Privacy Enhanced Recommender System

Zekeriya Erkin[1] Michael Beye[1] Thijs Veugen[1,2] Reginald L. Lagendijk[1]

[1]Information Security and Privacy Lab, Faculty of EEMCS
Delft University of Technology, 2628 CD, Delft, The Netherlands
[2]TNO Information and Communication Technology
P.O. Box 5050, 2600 GB Delft, The Netherlands
{z.erkin, m.r.t.beye, p.j.m.veugen, r.l.lagendijk}@tudelft.nl

Abstract

Recommender systems are widely used in online applications since they enable personalized service to the users. The underlying collaborative filtering techniques work on user's data which are mostly privacy sensitive and can be misused by the service provider. To protect the privacy of the users, we propose to encrypt the privacy sensitive data and generate recommendations by processing them under encryption. With this approach, the service provider learns no information on any user's preferences or the recommendations made. The proposed method is based on homomorphic encryption schemes and secure multiparty computation (MPC) techniques. The overhead of working in the encrypted domain is minimized by packing data as shown in the complexity analysis.

Keywords: Recommender systems, user privacy, secure multiparty computation, homomorphic encryption, data packing.

1 Introduction

In the last decade, we have experienced phenomenal progress in information and communication technologies. Cheaper, more powerful, less power consuming devices and high bandwidth communication lines enabled us to create a new virtual world in which people mimic activities from their daily lives without the limitations imposed by the physical world. Online shopping, banking, communicating and much more have become common for millions of people [1].

Personalization is a common approach to attract even more people to online services. Instead of making general suggestions for the users of the system, the system can suggest personalized services targeting only a particular user based on his preferences [2]. Since the personalization of the services offers high profits to the service providers and poses interesting research challenges, research for generating recommendations, also known as collaborative filtering, attracts attention both from academia and industry.

The techniques for generating recommendations for users strongly rely on the information gathered from the user. This information can be provided by the user himself as in profiles or the service provider can observe user's actions like click logs. On one hand, more information on the user helps the system to improve the accuracy of the recommendations. On the other hand, the information on the users creates a severe privacy risk since there is no solid guarantee for the service provider not to misuse the user's data. It is often seen that whenever a user enters the system, the service provider claims the ownership of the information provided by the user and authorizes itself to distribute the data to third parties for its own benefits [13].

In this paper, we propose a cryptographic solution for preserving the privacy of users in a recommender system. In particular, the privacy-sensitive data of the users are kept encrypted and the service provider generates recommendations by processing encrypted data. The cryptographic protocol developed for this purpose is based on homomorphic encryption [3] and secure multiparty computation (MPC) techniques [14]. While the homomorphic property is used for realizing linear operations, protocols based on MPC techniques are developed for non-linear operations (e.g. finding the most similar users). The overhead introduced by working in the encrypted domain is reduced considerably by data packing as shown in complexity analysis.

2 Related Work

In [4], Canny proposes a system where the private user data is encrypted and recommendations are generated by applying an iterative procedure based on conjugate gradient algorithm. The algorithm computes a characterization matrix of the users in a subspace and generates recommendations by calculating reprojections in the encrypted domain. Since the algorithm is iterative, it takes many rounds for convergence and in each round users need to participate in an expensive decryption procedure which is based on a threshold scheme where a significant portion of the users are assumed to be online and honest. The output of each iteration, which is the characterization matrix, is available in clear. In [5], Canny proposes a method to protect the privacy of users based on a probabilistic factor analysis model by using a similar approach as in [4].

While Canny works with encrypted user data, Polat and Du suggest to protect the privacy of users by using randomization techniques [11, 12]. In their paper, they blind the user data with a known random distribution assuming that in aggregated data this randomization cancels out and the result is a good estimation of the intended outcome. The success of this method highly depends on the number of users participating in the computation since for the system to work, the number of users need to be in vast amounts. This creates a trade-off between accuracy/correctness of the recommendations and the number of users in the system. Moreover, the outcome of the algorithm is also available to the server who constitutes a privacy threat to the users. Finally, the randomization techniques are believed to be highly insecure [15].

