ for $1 \leq i \leq n$, m, X, and $H = h_1, \ldots, h_k$ such that $k > n$.
Output: Global support $s_g(X)$ and global Bloom filter $BF_G(X)$.

1: **for all** Sites S_i, $1 \leq i \leq n$ **do**
▷ Phase 1: Decomposition
2: $Keys \leftarrow GenerateLocalKeys(k, n)$
3: **for** $j = 1$ *to* n **do**
4: $BF_j^i \leftarrow CreateBloomFilter(Keys[j], L_i(X))$
5: Send BF_j^i to site S_j
6: **end for**
▷ Phase 2: Reconstruction
7: initialize temporary variable BF_i' to 0.
8: **for** $j = 1$ *to* n **do**
9: Receive BF_i^j from site S_j.
10: $BF_i' \leftarrow BF_i' \vee BF_i^j$
11: **end for**
12: Send BF_i' to all sites.
▷ Merge intermediate results
13: Initialize $BF_G(X) \leftarrow BF_i'$.
14: **for** $j = 1$ *to* n **do**
15: Receive BF_j' from site S_j.
16: $BF_G(X) \leftarrow BF_G(X) \vee BF_j'$
17: **end for**
18: $s_g(X) =$ Approximate size of $BF_G(X)$ using equation 1.
19: **end for**

20: **procedure** GenerateLocalKeys(k, n)
Private: random subset length parameters a, b.
21: Create a set $K \leftarrow \{1, 2, \ldots, k\}$
Initialize: Create n random subsets of K, each of random size r.
22: **for** $i = 1$ to n **do**
23: $r = RandomNumber(a, b), 1 \leq a < b < k$
24: $K' = RandomSubset(K, r)$
25: $Keys[i] \leftarrow K'$
26: **end for**
Finalize: Ensure that all hash functions in HS are considered.
27: **for** $hf = 1$ to k **do**
28: $r' = RandomNumber[1, n]$
29: $Keys[r'] = Keys[r'] \cup \{hf\}$
30: **end for**
31: return $Keys$
32: **end procedure**

33: **procedure** CreateBloomFilter$(Key, List)$
34: Create set $H' \leftarrow \{h_j | j \in Key\}$
Create BF by hashing $List$ using H'.
35: Return BF
36: **end procedure**

of K, where K is the set of all Bloom filter hash function indices. The size of these initialized keys can be controlled using private parameters a and b. In the finalization phase (lines 27–30 in Algorithm 1), we ensure that each hash function index is present in at least one key. In other words, the k hash function indices are randomly distributed among the n keys just like throwing k balls into n bins assuming that each ball is equally likely to fall into any one of the n bins. The index subsets generated by *GenerateLocalKeys* function are then used by the *CreateBloomFilter* subroutine (Algorithm 1) to produce the *partial* Bloom filters.

At the end of the decomposition phase, partial bloom filter BF_j^i is sent to S_j, for every $1 \leq j \leq n$. This sharing is *safe* since the size and composition of $Keys[j]$ used in generating BF_j^i is unknown to S_j. Note that it is not a wise decision to share more than one *partial* Bloom filter with a single site as it is publicly known that these bloom filters collectively represent $L_i(X)$ and therefore may allow adversary to make stronger assumptions about the content of $L_i(X)$.

4.2 Reconstruction

The reconstruction phase is simple and based on the following insight - the global bloom filter BF_G is simply the bitwise 'or' of all the bloom filters generated by all the sites in the decomposition phase.

By the end of the decomposition phase, every site S_i receives BF_i^j, $1 \leq j \leq n$, at the beginning of the reconstruction phase. Therefore, as a first step, the protocol performs a bitwise 'or' of all received BF_i^j and broadcasts the corresponding intermediate result to all other sites. For example, S_1 performs $BF_1^1 \vee BF_1^2 \ldots \vee BF_1^n$ and sends the result of this operation to all sites. Once S_i receives all intermediate bloom filters from other sites, another round of Bitwise OR operation yields the global bloom filter BF_G. The global support $s_g(X)$ can then be obtained from BF_G using (1).

