Towards Efficient Privacy-Preserving Collaborative Recommender Systems

Justin Zhan[+], I-Cheng Wang[*], Chia-Lung Hsieh[+],
Tsan-Sheng Hsu[*], Churn-Jung Liau[*], Da-Wei Wang[*]
[+]The Heinz School,
Carnegie Mellon University, USA
Email: {justinzh, chialunh}@andrew.cmu.edu
[*]Institute of Information Science,
Academia Sinica, Taiwan
Email: {icw, tshsu, liaucj, wdw}@iis.sinica.edu.tw

Abstract

Recommender systems use various types of information to help customers find products of personalized interest. To increase the usefulness of recommender systems in certain circumstances, it could be desirable to merge recommender system databases between companies, thus expanding the data pool. This can lead to privacy disclosure hazards that this paper addresses by constructing an efficient privacy-preserving collaborative recommender system based on the scalar product protocol.

1 Introduction

A recommender system [9] is a web-based application best known for its usage on e-commerce websites, with the aim of helping customers in the decision making and product selection process by providing a list of recommended items. The most prominent example is the online bookstore Amazon.com, where collaborative filtering techniques are used to find similarities in users' profiles based on their navigation and buying history. The goal is to identify users who presumably have similar preferences and recommend items that were bought by these related users. Another technical approach is content-based filtering, which builds on the hypothesis that the preferred items of a single user can be extrapolated from their preferences in the past. The third approach is to use domain knowledge to base the recommendations on a thorough understanding of the user's current needs, comparable to real-life sales situations. The recommendations are the result of a reasoning process on domain knowledge that also forms the basis for explaining to the user why an item is proposed. Knowledge-based recommender systems explicitly elicit user preferences, i.e., they provide dynamic personalized, and potentially persuasive, sales dialogues.

Recommender systems can help consumers find the most valuable items by calculating the similarities among other consumers with collaborative filtering algorithms. From the business point of view, recommender systems have the potential to increase sales, because purchasing decisions are often strongly influenced by people who the consumer knows and trusts. In the networked virtual world, consumers also need some word of mouth to support their purchasing decisions, thus, the best source will be recommender systems. Recommender systems can integrate information from product rating matrices and user preference similarity matrices to generate personalized recommendations. It can also help corporations maximize the precision of targeted marketing.

Im and Hars [6] have claimed that the accuracy of a recommender system increases as the total number of users increases. This implies that accuracy of a recommender system decreases when the total number of users is limited. One way to solve this problem is to join recommender systems if they have similar product sets. By joining recommender systems, the user sets are enlarged, which means more accurate recommendation can be made and the precision of targeted marketing is enhanced. In combining recommender systems, consumers and companies may worry about the risk of privacy disclosure.

Schafer et al. [10] have come up with a detailed

taxonomy of recommender systems through analyzing famous e-commerce websites including Amazon.com, CDNOW, eBay, Levi Strauss & Co., Moviefinder.com, and Reel.com. According to their findings, recommender systems can be categorized into non-personalized recommendations, attribute-based recommendations, item-to-item correlation, and people-to-people correlation. Each of these taxonomies has different degrees of automation and persistence. For privacy issues, Canny [3] has proposed some schemes for privacy-preserving collaborative filtering. In his schemes, there is a community of users who can compute the aggregation of their private data without disclosing it. Another approach is the randomized perturbation approach proposed by Polat and Du [8]. They deployed a centralized server to store the perturbed numeric ratings, and then used these disguised ratings to provide predictions to users. Berkovsky et al. [1] have proposed an obfuscation scheme about accuracy and privacy to decentralize the rating profiles among multiple repositories. Thus, they can have more users to improve accuracy and to mitigate privacy issues. However, their approach cannot achieve 100% accuracy unless all the private data is disclosed. Hsieh et al. [5] have proposed a scheme based on homomorphic encryption that provides 100% accuracy. In this paper, we will present a more efficient privacy-preserving approach than the existing encryption approaches.

In section 2, we will introduce a recommender system algorithm. In section 3, we will define our problem. In section 4, we will describe solutions to privacy-preserving collaborative recommender systems. In section 5, we will present the experimental results. We will further discuss the experimental results in section 6.

2 Recommender Systems

Schafer et al. [10] define the recommender system as a system that uses the opinions of members of a community to help individuals in that community find information or products most likely to be interesting to them or relevant to their needs. There are two basic entities concerned in a recommender system. The *user* (also referred to as customer) is a person who uses the recommender system to provide their opinion and receive recommendation about items. The *item* (also referred to as product) is being rated by users. The inputs of a recommender system are usually arithmetic rating values, which express the users' opinion of items. Ratings are normally provided by the user and follow a specified numerical scale (example: 1-bad to 5-excellent). The outputs of a recommender system can be either predictions or recommendations. The following are the three main processes of recommender systems.