3 Generating Recommendations

A centralized system for generating recommendations is a common approach in e-commerce applications. To generate recommendations for user A, the server follows a two-step procedure. In the first step, the server searches for users similar to user A. Each user in the system is represented by a preference vector which is usually composed of ratings for each item within a certain range. Finding similar users is based on computing similarity measures between users' preference vectors. Pearson correlation is a common similarity measure (Eq. 1) for two users with preference vectors $V_A = (v_{(A,0)}, \ldots, v_{(A,M-1)})^T$ and $V_B = (v_{(B,0)}, \ldots, v_{(B,M-1)})^T$, respectively, where M is the number of items and $\bar{v}$ represents the average value of the vector v.

$$\mathrm{sim}_{A,B} = \frac{\sum_{i=0}^{M-1}(v_{(A,i)} - \overline{v}_A) \cdot (v_{(B,i)} - \overline{v}_B)}{\sqrt{\sum_{i=0}^{M-1}(v_{(A,i)} - \overline{v}_A)^2 \cdot \sum_{i=0}^{M-1}(v_{(B,i)} - \overline{v}_B)^2}}. \quad (1)$$

Once the similarity measure for each user is computed, the server proceeds with the second step. In this step, the server chooses the first L users with similarity values above a threshold δ and averages their ratings. These average ratings are then presented as *recommendations* to user A.

In e-commerce applications the number of items offered to users are usually in the order of hundreds or thousands. Apart from many smart ways of determining the likes and dislikes of users for the items, we assume the users are asked to rate the items explicitly with integer values in the range of $[0, K]$. Regarding the vast number of items and users' rating behavior, the data matrix is usually highly sparse, meaning that most of the items are not rated. Finding similar users in a sparse dataset can easily lead the server to generate inaccurate recommendations. To cope with this problem, one approach is to introduce a small set of items that is rated by most users. Such a base set can be explicitly given to the users or implicitly chosen by the server from the most commonly rated items. Having a small set of items that is rated by most users, the server can compute similarities between users more confidently, resulting in more accurate recommendations. Therefore, we assume that the user preference vector V is split into two parts: the first part consists of R elements that are rated by most of the users and the second part contains $M - R$ partly rated items that the user would like to get recommendations on [2].

4 Cryptographic Primitives and Security Model

We use encryption to protect user data against the service provider and other users. A special class of cryptosystems, namely homomorphic cryptosystems, allows us to process the data in the encrypted form. We chose the Paillier cryptosystem [10] as it is *additively homomorphic* meaning that the product of two encrypted values $[a]$ and $[b]$, where $[\cdot]$ denotes the encryption function, corresponds to a new encrypted message whose decryption yields the sum of a and b as $[a] \cdot [b] = [a + b]$. As a consequence of the additive homomorphism, any ciphertext $[m]$ raised to the power of a public value c corresponds to the multiplication of m and c in the encrypted domain: $[m]^c = [m \cdot c]$. In addition to the homomorphism property, the Paillier cryptosystem is semantically secure implying that each encryption has a random element that results in different ciphertexts for the same plaintext.

As a part of a cryptographic protocol introduced in Section 6, we use another additively homomorphic and semantically secure encryption scheme, DGK [7, 6]. The DGK is replaced with the Paillier cryptosystems in a subprotocol for efficiency reasons. Due to its much smaller message space, encryption and decryption operations are more efficient than Paillier cryptosystem.

We use the semi-honest security model, which assumes that all players follow the protocol steps but are curious and thus keep all messages from previous and current steps to extract more information than they are allowed to have. Our protocol can be adapted to the active attacker model by using the ideas in [9] with additional overhead.

