# An Efficient Approximate Protocol for Privacy-Preserving Association Rule Mining

Murat Kantarcioglu[1], Robert Nix[1], and Jaideep Vaidya[2]

[1] University of Texas at Dallas, Richardson, TX 75080, USA
{muratk,robertn}@utdallas.edu
[2] Rutgers University, Newark, NJ, 07102, USA
jsvaidya@rbs.rutgers.edu

**Abstract.** The secure scalar product (or dot product) is one of the most used sub-protocols in privacy-preserving data mining. Indeed, the dot product is probably the most common sub-protocol used. As such, a lot of attention has been focused on coming up with secure protocols for computing it. However, an inherent problem with these protocols is the extremely high computation cost – especially when the dot product needs to be carried out over large vectors. This is quite common in vertically partitioned data, and is a real problem. In this paper, we present ways to efficiently compute the approximate dot product. We implement the dot product protocol and demonstrate the quality of the approximation. Our dot product protocol can be used to securely and efficiently compute association rules from data vertically partitioned between two parties.

## 1 Introduction

Privacy-preserving data mining has made major advances in the recent years. Many protocols have been proposed for different data mining algorithms such as classification, association rule mining, clustering, and outlier detection, etc. [1] provides a comprehensive survey. There are two main types of technique – perturbation based methods and secure multiparty computation techniques. In the perturbation methods the data is locally perturbed before delivering it to the data miner. Special techniques are used to reconstruct the original distribution (not the actual data values), and the mining algorithm needs to be modified to take this into consideration. The seminal paper by Agrawal and Srikant [2] introduced this approach in the form of a procedure to build a decision tree classifier from perturbed data. The second approach assumes that data is distributed between two or more sites that cooperate to learn the global data mining results without revealing the data at individual sites. This approach was introduced by Lindell and Pinkas [3], with a method that enabled two parties to build a decision tree without either party learning anything about the other party's data, except what might be revealed through the final decision tree. Typically, the techniques used are cryptographic. While the first approach has known security problems[4,5] and cannot lead to provably secure solutions, the second approach has typically been too computationally intensive. This is especially the case for

T. Theeramunkong et al. (Eds.): PAKDD 2009, LNAI 5476, pp. 515–524, 2009.

vertically partitioned data (e.g., [6]), where unlike horizontally partitioned data (e.g.,[7]) little data summarization can be carried out before engaging in the distributed protocol.

So the essential question is – how can we mine data in an *efficient* and *provably secure* way? Since most data mining deals with aggregates, one way of solving this conundrum is through approximation. It may be possible to create highly efficient provably secure protocols that approximate the answer. In this paper, we present one of the first such approaches. Instead of directly approach one particular data mining technique, we present an approximate protocol for computing the scalar (dot) product of two vectors owned by two different parties. This is very important, since the dot product is actually used as a sub-protocol in many data mining tasks such as classification, association rule mining, etc. As a sample application, we show how a completely secure and efficient association rule mining protocol could easily be created using our protocol. We implement our protocol and demonstrate the significant efficiency and quality of the approximation, and discuss the implications for association rule mining.

### 1.1 Related Work

While there has been a lot of related work in privacy-preserving data mining, due to space constraints, we only focus on the tightly related efforts. Several protocols have been proposed for computing the scalar product[8,6,9,10]. Out of these [6] is an algebraic protocol that requires quadratic complexity though the constituent operations are quite simple. The other protocols all require significant cryptographic operations and are significantly (orders of magnitude) less efficient. [11] present a simple sampling based protocol for approximately computing the scalar product that is similar in notion to ours. However, since our protocol is based on Bloom filters, it is more sophisticated, has better bounds on accuracy, and is highly efficient. [12] proposes a technique using bloom filters to do association rule mining. However, here transactions are represented via bloom filters and the association rule mining is done centrally, where as enable efficient distributed association rule mining. There is also work on privacy-preserving association rule mining for vertically partitioned data[6,13]. However, by comparison, this is the first secure, highly efficient protocol proposed for this problem.

Securely computing the dot product of two vectors is a very important subprotocol required in many privacy-preserving data mining tasks. Many secure dot product protocols have been proposed in the past. [6,8,9,10,11]. For comparison purposes, we use the method of Goethals et al., described in [10], as an exact scalar product protocol, since it is quite simple and provably secure. Its runtime is $O(n)$ in public-key encryptions.

