EFFICIENTLY COMPUTING PRIVATE RECOMMENDATIONS

Z. Erkin, M. Beye, T. Veugen and R. L. Lagendijk*

Information Security and Privacy Lab
Mediamatics Department, Delft University of Technology
2628 CD, Delft, The Netherlands

ABSTRACT

Online recommender systems enable personalized service to users. The underlying collaborative filtering techniques operate on privacy sensitive user data, which could be misused by the service provider. To protect user privacy, we propose to encrypt the data and generate recommendations by processing them under encryption. Thus, the service provider observes neither user preferences nor recommendations. The proposed method uses homomorphic encryption and secure multi-party computation (MPC) techniques, which introduce a significant overhead in computational complexity. We minimize the introduced overhead by packing data and using cryptographic protocols particularly developed for this purpose. The proposed cryptographic protocol is implemented to test its correctness and performance.

***Index Terms*—** Recommender systems, 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. As a result, online applications have become very popular 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 tailored to 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 way personal user information is gathered. This information can be provided by the user himself as in profiles, or the service provider can observe users' actions like click logs. On one hand, more user information helps the system to improve the accuracy of the recommendations. On the other hand, the personal information on the users creates a severe privacy risk since there is no solid guarantee for the service provider not to misuse the users' 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].

*TNO Information and Communication Technology P.O. Box 5050, 2600 GB Delft, The Netherlands

To address the privacy considerations in recommender systems, in [3], Canny proposes a system where the private user data is encrypted and recommendations are generated by applying an iterative procedure based on the 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. This iterative algorithm 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 is assumed to be online and honest. The output of each iteration, which is the characterization matrix, is available in the clear. In [4], Canny proposes a method to protect the privacy of users based on a probabilistic factor analysis model by using a similar approach as in [3].

While Canny works with encrypted user data, Polat and Du suggest to protect the privacy of users by using randomization techniques [10, 11]. In their papers, they blind the user ratings with random data assuming that during the aggregation 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 users need to be present 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, which poses a privacy threat to the users. Finally, the randomization techniques are believed to be highly insecure [14].

In [5], Erkin et al. present a cryptographic protocol based on homomorphic encryption and MPC techniques. In that paper, the privacy sensitive data of the users of a recommender system are protected by means of encryption and recommendations are generated by processing the encrypted data. Being highly efficient compared to other techniques in terms of computational and communication costs as well as provably secure, [5] still suffers from a computationally expensive comparison protocol which is based on the ideas from [6].

In this paper, we improve the work by Erkin et al. further by introducing two significant changes. First, we introduce the *data packing* where we pack a number of data and then encrypt. This approach considerabley reduces the number of encryption to be transferred and processes. Second, we develop and use a more efficient comparison algorithm based on ideas from [12]. Moreover, to test the correctness and performance of our protocol, we also present experimental results using a dataset of 10,000 users and 1,000 items.

2. 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 $V_A = (v_{(A,0)}, \ldots, v_{(A,M-1)})$ and $V_B = (v_{(B,0)}, \ldots, v_{(B,M-1)})$, respectively, where M is the number of items and $\bar{v}$ represents the average value of the items in vector v.

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

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

In this paper, we assume that the user preference vector V, which consists of ratings from 0 to 5, is split into two parts: the first part consists of R very popular items that are rated densely by the users and is used for the characterization of the user, V^d and the second part contains $M - R$ scarcely rated items, V^s, that the user would like to get recommendations on [2].

3. CRYPTOGRAPHIC PRIMITIVES AND SECURITY MODEL

We use encryption to protect user data against the service provider and other users in the system. A special class of cryptosystems, namely homomorphic cryptosystems, allows us to process the data in the encrypted form. We chose the Paillier cryptosystem [9] 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, we use Goldwasser-Micali (GM) system [8], which is bitwise additively homomorphic, in our protocol to improve the efficiency. Both cryptosystems are semantically secure.

In this paper, 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 [7] with additional overhead.

4. 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 a user, let's say user A who wants to get some recommendations, 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 run a protocol to determine the similarity values that are above a threshold δ. The server - being unaware of the number of 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 to be used as recommendations. Each step of the proposed protocol is detailed in the following sections.

4.1. Key Generation and Preprocessing

Any user in the system who wants to get recommendations generates personal public key pairs for the Paillier and the GM 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:

$$\begin{aligned} \text{sim}_{A,B} &= \sum_{i=0}^{R-1} C_{A,i} \cdot C_{B,i}, \text{ where} \\ C_{X,i} &= \frac{(v_{(X,i)} - \overline{v}_X)}{\sqrt{\sum_{j=0}^{R-1}(v_{(X,j)} - \overline{v}_X)^2}}, \end{aligned} \tag{2}$$

the terms $C_{A,i}$ and $C_{B,i}$ 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 vector $V_i^d = (v_{(i,0)}^d, \ldots, v_{(i,R-1)}^d)$. Note that the threshold value δ should also be adjusted accordingly.