5 Privacy-Preserving Recommender System

In this section we propose a protocol based on additively homomorphic encryption schemes and MPC techniques. In particular the service provider, i.e. the server, receives the encrypted rating vector of user A and sends it to the other users in the system who can then compute the similarity value on their own by using the homomorphism property of the encryption scheme. Once the users compute the similarity values, they are sent to the server. After that, the server and user A runs a protocol to determine the similarity values that are above a threshold δ. The server, being unaware of the number users with a similarity value above a threshold and their identities, accumulates the ratings of all users in the encrypted domain. Then, the encrypted sum is sent to user A along with the encrypted number of similarities above the threshold, L. User A decrypts the sum and L and, computes the average values, obtaining the recommendations. Each step of the proposed protocol is detailed in the following sections.

5.1 Key Generation and Preprocessing

Any user in the system who wants to get recommendations generates personal public key pairs for the Paillier and the DGK cryptosystems. We assume that the public keys of the users are available publicly.

Since the Pearson correlation given in (1) for user A and B can be also written as:

$$\text{sim}_{A,B} = \sum_{i=0}^{R-1} \underbrace{\frac{(v_{(A,i)} - \overline{v}_A)}{\sqrt{\sum_{i=0}^{R-1}(v_{(A,i)} - \overline{v}_A)^2}}}_{C_1} \cdot \underbrace{\frac{(v_{(B,i)} - \overline{v}_B)}{\sqrt{\sum_{i=0}^{R-1}(v_{(B,i)} - \overline{v}_B)^2}}}_{C_2}, \tag{2}$$

the terms C_1 and C_2 can be easily computed by users A and B, respectively. Each user computes a vector from which the mean is subtracted and normalized. Since the elements of the vector are real numbers and cryptosystems are only defined on integer values, they are all scaled by a parameter f and rounded to the nearest integer resulting in a new vector $V'_i = (v'_{(i,0)}, \ldots, v'_{(i,R-1)})^T$ whose elements are now k-bit positive integers. Note that the threshold value δ should also be adjusted accordingly.

5.2 Computing Similarity Measures

The similarity value between user A and any other user B is computed over the rating vectors of size R. The elements of the user vector $V'_A = (v'_{(A,0)}, \ldots, v'_{(A,R-1)})$ are encrypted individually by using the public key of the user A. Then, the encrypted vector $[V'_A]_{pk_A}$ is sent to the server. The server then sends the encrypted vector to the other users in the system. Any user B who receives the encrypted vector $[V'_A]_{pk_A}$ can compute the encrypted similarity as follows:

$$\begin{aligned}[\text{sim}_{A,B}] &= [\sum_{i=0}^{R-1} v'_{(A,i)} \cdot v'_{(B,i)}] = [v'_{(A,0)} \cdot v'_{(B,0)} + \ldots + v'_{(A,R-1)} \cdot v'_{(B,R-1)}] \\ &= [v'_{(A,0)}]^{v'_{(B,0)}} \cdot [v'_{(A,1)}]^{v'_{(B,1)}} \cdot \ldots \cdot [v'_{(A,R-1)}]^{v'_{(B,R-1)}} = \prod_{i=0}^{R-1} [v'_{(A,i)}]^{v'_{(B,i)}}. \end{aligned} \tag{3}$$

Note that we omit the encryption key pk_A above and in the rest of the paper for the sake of readability. The computed similarity value is then sent back to the server in encrypted form.

5.3 Finding the Most Similar Users

Upon receiving similarity values from users, the server initiates a cryptographic protocol with user A to determine the most similar users whose similarity values are above a public threshold δ. The protocol receives N encrypted similarity values and outputs en encrypted vector $[\Gamma_A] = ([\gamma_{(A,0)}], [\gamma_{(A,1)}], \ldots, [\gamma_{(A,N-1)}])$. The elements of this vector $\gamma_{(A,i)}$ are either an encryption of 1, if the the similarity value between user A and user i is above the threshold δ, or an encryption of 0, otherwise. The details of this protocol can be found in Section 6.