### 1.2 Bloom Filters

We now give some background on Bloom Filters. Bloom filters have been extensively used for various application domains ranging from networking to databases [14]. Basically, a bloom filter represents a set $S = \{x_1, x_2, \ldots, x_n\}$ of n elements

using array of $m$ bits ($m \leq n$), initially all set to 0. For each element $x \in S$, we use $k$ independent random hash functions $h_1(), h_2(), \ldots, h_k()$ with range $\{1, \ldots, m\}$ such that the bits $h_i(x)$ of the array are set to 1 for $1 \leq i \leq k$. In this basic version, a location can be set to 1 multiple times but only the first change has an effect. To check whether an item $t \in S$, we need to check whether all $h_i(t)$ for $1 \leq i \leq k$ are set to 1. If they are not all set to 1, we can conclude that $t \notin S$. On the other hand, if all $h_i(t)$ for $1 \leq i \leq k$ set to 1, we can assume that $t \in S$, with some nonzero probability.

Bloom filters can be used to approximate the intersection size between two sets. Given two bloom filters with the same $m$ and $k$ values that represent two sets $S_1$ and $S_2$, we can approximate $|S_1 \cap S_2|$ by getting the dot product of the two filters. More precisely, let $Z_1$ (resp. $Z_2$) be the number of 0s in the filter $S_1$ (resp. $S_2$) and $Z_{12}$ be the number of 0s in the inner product, then we can approximate $|S_1 \cap S_2|$ using the following formula [14]:

$$\frac{1}{m}\left(1 - \frac{1}{m}\right)^{-k|S_1 \cap S_2|} \approx \frac{Z_1 + Z_2 - Z_{12}}{Z_1 Z_2} \tag{1}$$

$$\implies |S_1 \cap S_2| \approx \frac{\ln\left(m(Z_1 + Z_2 - Z_{12})\right) - \ln(Z_1) - \ln(Z_2)}{-k \ln\left(1 - \frac{1}{m}\right)} \tag{2}$$

## 2 Algorithm

Instead of looking at the computation of the scalar product, we look at a slight variant of this problem – checking if the scalar product of two distributed vectors is greater than some threshold. In other words, given two vectors $X_1$ and $X_2$, we need to compute whether $X_1 \cdot X_2 = \sum_{i=1}^{n} X_1[i].X_2[i] \geq t$ for a given threshold

**Algorithm 1.** Approximate threshold dot product algorithm

**Require:** $X_1$ belongs to $P_1$ and $X_2$ belongs to $P_2$ each of size $n$. Publicly known common parameters: $m << n$ (size of the bloom filters), hash functions $h_1, \ldots, h_k$, constant integer $c$ such that $c\frac{1}{m}(1 - \frac{1}{m})^{-kt}$ is an integer, prime $p$ such that $n^2 c < p$, threshold value $t$.

1: **for** each $P_i$ **do**
2: Create $S_i = \{j \mid X_i[j] = 1\}$
3: Create a bloom filter $B_i$ of size $m$ using $S_i$ and the common $k$ hash functions.
4: **end for**
5: Each party participates in a secure dot product protocol (e.g., [10]) to get random share $u_i$ where $u_1 + u_2 = B_1 \cdot B_2 \bmod p$ {Note that $Z_{12} = m - B_1 \cdot B_2$}
6: Each party participates in a secure multiplication protocol (e.g., [3]) to get random share $v_i$ where $v_1 + v_2 = (c\frac{1}{m}(1 - \frac{1}{m})^{-kt})Z_1 Z_2 \bmod p$
7: $P_1$ sets $y_1 \leftarrow c.(Z_1 + u_1 - m) \bmod p$
8: $P_2$ sets $y_2 \leftarrow c.(Z_2 + u_2) \bmod p$
9: Each party participates in a secure comparison protocol (e.g, [3]) to learn whether $((y_1 + y_2) \bmod p) \geq ((v_1 + v_2) \bmod p)$

value $t$. This variation is more useful in many applications, such as computing association rules (as will be seen later).