2.1 Representation

In the original representation, the input data is defined as a collection of numerical ratings of m users on n items, expressed by the $m*n$ user-item matrix R. We call this user-item matrix of the input data set, *original representation*. As mentioned earlier, users are not required to provide their opinion on all items. As a result, the user-item matrix is usually sparse, including numerous *no rating* values, making it harder for filtering algorithms to generate satisfactory results. Thus, some techniques, whose purposes are to reduce the sparsity of the initial user-item matrix, have been proposed in order to improve the results of the recommendation process.

2.2 Neighborhood Formation

The core step of the recommendation process is determining the similarity between users in the user-item matrix R. Users similar to the active user U_a will form a proximity-based neighborhood with U_a. The active user's neighborhood should then be used in the following step of the recommendation process in order to estimate their possible preferences. Neighborhood formation has been implemented by calculating the similarity between all the users in the user-item matrix, R, with the help of proximity metrics.

The proximity between two users is usually measured using correlation or cosine measures.

- Pearson Correlation Similarity

  To find the proximity between users U_i and U_k, we can use the Pearson correlation metric.

$$sim_{ik} = corr_{ik} = \frac{\sum_{j=1}^{l}(r_{ij} - \overline{r_i})(r_{kj} - \overline{r_k})}{\sqrt{\sum_{j=1}^{l}(r_{ij} - \overline{r_i})^2 \sum_{j=1}^{l}(r_{kj} - \overline{r_k})^2}}$$

  It is important to note that the summations of j are calculated over L items for which both users u_i and u_k have expressed their opinions. Obviously, $L \leq n$, where n represents the number of total items in the user-item matrix R.

- Cosine Similarity

In the n-dimensional item space, we can view different users as feature vectors. A user vector consists of n feature slots, one for each available item. The values used to fill those slots can either be the rating r_{ij} that a user u_i provided for the corresponding item, ij, or 0, if no such rating exists. Now we can compute the proximity between two users, u_i and u_k, by calculating the similarity between their vectors as the cosine of the angle is formed between them.

Based on the results of Breese et al. [2], Pearson Correlation is considered a better metric for similarity calculations in recommender systems. Thus, we will use Pearson correlation similarity.

2.3 Recommendation Generation

The final step in the recommendation process is to produce either a prediction, which will be a numerical value representing the predicted opinion of the active user, or a recommendation that will be expressed as a list of the top-N items that the active user will appreciate. In both cases, the result should be based on the neighborhood of users.

3 Problems

Let us assume there are two e-commerce entities, for example, online bookstores, both of which have similar product sets, but with different customer sets. Also, both of them already have their own recommender systems with some data records. These two entities want to cooperate with each other to strengthen their recommender system databases and improve the precision of their recommendations for their own customers. For merging the recommender databases while not disclosing the actual commercial data, we have to check the recommender system algorithm to find the vulnerability of potential privacy disclosure.

Among the three steps in the recommender system algorithm, representation and recommendation generation are only related to the accuracy of recommendations provided to customers. The neighborhood formation step is the source of possible privacy disclosure. In the neighborhood formation step, we measure the proximity between two customers by calculating the Pearson correlation similarity. For example, let us assume that, for item(product) j, the ratings of r_{ij} and r_{kj} are made by users u_i and u_k from different e-commerce entities. While joining these two recommender system databases, we have to share the values $(r_{ij} - \overline{r_i})$ and $(r_{kj} - \overline{r_k})$ with each other. But with value $(r_{ij} - \overline{r_i})$, others can see how much user i prefers item j compared to their average rating, r_i.

4 Privacy-Preserving Collaborative Recommender System

The database is separated into two parts, A and B, which both need each other's data to compute the correlation coefficient without disclosing their own data. The privacy issue is on the numerator of the Pearson correlation coefficient, which is a scalar product. A has $\overrightarrow{X_a}$ and B has $\overrightarrow{X_b}$. We want to know the scalar product of $\overrightarrow{X_a}$ and $\overrightarrow{X_b}$ without disclosing $\overrightarrow{X_a}$ or $\overrightarrow{X_b}$. We will introduce two approaches: the homomorphic encryption based approach and the scalar product based approach.