4.3 Privacy Analysis

The extent of privacy achieved by our protocol depends on the confidence with which a site S_i can predict the number and identities of hash functions used by another site S_j, $j \neq i$ for generating BF_i^j. This is because the knowledge of the number of hash functions can reveal the size of the input set through (1) and the knowledge of identities of used hash functions can reveal private data at any site. It is easy to notice that no meaningful information can be deduced from individual analysis of BF'_j, $1 \leq j \leq n$, received during reconstruction phase as these bloom filters are composed of various *partial* chunks belonging to different sites. Therefore, we limit our focus to BF_i^j, $1 \leq j \leq n$, received at the beginning of reconstruction phase.

For a given bloom filter BF_i^j received by S_i from S_j, let T be a random variable indicating the number of hash functions used by S_j to generate BF_i^j. Also,

Table 1. Commnication cost of various protocols. Notation: s - average size of input set of *TID*s and $|P|$ - domain size of input set *TID*s.

Protocol	Communication Cost(bits)
Commutative encryption (Vaidya [7])	$O(n(2n-2)s * \text{EncryptionSize})$
Homomorphic encryption (Kissener [8])	$O(n^2 s \log_2 \lvert P\rvert * \text{EncryptionSize})$
Proposed protocol	$2n(n-1)m$

let E_x denote the event that x hash function indices were placed in $Keys[i]$ after initialization phase of *GenerateLocalKeys* subroutine and let E'_y be the event that $Keys[i]$ is selected y times during finalization phase of *GenerateLocalKeys* subroutine.

$\Pr\{T = t\}$ represents the probability that t hash functions were used in generating BF_i^j. Ideally, we expect this probability to be low for all possible values of t, or in other words the distribution of $\Pr\{T = t\}$ to be uniformly spread out. $\Pr\{T = t\}$ can be computed using the law of total probability as shown in equation (2).

$$\Pr\{T = t\} = \sum_{x=a}^{b} \sum_{y=0}^{k} \Pr\{E_x \cap E'_y\} \Pr\{T = t | (E_x \cap E'_y)\} \tag{2}$$

Since E and E' are mutually independent events, $\Pr\{E_x \cap E'_y\}$ is simply the product of $\Pr\{E_x\}$ and $\Pr\{E'_y\}$. Informally, $\Pr\{T = t|(E_x \cap E'_y)\}$ represents the probability of obtaining t distinct hash functions as a result of union between two random subsets of hash functions having sizes x and y respectively. Therefore, $\Pr\{T = t|(E_x \cap E'_y)\}$ has a non-zero value shown in (3), only when both $x, y \leq t$ and $x + y \geq t$.

$$\Pr\{T = t|(E_x \cap E'_y)\} = \frac{\binom{k-x}{t-x}\binom{x}{y-(t-x)}}{\binom{k}{y}} \tag{3}$$

The probability of determining the identities of hash functions H used in generating BF_i^j can therefore be computed as $\frac{\Pr\{T=|H|\}}{\binom{k}{|H|}}$. These probabilities are typically small as shown in the experimental section. Next, we evaluate the cost of our protocol.

4.4 Cost Analysis

Table 1 compares the communication cost of our protocol with existing popular cryptography dependent approaches proposed in [7] and [8]. The communication cost in our approach is easy to derive since each site transmits $(n-1)m$ bits of information to other sites **twice** during the execution of the protocol. As seen in Table 1, our approach reduces the communication cost to a considerable extent since, in practice, $m << s * \text{EncryptionSize}$. In addition, the absence of encryption tends to lower the computation cost as well.

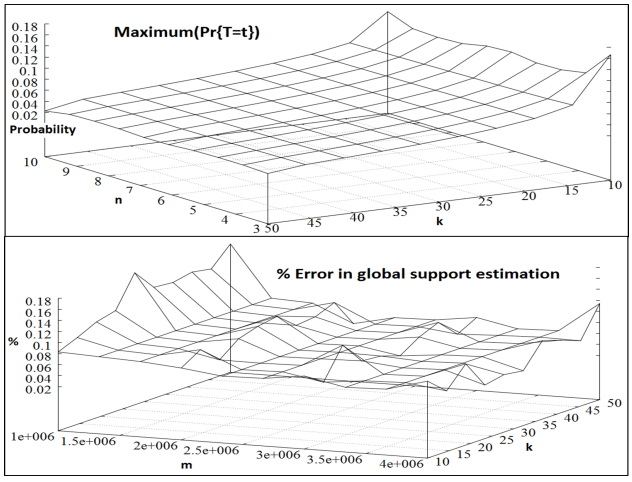

Fig. 1. (top:) Height of density function of T given n and k. (bottom:) Percentage error in global support approximation where actual support is $100,000$.