To do this, we use an approximation based on bloom filters. Note that for a given binary vector $X_i$, we can create a set $S_i$ where $j$ is an element of $S_i$ if the $j^{th}$ bit of $X_i$ is set to one (i.e., $S_i = \{j \mid X_i[j] = 1\}$). It is easy to see that $X_1 \cdot X_2 = |S_1 \cap S_2|$. Using this fact, we can efficiently approximate $X_1 \cdot X_2$, as follows: First, for each $X_i$, we can create the corresponding set $S_i = \{j \mid X_i[j] = 1\}$. Given $S_i$, and $k$ hash functions, we can create a bloom filter of size $m$ for each $S_i$. Let $Z_1$ (resp. $Z_2$) be the number of 0s in the filter $S_1$ (resp. $S_2$) and $Z_{12}$ be the number of 0s in the inner product of the bloom filters for $S_1$ and $S_2$. Using equation 1, we can compute the following approximations:

$$X_1 \cdot X_2 = |S_1 \cap S_2| \geq t \tag{3}$$

$$\iff -k|S_1 \cap S_2| \leq -kt \tag{4}$$

$$\iff (1 - \frac{1}{m})^{-k|S_1 \cap S_2|} \geq (1 - \frac{1}{m})^{-kt} \tag{5}$$

$$\iff \frac{Z_1 + Z_2 - Z_{12}}{Z_1 Z_2} \approx \frac{1}{m}(1 - \frac{1}{m})^{-k|S_1 \cap S_2|} \geq \frac{1}{m}(1 - \frac{1}{m})^{-kt} \tag{6}$$

$$\iff Z_1 + Z_2 - Z_{12} \geq Z_1 Z_2 \frac{1}{m}(1 - \frac{1}{m})^{-kt} \tag{7}$$

The above approximations could be used to give an efficient secure approximate protocol for checking whether $X_1 \cdot X_2 \geq t$ for given binary vectors $X_1$ and $X_2$ and threshold value $t$. The algorithm 1 gives the details of our secure approximate threshold dot product algorithm. In the algorithm, each party first creates its own bloom filter using the common bloom filter parameters (line: 3). Secondly, they participate in a secure dot product algorithm using their own private bloom filters and get the random shares of the dot product result (line: 5). Thirdly, each party participates in secure multiplication protocol using their private $Z_i$ values to get the random share of the multiplication result (line: 6). Finally, they combine their local inputs and random shares accordingly to get the final result (lines: 7-9).

We would like to stress that similar secure approximate dot product algorithm could be given using existing secure ln() function evaluation techniques [3] and the equation 2. For secure approximate dot product case, we need to use just one invocation of the secure ln() protocol to get the shares of the $ln(Z_1 + Z_2 - Z_{12})$. Similarly, $Z_{12}$ could be computed by using any secure dot product protocol.

### 2.1 Security

It is easy to prove that algorithm 1 is secure in semi-honest model [15] assuming that the dot product, multiplications and comparison protocols that are used are secure. Below, we give a sketch of the security proof.

**Theorem 1.** *Given secure dot product, multiplication and comparison protocols, algorithm 1 is secure in the semi-honest model.*

*Proof.* **(Sketch)** After secure multiplication and secure dot product protocols, each party only gets random shares. Therefore simulator for each $P_i$ could simply generate random values to simulate $u_i$ and $v_i$. Since each $y_i$ value only depends on the local or publicly known inputs, each party does not learn anything while computing $y_i$ values. Finally, each party participates in a secure comparison protocol to learn the final result. Again assuming that comparison protocol is secure, we can provide a simulator for each $P_i$ that can just return the final result. Using composition theorem [15], we can conclude our proof sketch.

### 2.2 Computational and Communicational Cost

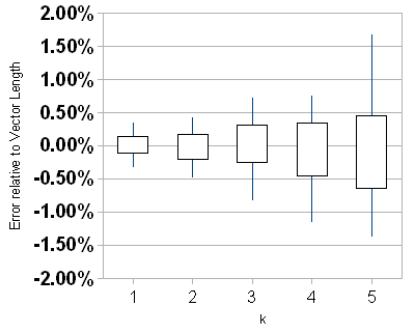
It is clear from the description of algorithm 1 that we need to use a secure dot product algorithm on binary vectors of size $m$ (i.e., the bloom filters). All the other operations are constant time and do not depend on either $n$ or $m$. In our method, we also have an additional cost of $O(nk)$ ($n$ is the size of the original binary vectors, $k$ is the number of hash functions used for bloom filter) hash function operations to create the bloom filters. As we show in the next section, the hashing cost is negligible compared to expensive public key operations used in the secure dot product protocol. As long as, $m << n$, our method would be much faster than the typical implementation of a secure dot product protocol. Even more, if more efficient techniques are found for securely computing dot product of two binary vectors, we may just use it to improve our performance even further. The communication cost of our protocol is also much lower, since it is only proportional to $m$. In the next section, we discuss the appropriate values of $m$ and $k$ that need to be used in practice for efficiency and accuracy.