4.1 Homomorphic Encryption Approach

Within the homomorphic encryption framework, we introduce two approaches: one is based on ElGamal homomorphic encryption [4]; the other is Paillier homomorphic encryption [7]. ElGamal encryption provides the multiplicative homomorphism that works as follows: the multiplication of two cypher texts equals the encryption of the multiplication of the plain texts. Paillier encryption provides the additive homomorphism, where the multiplication of two encrypted pieces of data equals the encryption of the addition of the plain text.

To compute the Pearson correlation similarity, we need the operators of $(r_{ij} - \overline{r_i})$ from one party and $(r_{kj} - \overline{r_k})$ from the other one. Let us assume that no party wants to take the risk of disclosing customer preferences. Because the computations are multiplications, we can use the homomorphic property of the ElGamal encryption. Two parties can encrypt their data on their own with a public key, and then after the multiplication computation, the multiplication of two encrypted pieces of data will become the encryption of the multiplication of data. Thus, the private data of two parties can be preserved during the similarity computation. Through any of the above approaches, the privacy of user preference can be preserved throughout the computation of similarities.

4.2 The Scalar Product Approach

The problem of Secure Multiparty Computation (SMC) was first addressed by Yao in his seminal paper, "Protocols for Secure Computations" [13]. The security of these solutions is based on cryptographic assumptions, such as the existence of trapdoor permuta-

tions. The solutions are generic and elegant, but their prohibitive cost in protecting privacy makes them unsuitable for large-scale applications. Therefore, practical solutions need to be developed. We will introduce the secure two-party product protocol proposed in [14], which has been proven information-theoretically secure [11].

PROTOCOL π.
$f_2(x_a, x_b) \mapsto (y_a, y_b)$, where $x_a \cdot x_b = y_a + y_b$.

1. The commodity server generates random vectors R_a, R_b, and random number r_a, and lets $r_b = R_a \cdot R_b - r_a$. It then sends (R_a, r_a) to *Ailce* and (R_b, r_b) to *Bob*.
2. *Alice* sends $\overrightarrow{X_a}' = \overrightarrow{X_a} + \overrightarrow{R_a}$ to *Bob*.
3. *Bob* sends $\overrightarrow{X_b}' = \overrightarrow{X_b} + \overrightarrow{R_b}$ to *Alice*.
4. *Bob* computes $t = \overrightarrow{X_a}' \cdot \overrightarrow{X_b} + r_b - y_b$ and sends it to *Bob*, where y_b is a randomly generated number.
5. *Alice* computes $y_a = t - \overrightarrow{X_b}' \cdot \overrightarrow{R_a} + r_a$.

Theorem 1 (Secure Two-party Product Protocol) *π is an information-theoretically secure protocol that implements function f_2 in the semi-honest adversary model with private channels.*

Proof 1 *Please refer to [11] for the detailed proof.*

It has been shown that the commodity-based scalar product protocol is a very efficient approach compared to existing benchmarks [12]. The scalar product approach is based on the above scalar product protocol. This approach needs a neutral commodity server to generate random numbers r_a, $\overrightarrow{R_a}$ for A and r_b, $\overrightarrow{R_b}$ for B. A and B exchange $\overrightarrow{X_a}' = \overrightarrow{X_a} + \overrightarrow{R_a}$ and $\overrightarrow{X_b}' = \overrightarrow{X_b} + \overrightarrow{R_b}$. B then computes $t = \overrightarrow{X_a}' \cdot \overrightarrow{X_b} + r_b - y_b$ and sends it to A, where y_b is a randomly generated number. Finally, A computes $y_a = t - \overrightarrow{X_b}' \cdot \overrightarrow{R_a} + r_a$. The result of the scalar product equals the sum of y_a and y_b. The summation and multiplication computation in the numerator of the Pearson correlation formula is a scalar product computation. It is straightforward to adapt the commodity-based scalar product approach, where the plain texts from two parties can be preserved and the computation of similarities can be correctly computed.

5 Performance Evaluation

In this section, we will implement both the homomorphic encryption approach and the scalar product approach. Based on our experiments, the ElGamal algorithm is about five times faster in a single operation than the Paillier algorithm. Thus, we will only compare ElGamal approach with both the original and revised commodity based scalar product approaches.

We choose the same database as Hsieh et al. [5], which is the "Jester Joke Recommender System", released by Ken Goldberg from the University of California at Berkley (http://shadow.ieor.berkeley.edu/humor/). It has 4.1 million continuous ratings (-10.00 to +10.00) of 100 jokes from 73,421 users, collected between April 1999 and May 2003. We take the densest sub-dataset of ratings from 23,500 users who have rated 36 or more jokes, which is a matrix with dimensions of 23,500 * 101. For the representation process of recommendation generation, we add the default value 0 for the items not rated. The first column of every row stores how many items are rated by the user, which is not necessary for computing the Pearson correlation coefficient. Therefore, we simply ignore the first column of the database.