5 Empirical Results and Analysis

In this section, we study the impact of various protocol parameters on accuracy and privacy of the global frequent itemset mining. As explained earlier, while accuracy is dependent on correct estimation of global union cardinality using equation (1), privacy is dependent on the high confidence prediction of the number (T) and identity of hash functions constituting the keys generated privately using *GenerateLocalKeys*.

Let the height of a probability distribution of T given n and k be defined as the maximum value on the curve representing that distribution. Obviously, smaller the height, better the privacy. Figure 1(top) shows the heights of different probability curves of T obtained by varying the number of sites n and total hash functions k. It is observed that irrespective of n, the height is lowered when k is increased. However, k cannot be arbitrarily increased as it adds to both computation and communication overhead (since $m \propto k$).

We now look at the accuracy measure. In order to assess the impact of m and k on accuracy, we plot average percentage error in global support estimation for an itemset X in Figure 1(bottom) where actual global support $s'_g(X)$ is $100,000$. Specifically, for a given combination of m and k, we plot $\frac{Abs(s'_g(X) - s_g(X))}{s'_g(X)}$. $s_g(X)$ is computed by averaging results over $10,000$ runs. Each of the runs constitutes a simple experiment where we first insert $s'_g(X)$ random elements into a Bloom filter. We compute the number of 0s (z) in the Bloom filter, and then estimate the size of input set from z using (1). As observed in Figure 1(bottom), the global support approximation method is very robust as the maximum percentage error for this scenario is 0.18%. Also, for a few values of m (1.5, 2.5 million bits), the error is very low and independent of k.

In summary, the proposed method has mechanisms to control the accuracy and privacy of the resulting global frequent itemsets obtained through data mining across different data owners.

6 Conclusions and Future Work

In this paper, we looked at an efficient mechanism to approximate global frequent itemset support from data owned by several independent data owners. The primary objectives that inspired the protocol development were local data privacy, scalability, reduced communication and computational cost, and finally high accuracy. While the earlier schemes in literature had high accuracy and privacy, they were deficient in offering scalability and efficiency due to extensive use of cryptographic mechanisms. On the other hand, we employed bloom filters as a means to preserve privacy across data owners. We also carried out empirical analysis of the protocol and determined the relationship between the parameter values of the protocol and the resulting accuracy and privacy. In future, we plan to extend this work by building full-scale prototypes of the protocol to validate the results and gain a deeper understanding of the impact of the hash functions on the performance.

References

1. K. Liu, H. Kargupta and J. Ryan. *Random projection-based multiplicative data perturbation for privacy preserving distributed data mining.* IEEE Trans. Knowledge and Data Engg, 18(1):92-106, January 2006.
2. Y. Lindell and B. Pinkas. *Privacy preserving data mining.* In Advances in Cryptology - CRYPTO 2000, pages 36-54. Springer-Verlag, Aug. 20-24 2000.
3. M. Kantarcioglu, R. Nix, and J. Vaidya. *An efficient approximate protocol for privacy-preserving association rule mining.* In Advances in Knowledge Discovery and Data Mining, pp. 515-524. Springer Berlin Heidelberg, 2009.
4. H. Kargupta, S. Datta, Q. Wang, K. Sivakumar. *On the privacy preserving properties of random data perturbation techniques.* Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003), November 19-22, IEEE Computer Society, Los Alamitos (2003).
5. B. H. Bloom. *Space/time Trade-offs in Hash coding with Allowable Errors*, Communications of the ACM, Vol. 13, No. 7 (1970: 422-426.
6. L. Qiu, Y. Li, and X. Wu. *Preserving privacy in association rule mining with Bloom filters*, Journal of Intelligent Information Systems 29, no. 3 (2007): 253-278.
7. J. Vaidya, and C. Clifton. *Secure set intersection cardinality with application to association rule mining*, Journal of Computer Security 13, no. 4 (2005): 593-622.
8. L. Kissner, and D. Song. *Privacy-preserving set operations*, Advances in Cryptology–CRYPTO 2005, pp. 241-257. Springer Berlin Heidelberg, 2005.
9. A. Andrei, and M. Mitzenmacher. *Network applications of Bloom filters: A survey*, Internet Mathematics 1, no. 4 (2004): 485-509.
10. O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, Vol. 2. Cambridge university press, 2009.