## 3 Experimental Results

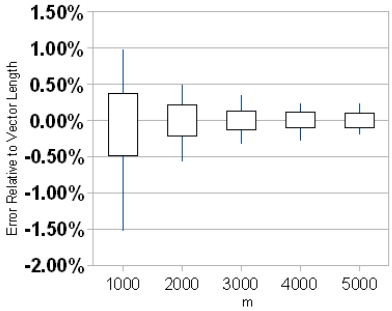
To evaluate the efficiency and accuracy of our approximation algorithm, we designed and ran several experiments. We considered the effects of vector length ($\ell$), vector density ($d$), the actual intersection of the two vectors ($i$), and the bloom filter parameters $m$ (length of filter) and $k$ (number of hash functions) on the performance of the algorithm. As an underlying secure dot product protocol, we used the completely secure homomorphic encryption-based protocol [10]. The Paillier key length was set to 512 bits. The hash functions used were keyed versions of SHA-1 via the HMAC protocol [16].

### 3.1 Accuracy

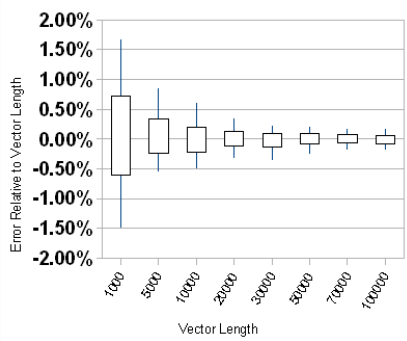
Several experiments were carried out to determine the accuracy of the approximate dot product method. The results are depicted in the figures below. Each data point represents the results of 100 trials, depicting the distribution in which the dot product estimates fell. Note that the bars on the figures represent plus or minus one standard deviation from the mean, while the extended lines denote the minimum and maximum error. Error is given relative to the vector length.



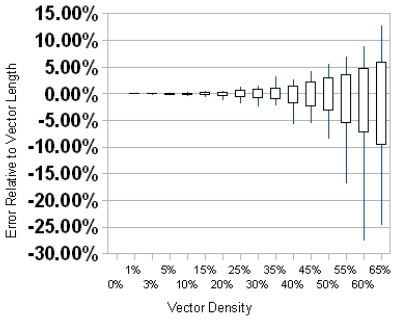

(a) Varying $k$



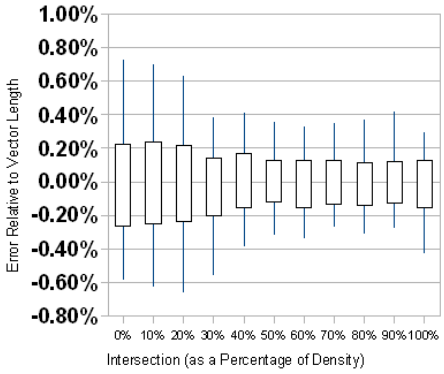

(b) Varying $m$




(c) Varying Vector Length




(d) Varying Vector Density




(e) Varying Intersection

| | |
|---|---|
| length of vector ($l$) | 20000 |
| number of filters ($k$) | 1 |
| length of filter ($m$) | 15% of $l$ |
| density ($d$) | 10% of $l$ |
| intersection | 50% of $d$ |

(f) Default Values

**Fig. 1.** Accuracy

Figure 1 shows the accuracy when different parameters are varied. Figure 1(f) gives the default values of the different parameters in all of the experiments. Figure 1(a) plots the accuracy for different values of $k$ (number of hash functions). The results show that accuracy actually decreases with increasing $k$. This

is because the distortion increases while increasing $k$ when the filter is small. In a separate experiment, it was determined that an increase in $k$ did increase accuracy for $m = 40000$, however, this is not a practical use of the protocol, since the Bloom filters would no longer give us the efficiency we desire. Overall, the experiment shows that for both accuracy and efficiency, nothing is gained by increasing $k$ for small filters.

In Figure 1(b), we study the relation of accuracy with filter length. We can see that the accuracy of the approximation increases as $m$ increases. This makes sense, since as the filter length increases, there is less distortion (and collision) in the data, and more accurate results occur. However, even in the lowest case of $m = 1000$ (95% data compression), the error was no more than 1.5% of the vector length, and the average absolute error was only around 9, meaning that with a significant probability, even a high compression ratio can result in high utility.