5.4 Generating Recommendations

After obtaining the vector $[\Gamma_A]$, the server can generate the recommendation for user A. For this purpose, the server sends $[\gamma_{(A,i)}]$ to the i^{th} user in the system. User i, referred as user B, can raise $[\gamma_{(A,B)}]$ to the power of each rating he has left in his ratings vector to obtain another encrypted vector $[\Phi_{(A,B)}] = ([\phi_{(A,R)}], [\phi_{(A,R+1)}], \ldots, [\phi_{(A,M-1)}])$ where $\phi_{(A,j)} = [\gamma_{(A,B)} \cdot v'_{(B,j)}] = [\gamma_{(A,B)}]^{v'_{(B,j)}}$ for $j = R$ to $M-1$. Notice that user B does not know the content of $\gamma_{(A,B)}$. The resulting vector $[\Phi_{(A,B)}]$ is either the encrypted rating vector of user B or a vector of encrypted 0's. Vector $[\Phi_{(A,B)}]$ then is sent to the server to be accumulated with other vectors from every user.

The above procedure can be improved in order to minimize the computational and communication cost. Instead of raising $[\gamma_{(A,B)}]$ to the power of each rating, the ratings can be represented in a compact form and then used as an exponent:

$$v'_{(B,R)}|v'_{(B,R+1)}|\ldots|v'_{(B,M-1)}, \tag{4}$$

where $|$ represents the concatenation operation. Assuming that each $v'_{(B,j)}$ is k-bits and N of such vectors are to be accumulated by the server, where N is the number of users participating in the protocol, each compartment should have a bit size of $k + \log(N)$. Thus, packing is achieved by the following formula:

$$v''_B = \sum_{j=0}^{M-R} 2^{j(k+\log(N))} \cdot v'_{(B,j+R)}. \tag{5}$$

By packing values, the communication cost reduces significantly as we obtain a packed value rather than a vector of encrypted vectors. Packing also reduces the number of exponentiations which is a costly operation in the encrypted domain, introducing a gain in computation. However, depending on the message space of the encryption scheme, n, and the number of ratings, $M-R$, it may not be possible to pack all values in one encryption. The number of values that can fit into one encryption is $T = n/(k+\log(N))$. Therefore, we may need $S = \lceil M - R/T \rceil$ encryptions.

Once user B packs his ratings to obtain v''_B, he can compute $[\Phi_{(A,B)}]$ as follows:

$$\left[\Phi_{(A,B)}\right] = [\gamma_{(A,B)}]^{v''_B} = \begin{cases} [v''_B] & \text{if } \gamma_{(A,B)} = 1 \\ [0] & \text{if } \gamma_{(A,B)} = 0, \end{cases} \tag{6}$$

and sends $[\Phi_{(A,B)}]$ to the server. Upon receiving $[\Phi_{(A,i)}]$ values from all users, the server accumulates them:

$$[\Phi_A] = \prod_{i=0}^{N} [\Phi_{(A,i)}] = [\sum_{i=0}^{N} \Phi_{(A,i)}]. \tag{7}$$

Notice that the result will be equal to the sum of ratings of the users who have similarity values above threshold δ. The server also accumulates the $[\gamma_{(A,i)}]$ values to obtain the number of users above the threshold:

$$[L] = \prod_{i=0}^{N} [\gamma_{(A,i)}] = [\sum_{i=0}^{N} \gamma_{(A,i)}]. \tag{8}$$

These two values, $[\Phi_A]$ and $[L]$ are then sent to user A. After decrypting, user A decomposes Φ_A and divides each extracted value by L, obtaining the average ratings of L users. This concludes our protocol.

An important observation at this point is the value of L. If $L = 0$, the user can notify the server to repeat the second step of the protocol with a new threshold. If $L = 1$, the user obtains exactly the same ratings vector of some user but he does not have the identity of that particular user.

