Generating Private Recommendations in a Social Trust Network

Z. Erkin*, T. Veugen*† and R. L. Lagendijk *
*Information Security and Privacy Lab
Delft University of Technology, 2628 CD, Delft, The Netherlands
†TNO, P.O. Box 5050, 2600 GB, Delft, The Netherlands
Email: {z.erkin, p.m.j.veugen, r.l.lagendijk}@tudelft.nl

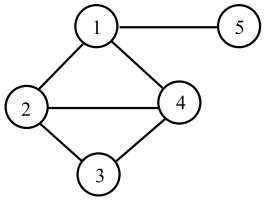
***Abstract*—Recommender systems have become increasingly important in e-commerce as they can guide customers with finding personalized services and products. A variant of recommender systems that generates recommendations from a set of trusted people is recently getting more attention in social networks. However, people are concerned about their privacy as the information revealed in recommender systems, particularly in social networks, can be misused easily. A way to eliminate the privacy risks is to make the privacy-sensitive data inaccessible by means of encryption. While the private data is inaccessible to any outsiders, the same functionality of the system can be achieved by processing the encrypted data. Unfortunately, the efficiency of processing encrypted data constitutes a big challenge. In this paper, we present a privacy-enhanced recommender system in a social trust network, which is designed to be highly efficient. The cryptographic protocol for generating recommendations is based on homomorphic encryption and secure multi-party computation techniques. The additional overhead with regard to computation and communication is minimized by packing data. The experimental results show that the performance of our proposal is promising to be deployed in real world.**

***Index Terms*—Social trust networks, recommender systems, privacy, homomorphic encryption, secure multi-party computation.**

I. Introduction

Recommender systems have become increasingly important for e-commerce as these systems help customers in decision making for a service or a product by providing personalized recommendations. The recommender systems based on collaborative filtering techniques are being used widely in numerous e-commerce applications like Amazon, eBay and youtube. However, in certain online applications, particularly those that also involve social interaction, people are interested in the recommendations of a particular group which consists of people with a certain *trust* relation rather than a number of similar people in the network. These networks, so called Social Trust Networks (STNs), are attracting more attention recently as they can be used for more accurate recommendation generation [12], [16], [23].

In an STN, a network graph or a matrix depicting the trust relation among the users of a recommender system is created. As an example Fig. 1 consists of 5 nodes, each representing a user. The link between nodes represents the trust relation: it either exists or not. Notice that the graph is symmetric, meaning that user $\mathcal{A}$ trusts user $\mathcal{B}$ and vice versa. However, the graph can also be directional. Given a graph or matrix representation of a trust network, recommendations can be easily generated by averaging the ratings of the trusted users.



	1	2	3	4	5
1	0	1	0	1	1
2	1	0	1	1	0
3	0	1	0	1	0
4	1	1	1	0	0
5	1	0	0	0	0

Fig. 1. Trust relation graph and the corresponding matrix.

Recommender systems in general, including STNs, possess high privacy risks for the users [21]. The privacy concerns arise because the recommendations are generated by a central entity, the service provider, and this entity has access to privacy-sensitive information, which can be easily used to identify and track any individual. Moreover, the private information on the user of recommender systems can be re-purposed or transferred to third parties without the knowing of the users. The risk is even higher in the case of STNs as these systems have a social network part integrated where people tend to reveal personal information. For example, any misuse in a medical application where patients are suggested contents or encouraged to make contact with other patients with the same disease as in www.patientslikeme.com will result in considerable damage on privacy.

In this paper, we present an efficient privacy-preserving recommender system for STNs that hides the private information from the service provider. We achieve this goal by providing only encrypted data to the service provider, which cannot access to the private data directly but has cryptographic tools to process the encryptions to generate the recommendations. The cryptographic tools designed for this purpose are based on *Homomorphic Encryption* (HE) and *Multi-Party Computation* (MPC) techniques. The privacy-preserving recommender system requires more resources compared to its plain version due to operating in the encrypted domain. Therefore, we focus on efficiency and present a fine-tuned cryptographic protocol with low storage, bandwidth and computation requirements.