Next, we study the effect of vector length. Figure 1(c) reports accuracy results when the vector length varies from 1000 to 100000. The figure shows that the relative error decreases as the vector length increases. This makes sense since we keep the filter length a fixed percentage of the vector length. A key property of Bloom filters is that increase in vector length requires only sub-linear increase in the Bloom filter length to maintain similar accuracy. This helps us significantly for large vectors. As can be seen, the relative error was always within 2% of the vector length.

Next, we study the effect of varying the density of the vector. Figure 1(d) shows the relative error when the density was varied from 0% to 65%. The rest of the parameters were kept at the default values. As expected, the accuracy drastically fell off as the density of the vectors increased, although reasonable (within 5%) accuracy could still be achieved for vectors as dense as 35%. One interesting thing to note is the fact that at 0% density (that is, no 1's in either vector) the approximate algorithm will give an exact answer of 0, and there will be no error.

Finally, we vary the vectors to achieve the size of the intersections to be from 0% to 100% of their densities. The results are shown in figure 1(e). The error was, in all cases, less than 1%, although slightly greater error was produced toward the lower end of the intersection spectrum. We hypothesize that this is because the lower end is closer to the "average" dot product for two randomly generated 10% dense vectors, and is therefore less distinguishable. On another interesting note, when there was zero intersection between the two vectors, the approximation function sometimes produced a negative estimate. On the whole, though, the function produced exceptional accuracy, no matter what the actual intersection was.

### 3.2 Efficiency

We also demonstrate the efficiency of the algorithm. An experiment was carried out on a 2.8GHz dual core AMD Athlon machine with 2 GB of RAM to determine the run time efficiency of the approximation algorithm versus the run

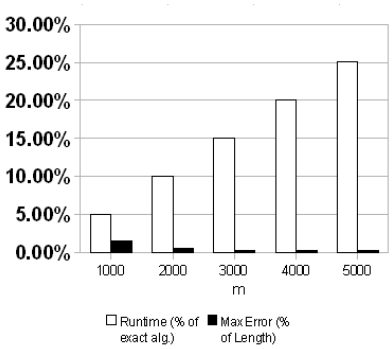



**Fig. 2.** Expected Run Time vs. Max Error

time efficiency of the exact algorithm. On two vectors of 20000 elements, whose density was 10%, (that is each vector contained 2000 1's) and whose intersection was 50% of the density (that is, 1000), the exact version took 27 minutes and 51 seconds to run. On the same vectors, the approximation algorithm with bloom filter parameters $m = 3000$, $k = 2$, ran in only 4 minutes and 4 seconds.

We also compared the relative run times of the public key encryption used versus the hash functions used in the bloom filter. On average, a single Paillier encryption ($t_e$) took 82.454ms, a Paillier decryption ($t_d$) took 81.583ms, and the hash function computation ($t_h$) was 0.00418ms. Since, with the approximation algorithm, we need only run $m$ public key encryptions instead of the full $\ell$ public key encryptions, we can account for the vast majority of the difference in the algorithms' run times. In fact, the approximate runtime of the approximate algorithm is given by $t_e m + t_d + t_h \ell \cdot d \cdot k$. By comparison, the runtime of the exact algorithm is given by $t_e \ell + t_d$. Since $m < \ell$, usually significantly (80-90%), and $t_h << t_e$, the runtime of the approximate algorithm is significantly lower than the exact algorithm. Figure 2 is a chart of the expected run times (relative to the exact algorithm) of the previous experiments varying $m$, together with the maximum error observed in any run with that given $m$ (relative to the vector length), in order to show the trade-off between accuracy and efficiency. This clearly demonstrates the great accuracy/efficiency trade-off of our algorithm.

## 4 Privacy-Preserving Association Rule Mining

We now show how the protocol described above would affect the process of privacy-preserving assocation rules mining. We assume that the data is vertically partitioned – each site holds some attributes of each transaction, and the sites wish to collaborate to identify globally valid association rules. However, the sites must not reveal individual transaction data.

The association rule mining problem can be formally stated as follows[17]: Let $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ be a set of literals, called items. Let $\mathcal{D}$ be a set of transactions, where each transaction $T$ is a set of items such that $T \subseteq \mathcal{I}$. Associated

with each transaction is a unique identifier, called its *TID*. We say that a transaction $T$ *contains* $X$, a set of some items in $\mathcal{I}$, if $X \subseteq T$. An *association rule* is an implication of the form $X \Rightarrow Y$, where $X \subset \mathcal{I}$, $Y \subset \mathcal{I}$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ holds in the transaction set $\mathcal{D}$ with **confidence** $c$ if $c$% of transactions in $\mathcal{D}$ that contain $X$ also contain $Y$. The rule $X \Rightarrow Y$ has **support** $s$ in $\mathcal{D}$ if s% of the transactions in $\mathcal{D}$ contain $X \cup Y$. Vaidya and Clifton[6] show that this problem can be reduced to the simple calculation of a zero-one dot product of vectors whose length is the size of the transaction set.