6 Cryptographic Protocol for Finding Similar Users

Finding similar users is based on comparing the similarity value between user A and B, $\text{sim}_{A,B}$, to a public threshold δ. As the similarity value is privacy sensitive and should be kept secret both from the server and the user, we compare it in the encrypted domain. For this purpose, we use a comparison protocol that has been introduced in [8]. The cryptographic protocol in [8] takes two encrypted values, $[a]$ and $[b]$, and outputs the result λ again in the encrypted form: if $a > b$ $[\lambda = 1]$, and $[\lambda = 0]$ otherwise. For the completeness of the paper, we give a brief description of the protocol. More explanation and implementation details on the comparison protocol can be found in [8].

Given the similarity value $\text{sim}_{A,M}$ and public threshold δ, both of which are ℓ bits, the most significant bit of the value $z = 2^\ell + \text{sim}_{A,B} - \delta$ is the outcome of the comparison. However, we need to obtain the most significant bit of z in the encrypted domain. While the encrypted value $[z]$ can be computed by the server, the most significant bit of $[z]$ requires running a protocol between the server and user A who has the decryption key. Note that the similarity value cannot be trusted to the user as it leaks information about other users in the system. Therefore, the server adds a random value r to z: $[c] = [z+r]$ and sends it to user A who then decrypts it. Notice that the most significant bit now can be computed as

$$\left[\gamma_{(A,i)}\right] = [2^{-\ell}(c \mod 2^\ell - r \mod 2^\ell) + \alpha \cdot 2^\ell], \tag{9}$$

where the last term is necessary depending on the relation between c and r. The variable α is a single bit representing whether $c > r$ or not. At this point, we convert the problem of comparing $[\text{sim}_{A,i}]$ and δ to the problem of comparing c and r which are owned by the user and the server respectively.

Comparing c and r requires another cryptographic protocol in which the server and user A evaluate the following formula for each of ℓ bits:

$$[e_i] = [1 - c_i + r_i + 3 \sum_{j=i+1}^{\ell-1} c_j \oplus r_j], \tag{10}$$

where c_i and r_i are the i^{th} bits of c and r, respectively. The value of e_i can be 0 if and only if $c > r$, when $c_i = 0$, $r_i = 1$ and the upper part of c and r are the same. After these computations, the server sends the randomized and shuffled $[e_i]$ values to the user A. User A decrypts them and checks whether there is a zero among the values e_i. Existence of a 0 value indicates that $r > c$. However, this leaks information about the comparison of $\text{sim}_{A,B}$ and δ thus, the server randomizes the direction of the comparison by replacing $1 - c_i + r_i$ in Eq. 10 with $-1 - c_i + r_i$ at random. User A then returns $[\alpha]$ which is either $[1]$ or $[0]$ depending on the existence of a 0 among the e_i values. The server can correct the direction of the comparison and obtain the $[\gamma_{(A,i)}]$ by replacing α in Eq. 9.

By using this comparison protocol, each similarity value is compared to threshold δ simultaneously. The outcomes of the comparisons, $[\Gamma_A] = ([\gamma_{(A,0)}], [\gamma_{(A,1)}], \ldots, [\gamma_{(A,N-1)}])$, are then used in the subsequent steps.

7 Complexity Analysis

The performance of our protocol is mainly determined by the interaction among the server, and user A, who asks for recommendation, and other users in the system. In our construction, the server participates in the computation and relays messages

Table 1: Computational complexity.

	Server		User A		User B	
	Paillier	DGK	Paillier	DGK	Paillier	DGK
Encryption	$\mathcal{O}(N)$	$\mathcal{O}(N\ell)$	$\mathcal{O}(R)$	$\mathcal{O}(\ell)$	-	-
Decryption	-	-	$\mathcal{O}(1)$	$\mathcal{O}(\ell)$	-	-
Multiplication	$\mathcal{O}(NS)$	$\mathcal{O}(N\ell^2)$	-	-	$\mathcal{O}(R)$	-
Exponentiation	-	$\mathcal{O}(N\ell)$	-	-	$\mathcal{O}(R+S)$	-

among users. User A, on the other hand, only participates in the protocol in two stages: 1) when he asks for a recommendation and uploads his encrypted data and 2) when he receives the encrypted recommendation. Other users help the server with the recommendation generation.