In particular, we propose to add a third party to our protocol and pack data to improve the efficiency. The third party, namely *Privacy Service Provider* (PSP), is an entity with business interest in providing computational resources. Packing data, on the other hand, is a technique in which relevant data are packed in one encryption rather than encrypting them separately to reduce the communication and computational costs. With this construction, we do not rely on the computational resources of the users, which is very suitable for a server-client setting.

To test the correctness and the performance of our proposal, we also conducted experiments. The privacy-preserving version of the recommender system was implemented using C++ and tested on Epinions dataset [15] with 10,000 users and 1,000 items. We evaluate our results regarding data storage, run time and communication overhead. Experiments show that our privacy-preserving recommender system outperforms the previous works, which is encouraging in the direction of deploying privacy-preserving recommender systems in the real world.

II. Related Work

Preserving the privacy of users in recommender systems has been addressed by several researchers in the last decade. The proposed methods range from statistical techniques to distributed profiles and cryptographic techniques. In [18], [19], Polat and Du propose a centralized system to perturb data as a statistical approach. In their proposals, random values with a known distribution are added to the user's ratings such that when the randomized data are aggregated, the randomized parts cancel out and the desired outcome is obtained. As it is mentioned in [18], [19], this type of privacy protection introduces a trade-off between privacy and accuracy. Moreover, it is highly believed that masking user ratings may not protect the user privacy as much as anticipated [25]. In [1], Berkovsky et al. propose an obfuscation scheme for decentralizing the rating profiles among multiple repositories. However, 100% accuracy is achieved only if all private data is disclosed.

In [22] a recommender system based on distributed aggregation of off-line profiles is suggested. In that work, users have two types of profiles, namely off-line and online. While users keep their off-line profile locally, the server stores the online one. These two profiles are regularly synchronized. The proposed method assumes that the users communicate over another media too such as face-to-face, mobile phone or e-mail, which is a strong assumption to realize in large scale deployment of the system. As in [18], [19], [22] also suffers from the trade-off between privacy and accuracy. In [5], an agent based system is introduced for privacy-preserving recommender systems where trusted software and environment are required.

In [3], [4], Canny proposes to use cryptographic tools like HE schemes [9], MPC techniques [24] and zero-knowledge proofs [13] for generating recommendations. Despite the fact that such tools are proven to be secure, the construction of the cryptographic protocol for generating recommendations suffers from several information leakages. For instance in [3], the intermediate value of the algorithm, namely the characterization matrix of the users, is available to the server in clear. Moreover, both works use resource demanding, generic cryptographic tools such as Yao's garbled circuits [24] and threshold decryption [6], [7]. These generic techniques are computationally expensive as garbled circuits require the use of computationally intensive Oblivious Transfer [20] protocols and threshold decryption requires a significant number of users to participate in the decryption procedure. In [8], Erkin et al. address these drawbacks and propose a cryptographic protocol that is custom tailored for generating recommendations. The proposed protocol exploits the signal properties of the ratings and uses dedicated protocols for computing similarity measures and finding the best matches, instead of using generic techniques however, their proposal requires active involvement of users in heavy computations.

III. Building Blocks

To process data under encryption, we exploit the *additive homomorphism* property of the Paillier [17] and ElGamal [11] public key cryptosystems. With additive homomorphism, the product of two encrypted messages, m_1 and m_2, corresponds to a new encrypted message whose decryption yields the sum of these two messages:

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(m_1) \cdot \mathcal{E}_{pk}(m_2)) = m_1 + m_2 \,, \tag{1}$$

where $\mathcal{E}_{pk}(\cdot)$ and $\mathcal{D}_{sk}(\cdot)$ represent the encryption and decryption functions; pk and sk corresponds to the public key and the secret key of the scheme, respectively. As a consequence of the additive homomorphism, any ciphertext $\mathcal{E}_{pk}(m)$ raised to the power of a public value, c, corresponds to the multiplication of m and c in the plain domain:

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)^c) = m \times c \,. \tag{2}$$

In addition to the homomorphism property, the Paillier and the ElGamal cryptosystems are semantically secure, implying that there are different ciphertexts for the same plain message. We use the variant of ElGamal for efficiency reasons since operations on bits are more suitable to realize with ElGamal. However, efficient decryption in ElGamal is not possible and therefore, we use Paillier for the remaining parts of the protocol. Instead of efficient decryption in ElGamal, we can easily check whether the cipher text is an encryption of 0 or not, which is convenient for our purpose.