4.2. 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 $V_A^d = (v_{(A,0)}^d, \ldots, v_{(A,R-1)}^d)$ are encrypted individually by using the public key of the user A. Then, the encrypted vector $[V_A^d]_{pk_A}$ 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 $[V_A^d]_{pk_A}$ can compute the encrypted similarity as follows:

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

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.

4.3. 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. An overview of this protocol can be found in Section 5.

4.4. Generating Recommendations

After obtaining the vector $[\Gamma_A]$, the server can generate the recommendations for user A. For this purpose, the server sends $[\gamma_{(A,i)}]$ to the i^{th} user in the system. User i, hereafter referred to as user B, can raise $[\gamma_{(A,B)}]$ to the power of $v^s_{B,j}$ to obtain another encrypted vector $[\Phi_{(A,B)}] = ([\phi_{(A,R)}], [\phi_{(A,R+1)}], \ldots, [\phi_{(A,M-1)}])$ where, for $j = R$ to $M-1$, $\phi_{(A,j)} = [\gamma_{(A,B)} \cdot v^s_{(B,j)}] = [\gamma_{(A,B)}]^{v^s_{(B,j)}}$. 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 $[\Phi_{(A,B)}]$ is then 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 a single exponent:

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

where $|$ represents the concatenation operation. Assuming that each $v^s_{(B,j)}$ is k-bits and L of such vectors are to be accumulated by the server, where L is the number of users with a similarity above the threshold, each compartment should have a bit size of $k + \log(L)$. Note that the value L is unknown in the beginning of the protocol. Therefore, we assume that L is set to a predefined maximum value. Given L, packing is achieved by the following formula:

$$v^P_B = \sum_{j=0}^{M-R} 2^{j(k+\log(L))} \cdot v^s_{(B,j+R)} \ . \tag{5}$$

By packing values, the communication costs are reduced significantly as we obtain one encrypted packed value rather than a vector of encrypted values. Packing also reduces the number of exponentiations which are costly operations in the encrypted domain, introducing a gain in computation. However, depending on the message space of the encryption scheme, n, L 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 $T = n/(k + \log(L))$. Therefore, we need $S = \lceil (M-R)/T \rceil$ encryptions to pack all of the ratings.

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

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

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

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

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

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

$[\Phi_A]$ 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 the L most similar users.

5. 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.

Given the similarity value $\text{sim}_{A,B}$ 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. To obtain the most significant bit of $[z]$, we run a cryptographic protocol between the server and user A who has the decryption key. In this protocol, the server hides the value z from user A by adding a random value r: $[c] = [z + r]$ and then he sends it to user A who decrypts it. When user A sends $[z \div 2^\ell]$ back to the server, the server is able to compute the most significant bit of z, which is the result of the comparison between $\text{sim}_{A,B}$ and δ, as:

$$\begin{aligned} \left[\gamma_{(A,i)}\right] &= [z \div 2^\ell - r \div 2^\ell - t] \\ &= [z \div 2^\ell] \cdot ([r \div 2^\ell] \cdot [t])^{-1} \ , \end{aligned} \tag{9}$$

where the term t is a single bit either 0 or 1 depending on the relation between $c \bmod 2^\ell$ and $r \bmod 2^\ell$. At this point, we convert the problem of comparing $[\text{sim}_{A,i}]$ and δ to the problem of comparing $c \bmod 2^\ell$ and $r \bmod 2^\ell$ which are available in clear to the user and the server, respectively.

For this purpose, we use a comparison protocol that is based on [12]. This protocol takes two private input values, $x = c \bmod 2^\ell$ and $y = r \bmod 2^\ell$, and outputs the result $[t]$ in encrypted form: if $x > y$, $t = 1$, and $t = 0$, otherwise. The comparison result $t = t_\ell$ is computed recursively as follows:

$$t_{i+1} = (1 - (x_i - y_i)^2) t_i + x_i (1 - y_i), \text{for } 0 \le i < \ell \ , \tag{10}$$

where $t_0 = 0$ and x_i and y_i are the i^{th} bits of x and y, respectively. The recursive equation (10) can be computed efficiently by using the GM scheme. Note that this efficient cryptographic protocol has rounds linear in ℓ unlike the constant round protocol in [6].

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.

6. COMPLEXITY ANALYSIS

We implemented our protocol to determine its correctness and performance. For this purpose, we created a synthetic dataset with 10,000 users and 1,000 items, each rated between 0 to 5. The protocol is implemented in C++, using GMP library version 4.2.1 and is tested on a single computer with Intel Xeon 2.33 GHz processor and 16 GB of RAM.