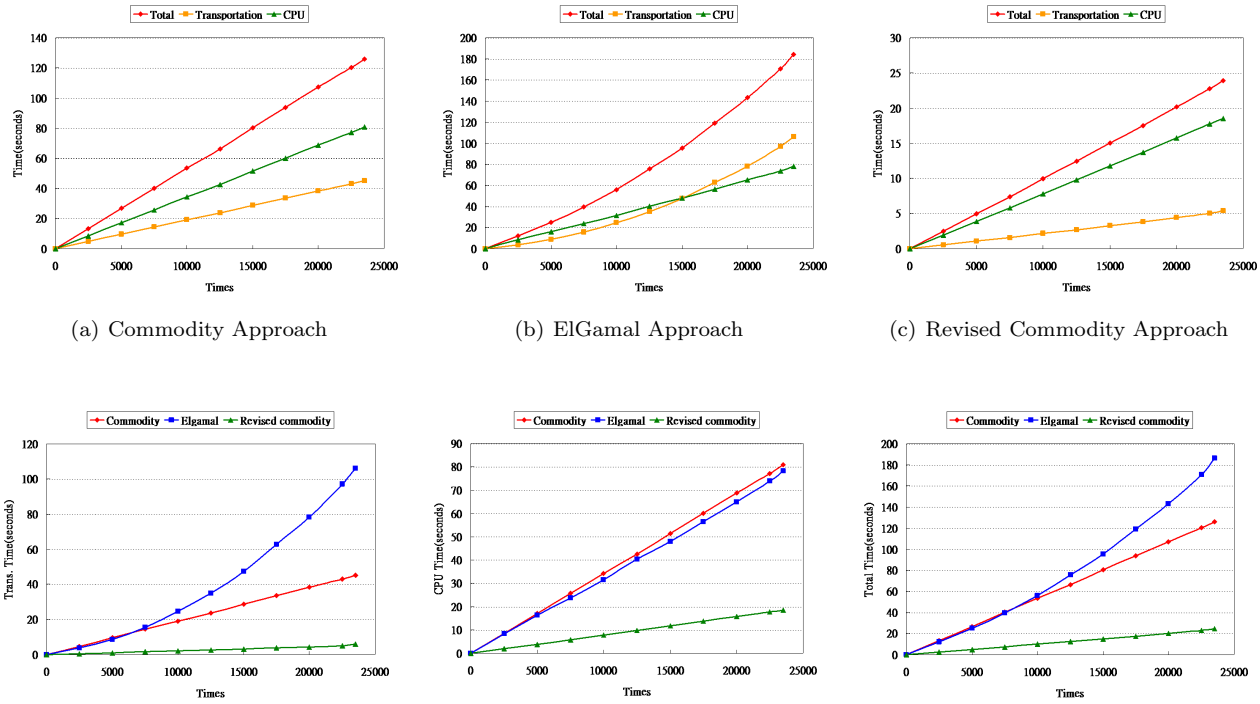
Consider that Alice has one user rating ($\overrightarrow{A}$) while Bob has 23499 ($\overrightarrow{B_1}$ to $\overrightarrow{B_{23499}}$). Without disclosing their original data, they want to know the Pearson correlation coefficient sim_{AB_1}, sim_{AB_2}, ..., sim_{AB_n} ($1 \leq n \leq 23499$). We implemented these approaches with Ruby 1.8.6. Alice's code runs on a server with an AMD Opteron 2.8 GHz processor, and 4GB DDR2 RAM, while Bob's code runs on a server with an Intel Xeon 3.0 processor and 4GB DDR2 RAM.

To minimize the probabilistic variation, our experimental results are the average of 100 effective executions. The experiments focus on different numbers of user data for secure two party computation: execution time, transportation time, and CPU time. The amount of CPU time is the aggregation result by adding Alice and Bob's CPU time. The transportation time is equal to the difference of the execution time and the CPU time.

Since the scalar product based approach is integer-based, the inputs must be positive integers. Let $X, min,$ and dig represent the original data, the effective minimum rating, and the effective decimal digits of the database, respectively. Let

$$Y = \begin{cases} (X - min) \times 10^{dig}, & \text{if } min < 0 \\ X \times 10^{dig}, & \text{otherwise} \end{cases}$$

be our integer input data. It is trivial to prove that, after replacing X with Y, they will still have the same Pearson correlation coefficient. The following section introduces each approach we implemented and show the experimental results.