In the rest of the paper, for the sake of simplicity we denote the Paillier encryption with $[m]$ and the ElGamal encryption with $[\![m]\!]$. In this notation, we also omit the keys since there will be only one key pair for each cryptosystem.

IV. Setting, Privacy and Security Requirement

We present three roles in our setting for the recommender system.

1) **The Service Provider (SP).** The service provider has a business interest in generating recommendations for the users.

2) **The Privacy Service Provider (PSP).** The PSP has a business interest in providing computational resources but it is not trusted with the private data. The PSP generates the Paillier and the ElGamal key pairs and broadcasts the public keys.
3) **Users.** The user creates his own trust list locally and would like to get a recommendations from the service provider. We assume that there are N users and each of them has a unique identifier, ID_i and two private data sets: a) a list of trusted users, $\mathcal{V}_i = (v_{(i,0)}, v_{(i,1)}, \ldots, v_{(i,N-1)})$ where $v_{(i,j)}$ is 1 if user i trusts user j and 0 otherwise and b) ratings for M items, $\mathcal{W}_i = (w_{(i,0)}, \ldots, w_{(i,M-1)})$, where $w_{(i,j)}$ is an integer value in a small range like 0 to 5.

Our goal is to protect the privacy of any user, who requested recommendations from the service provider, by keeping his trust list, his ratings and the generated recommendations secret from the service provider, the PSP and all other users.

We use the semi-honest security model, which assumes that all players follow the protocol steps but are curious at the same time and thus, they keep all messages from previous and current steps to extract more information than they are allowed to have. Moreover, we also assume that the PSP and the service provider do not collude. This assumption can be justified in real life as these entities have business interest and reputation. Of course, the protocol can be adapted to the active attacker model and can be secured against collusion by using the ideas in [14], [2] with overhead.

V. Privacy Enhanced Recommender System

In this section, we present the privacy-preserving recommender system that hides the trust relation among users, the user ratings and the generated recommendations from the service provider, the PSP and the other users in the system.

A. Initialization

Having a trust vector $\mathcal{V}_i$, user i creates a polynomial $\mathcal{P}$ of degree k_i whose roots are the identities of the users whom user i trusts:

$$\mathcal{P}_i(x) = \prod_{j=0, v_{(i,j)}==1}^{N-1} (x - ID_j) = \sum_{j=0}^{k_i} p_{(i,j)} \cdot x^j, \quad (3)$$

where $p_{(i,j)}$'s are the coefficients of the constructed polynomial, k_i is the number of trusted users and $k_i \ll N$. Since the number of trusted users for each user, k_i, may change, we assume that there is an upper limit for all k_i values in a social network and we denote it by k.

After having computed the polynomial, user i encrypts the coefficients, $p_{(i,j)}$, by using the ElGamal public key of the PSP. In addition, he packs his ratings as follows:

$$\mathcal{W}_i = w_{(i,M-1)} || \ldots || w_{(i,1)} || w_{(i,0)}, \quad (4)$$

where $w_{(i,j)}$ is the ℓ bit rating for the item j. Since in the subsequent steps, $\mathcal{W}_i$ values from at most k users will be added up, we need to reserve enough space for each compartment to prevent an overflow. This is achieved as follows,

$$\mathcal{W}_i = \sum_{j=0}^{M-1} 2^{j \cdot (\ell + \lceil \log(k) \rceil)} \cdot w_{(A,j)}. \quad (5)$$

Subsequently, user i encrypts $\mathcal{W}_i$ by using the public Paillier key of the PSP and sends the encrypted coefficients of polynomial $\mathcal{P}_i$, encrypted $\mathcal{W}_i$ and his identity ID_i to the service provider, $\{ID_i, ([\![p_{(i,0)}]\!], [\![p_{(i,1)}]\!], \ldots, [\![p_{(i,k-1)}]\!]), [\mathcal{W}_i]\}$. The service provider stores this tuple from every user in the system.