Our experimental results show that the Bloom filter approximation method should apply to association rule mining very well. In terms of accuracy, the Bloom filter approximation method is extremely good for sparse vectors, which tends to be the norm in association rule mining. In addition, the filter length can be increased more gradually for larger vectors, allowing the algorithm to scale better than linearly in terms of efficiency.

As an example, consider an association rule mining whose required confidence is 70%, and support is 7%. Using our protocol, assuming 10% of the sample had entries for a given item on either side, and based on figure 1(e), the absolute least confidence that a selected rule could have is 65%, and the absolute most that an unselected rule could have is 75%, and with a high probability, most selected would have at least 68% and most not selected would have at most 72%. The number of false positives and false negatives, therefore, would be quite small, even given large data compression.

## 5 Conclusion

The main contribution of paper is an efficient and provably secure protocol that approximately computes the scalar (dot) product of two vectors based on Bloom Filters. We show how this can be used to mine association rules in a privacy-preserving way. Our experimental results show that protocol is highly efficient, accurate and easily applicable to association rule mining. The dot product is actually useful in many different privacy-preserving data mining protocols beyond association rule mining. In the future, we intend to explore our protocol's applicability in other tasks. We also plan on extending it to work for more than two parties. Other dimension reduction techniques could also be used to approximate the vector space. Using these, the scalar product could also be computed. Different techniques may have different trade offs of security, efficiency and accuracy. In the future, we plan to look at these alternatives such as using space filling curves or techniques utilizing the Johnson-Lindenstruass theorem.

## References

1. Vaidya, J., Clifton, C., Zhu, M.: Privacy-Preserving Data Mining, 1st edn. Advances in Information Security, vol. 19. Springer, Heidelberg (2005)
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of the 2000 ACM SIGMOD Conference on Management of Data, Dallas, TX, May 14-19, pp. 439–450. ACM, New York (2000)

3. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 36–54. Springer, Heidelberg (2000)
4. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: On the privacy preserving properties of random data perturbation techniques. In: Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003), November 19-22, IEEE Computer Society, Los Alamitos (2003)
5. Huang, Z., Du, W., Chen, B.: Deriving private information from randomized data. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, June 13-16. ACM, Baltimore (2005)
6. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, July 23-26, pp. 639–644. ACM, Alberta (2002)
7. Kantarcıoğlu, M., Clifton, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned data. IEEE Transactions on Knowledge and Data Engineering 16(9), 1026–1037 (2004)
8. Du, W., Atallah, M.J.: Privacy-preserving statistical analysis. In: Proceeding of the 17th Annual Computer Security Applications Conference, New Orleans, Louisiana, USA, December 10-14 (2001)
9. Ioannidis, I., Grama, A., Atallah, M.: A secure protocol for computing dot-products in clustered and distributed environments. In: The 2002 International Conference on Parallel Processing, Vancouver, British Columbia, August 18-21 (2002)
10. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On Private Scalar Product Computation for Privacy-Preserving Data Mining. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 104–120. Springer, Heidelberg (2005)
11. Ravikumar, P., Cohen, W.W., Fienberg, S.E.: A secure protocol for computing string distance metrics. In: Proc. the Workshop on Privacy and Security Aspects of Data Mining at the Int. Conf. on Data Mining, pp. 40–46 (2004)
12. Qiu, L., Li, Y., Wu, X.: Preserving privacy in association rule mining with bloom filters. J. Intell. Inf. Syst. 29(3), 253–278 (2007)
13. Vaidya, J., Clifton, C.: Secure set intersection cardinality with application to association rule mining. Journal of Computer Security 13(4), 593–622 (2005)
14. Mitzenmacher, A.B.M.: Network applications of bloom filters: A survey. Internet Mathematics 1(4), 485–509 (2005)
15. Goldreich, O.: General Cryptographic Protocols. In: The Foundations of Cryptography, vol. 2, pp. 599–764. Cambridge University Press, Cambridge (2004)
16. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
17. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Buneman, P., Jajodia, S. (eds.) Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C, May 26–28, pp. 207–216 (1993)