(a) Commodity Approach (b) ElGamal Approach (c) Revised Commodity Approach

(d) Transportation Time Comparison (e) CPU Time Comparison (f) Total Execution Time Comparison

Figure 1. Experimental Results

- Commodity Approach

  We use *Commodity Approach* to stand for the original commodity server based scalar product approach. Each round can compute a scalar product. In other words, if there are N scalar products needed to be computed, simply run this protocol N times. Figure 1(a) shows the total execution, transportation and CPU time of this approach. As expected, all of them are linear.

- ElGamal Approach

  We implemented the same algorithm as that of Hsieh et al.[5] to compare the performance of privacy-preserving recommender systems with that of others. A neutral third party is unnecessary in this approach since only two random numbers are needed for Alice and Bob's private keys. Figure 1(b) shows the total execution, transportation, and CPU time of this approach.

- Revised Commodity Approach

  It is much more reasonable to make sure that all the needed pieces of data are ready before executing any multi-party computation. As a result, we pre-produce and transport the random numbers to A and B before executing the tasks. In other words, the commodity server, which is the neutral third party producing only random numbers, is not necessarily included in computing transportation, CPU, and total execution time. Both A and B know that they are going to do N scalar products, so they exchange all the needed information in one round, rather than in N rounds. Figure 1(c) shows the total execution, transportation, and CPU time of this approach.

6 Discussion

Figures 1(d), 1(e), and 1(f) compare the results of transportation, CPU, and total execution time among the aforementioned approaches. The revised commodity-based approach has the lowest time cost and needs the least computing resources among the three approaches. Therefore, we can conclude that the revised commodity-based approach has the best performance. However, additional storage is needed for random numbers. At the same time, the stored random numbers must not be disclosed to each party. In Figure 1(d), the ElGamal's transportation time line is not linear. A possible reason for this is that Ruby uses a fixed-size buffer for IO. It may become non-linear once the transportation data

becomes too large. To show the total time cost for a 100 dimensional scalar product, we list the experimental results for each approach in the following table.

Approach	ElGamal	Comm.	Revised Comm.
Time(sec)	0.10301	0.00515	0.00178

7 Conclusion and Future Works

Because of the development of e-commerce, recommender systems have become more and more popular. Customers may not trust imprecise recommendations from a recommender system that has a limited database. It is then desirable for e-commerce entities with limited databases to merge their recommender system databases to enhance the reliability of recommendations for customers and to maximize the precision of targeted marketing while preserving the privacy of customer preferences. With the algorithms introduced in this paper, e-commerce entities can merge their recommender system databases without disclosing customers' private data. In future works, we will design and implement a prototype of this privacy-preserving collaborative recommender system with the proposed approaches.

References

[1] S. Berkovsky, Y. Eytani, T. Kuflik, and F. Ricci. Enhancing privacy and preserving accuracy of a distributed collaborative filtering. In *the Proceedings of the 2007 ACM conference on Recommender systems*, pages 9–16, 2007.

[2] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, 1998.

[3] J. Canny. Collaborative filtering with privacy via factor analysis. In *the Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245, 2002.

[4] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory, Vol. IT-31, No.4, 1985, pp469472*, 1985.

[5] C. Hsieh, J. Zhan, D. Zeng, and F. Wang. Preserving privacy in joining recommender systems. In *International Conference on Information Security and Assurance (ISA 2008)*, pages 561–566, 2008.

[6] I. Im and A. Hars. Does a one-size recommendation system fit all? the effectiveness of collaborative filtering based recommendation systems across different domains and search modes. *ACM Transaction of Information Systems*, 26(1):4, 2007.

[7] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptography - EUROCRYPT '99, pp 223-238, Prague, Czech Republic*, 1999.

[8] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *the Third IEEE International Conference on Data Mining*, 2003.

[9] P. Resnick and H. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.

[10] J. Schafer, J. Konstan, and J. Riedi. Recommender systems in e-commerce. In *the Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166, 1999.

[11] C. Shen, J. Zhan, D. Wang, T. Hsu, and C. Liau. Information theoretically secure number product protocol. In *International Conference on Machine Learning and Cybernetics, August 19-22, HongKong, 2007*, 2007.

[12] I. Wang, C. Shen, J. Zhan, T. Hsu, C. Liau, and D. Wang. Towards empirical aspects of secure scalar product. *IEEE Transaction of Systems, Man, and Cybernetics, Part C, to appear*, 2008.

[13] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, 1982.

[14] Z. Zhan and L. Chang. Privacy-preserving collaborative data mining. In *IEEE Intertional Workshop of Foundations and New Directions in Data Mining, Melbourne, Florida, USA November 19 - 22*, 2003.