Note that the number of heavily rated items is $R = 20$ out of $M = 1,000$ items, the number of users is $N = 10,000$, the length of the similarity values $\ell = 22$ bits, the threshold $\delta = 256$, , the scaling parameter is $f = 1,000$ which is chosen experimentally to give identical results to non-private protocol, the number of encryptions required (with packing) is $S = 11$ and the number of values that fit into one encryption is $T = 93$. S and T are computed for a key size of 1,024 bits which provides a modest level of security.

Round Complexity. Our protocol consists of $\ell + 4$ rounds of communication. The data transfer from users to the server in the initialization stage, computing the similarity values and sending back the average ratings of all users to user A takes 3 rounds. The

Table 1. Computational complexity.

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

determination of the similar users, which is comparing the similarity values with the threshold δ, takes $\ell + 1$ rounds. Notice that to obtain $[\Gamma_A]$ in the comparison protocol, all encrypted values are compared to a public value δ and thus, all comparisons can be done in parallel. The total round complexity of our protocol is $\mathcal{O}(\ell)$.

Communication Complexity. The amount of data transferred during the protocol is primarily influenced by the size of the encrypted data which is 2,048 bits for the Paillier scheme. The communication overhead for the server, the users A and any other user B is $\mathcal{O}(N(R + S + \ell)), \mathcal{O}(N\ell + R), \mathcal{O}(R + S)$, respectively. As mentioned in Section 4.4, by packing multiple values into one single encryption, we can obtain a reduction by a factor of T, significantly reducing our communication overhead.

For the given values of the parameters above, the server sends and receives a total of 168 MB of data. At the same time, this value for user A is 88 MB and, for other users it is only 8 KB. Note that the values for the server and user A are heavily affected by the comparison protocol.

Computational Complexity. The computational complexity depends strongly on the cost of operations in the encrypted domain, which can be categorized into four classes: encryptions, decryptions, multiplications and exponentiations. In Table 1, we provide the order of each operation in the Paillier cryptosystems.

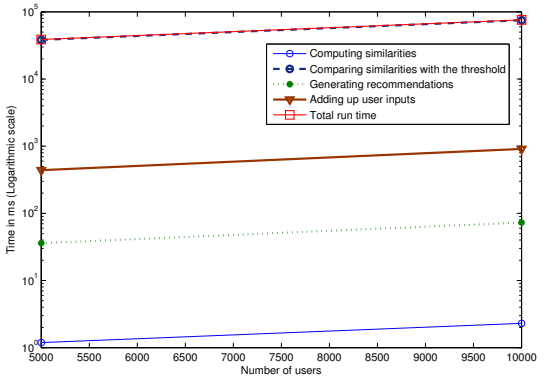

Fig. 1. Run time of the protocol without network latency.

Figure 1 shows the run time of the several stages of the protocol. As seen in the figure, the total run time of 75 seconds heavily depends on the time spent for comparison of similarity values with the threshold. Compared to [5] which takes 415 seconds for the same experimental setting, our protocol outperforms it by a factor of 5.5.

7. CONCLUSION

In this paper, we propose a cryptographic protocol for generating recommendations based on homomorphic encryption and MPC techniques. In particular, we propose to encrypt the privacy sensitive data such as user preferences and similarity values between users and generate recommendation by processing encrypted data. While the homomorphic property allows us to realize linear operations in the encrypted domain, non-linear operations like comparing encrypted values require to realize cryptographic protocols. As we aim at practical privacy-enhanced solutions, efficiency plays a vital role in the success of such protocols.

In order to achieve further efficiency, we introduce a cryptographic protocol based on the ideas of [12]. In addition to that, we exploit data packing to reduce the costs. Both complexity analysis and experimental results in a modest test environment show that our protocol significantly outperforms the previous work by Erkin et al. [5]. Moreover, our protocol is computationally secure and not reliant on the number of users in the system, as opposed to randomization techniques [10, 11]. We conclude that our proposal is based on a realistic scenario and the required technology is not overly demanding compared to cryptographic tools like thresholding schemes, as used in other approaches [3, 4].

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

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

[5] Z. Erkin, M. Beye, T. Veugen, and R. Lagendijk. Privacy enhanced recommender system. In *Thirty-first Symposium on Information Theory in the Benelux*, pages 35–42, Rotterdam, 2010.

[6] 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, 2009.

[7] 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.

[8] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[9] 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.

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

[11] 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.

[12] B. Schoenmakers and P. Tuyls. Practical two-party computation based on the conditional gate. In *Proceedings of Advances in Cryptology*, volume 3329, pages 119–136. Springer-Verlag, 2004.

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

[14] 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.