Round Complexity. Our protocol consists of 5 rounds. The data transfer from users to the server in the initialization stage is 0.5 round. To determine the similar users and generating the recommendation, the server needs 4 rounds of interaction. Notice that during the comparison protocol to obtain $[\Gamma_A]$, all encrypted values are compared to a public value δ, and all comparisons can be done in parallel. In the last stage, the server sends the recommendation to user A which requires another 0.5 round. This gives $\mathcal{O}(1)$ rounds.

Communication Complexity. The amount of data transferred during the protocol is primarily influenced by the size of the encrypted data. For user A, the amount of encrypted data to be transferred is $\mathcal{O}(R + N\ell)$. The server, on the other hand, has to receive and send $\mathcal{O}(N(R + S + \ell))$ encrypted data which is heavily influenced by the data transmission during the comparison of N similarity values. Other users in the system need to receive and send data in the order of $\mathcal{O}(R + S)$.

Computational Complexity. The computational complexity depends on the cost of operations in the encrypted domain and can be categorized into four classes: encryptions, decryptions, multiplications and exponentiations. In Table 1, we provide the average numbers for each operation in the Paillier and the DGK cryptosystems. One exception is for the decryption operation, which is actually a *zero-check* which is a fast and less expensive operation compared to original decryption in DGK cryptosystem.

8 Conclusion

In this paper we proposed a cryptographic approach for generating recommendations to the users within online applications. The proposed method is constructed by homomorphic encryption schemes and MPC techniques. As shown in the complexity analysis, the overhead introduced by working in the encrypted domain is reduced significantly by packing data and using the DGK cryptosystem. Unfortunately, we do not have the chance of comparing our result with previously proposed systems due to space problems. However, we conclude that our proposal is based on a realistic scenario and the required technology is not overly demanding compared to the cryptographic tools like thresholding schemes that other approaches are using [4]. Compared to randomization techniques [11, 12], our proposal is provably secure and does not rely on the number of users in the system.

References

[1] Internet usage statistics. `http://www.internetworldstats.com/stats.htm`, 2009.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.

[3] N. Ahituv, Y. Lapid, and S. Neumann. Processing encrypted data. *Commun. ACM*, 30(9):777–780, 1987.

[4] J. F. Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, pages 45–57, 2002.

[5] J. F. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR*, pages 238–245, New York, NY, USA, 2002. ACM Press.

[6] I. Damgård, M. Geisler, and M. Krøigaard. Efficient and Secure Comparison for On-Line Auctions. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *Australasian Conference on Information Security and Privacy — ACSIP 2007*, volume 4586 of *LNCS*, pages 416–430. Springer, July 2-4, 2007.

[7] I. Damgård and M. Jurik. A Generalization, a Simplification and some Applications of Paillier's Probabilistic Public-Key System. Technical report, Department of Computer Science, University of Aarhus, 2000.

[8] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, R. L. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Proceedings of the Privacy Enhancing Technologies Symposium*, pages 235–253, Seattle, USA, 2009.

[9] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *ACM Symposium on Theory of Computing — STOC '87*, pages 218–229. ACM, May 25-27, 1987.

[10] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *LNCS*, pages 223–238. Springer, May 2-6, 1999.

[11] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *ICDM*, pages 625–628, 2003.

[12] H. Polat and W. Du. SVD-based collaborative filtering with privacy. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 791–795, New York, NY, USA, 2005. ACM Press.

[13] Shopzilla, Inc. Privacy policy, 2009. `http://www.bizrate.com/content/privacy.html`.

[14] A. C.-C. Yao. Protocols for Secure Computations (Extended Abstract). In *Annual Symposium on Foundations of Computer Science — FOCS '82*, pages 160–164. IEEE, November 3-5, 1982.

[15] S. Zhang, J. Ford, and F. Makedon. Deriving private information from randomly perturbed ratings. In *Proceedings of the Sixth SIAM International Conference on Data Mining*, pages 59–69, 2006.