B. Generating Recommendations

1) Checking the Roots of the Polynomial: Upon a request for recommendation by a user, let's say user $\mathcal{A}$, the service provider evaluates the polynomial $\mathcal{P}_A$ with the identity of the users, namely ID_i, in the encrypted domain as follows:

$$[\![\mathcal{P}_{(A,i)}]\!] = [\![\mathcal{P}_A(ID_i)]\!] = \prod_{j=0}^{k-1} [\![p_{(A,j)}]\!]^{(ID_i)^j} = [\![\sum_{j=0}^{k-1} p_{(A,j)} \cdot (ID_i)^j]\!]. \quad (6)$$

Notice that the value of $\mathcal{P}_{(A,i)}$ is 0 if and only if ID_i is a root of the polynomial, and thus user $\mathcal{A}$ has user i in his trust list, and $\mathcal{P}_{(A,i)}$ will be a random value, otherwise [2]. Of course, the service provider cannot observe the contents of the encryptions since he does not have the decryption key. He can not distinguish between an encryption of zero and a random value either, due to the semantically secure cryptosystem.

The service provider also generates N random values, r_i, of length $\lceil \log_2(W) \rceil + \kappa$ bits, where $\lceil \log_2(\mathcal{W}) \rceil$ is the maximum bit length of $\mathcal{W}_i$ values and κ is a security parameter (usually 40-100 bits), and masks each $\mathcal{W}_i$ with a different r_i as follows:

$$[\mathcal{W}_i^{masked}] = [\mathcal{W}_i] \cdot [r_i] = [\mathcal{W}_i + r_i]. \quad (7)$$

After that, the service provider sends $[\mathcal{W}_i^{masked}]$ and all $[\![\mathcal{P}_{(A,i)}]\!]$ values to the PSP.

2) Computing the Sum of the Ratings of Trusted Users: Having the secret key, the PSP can easily verify whether each $\mathcal{P}_{(A,i)}$ is zero or not by using the zero-check function of the ElGamal cryptosystem. The PSP creates two counters C and T, which are initially set to 0, and N variables, γ_i, for each user and then,

- if $\mathcal{P}_{(A,i)}$ is 0, the PSP sets γ_i to 1, increments C and sets $[T] = [T + \mathcal{W}_i^{masked}] = [T] \cdot [\mathcal{W}_i^{masked}]$,
- if $\mathcal{P}_{(A,i)}$ is not zero, he sets γ_i to 0.

Notice that γ_i is exactly the same as the vector $\mathcal{V}_A$: it's elements are either 1 or 0 depending whether user i is in the trust list of user $\mathcal{A}$ or not. Since neither the PSP nor the service provider has access to $\mathcal{V}_A$, it is necessary to create γ_i values for the subsequent steps.

When every $\mathcal{P}_{(A,i)}$ value is checked, T becomes $\sum_{i=0, v_{(A,i)}==1}^{N-1} \mathcal{W}_i^{masked}$. After then, the PSP sends, $[C]$, $[T]$ and all the $[\gamma_i]$ values to the service provider. Note that

the PSP can also decrypt the $[\mathcal{W}_i^{masked}]$. However, as they are all masked with random r_i values, he cannot obtain any meaningful information on the original $\mathcal{W}_i$.

3) Computing the Recommendations: After receiving $[T]$, $[C]$ and the $[\gamma_i]$, the service provider has to remove the random mask from $[T]$. Since he does not know which values are added up by the PSP, the service provider computes the following:

$$[R] = \prod_{i=0}^{N-1} [\gamma_i]^{r_i} = \left[\sum_{i=0, v_{(A,i)}==1}^{N-1} r_i \right]. \quad (8)$$

Notice that the sum above contains the random values of users who are in the trust list of user $\mathcal{A}$. After having computed the random factor, the service provider removes the mask from T as follows:

$$\left[T^{plain}\right] = [T] \cdot [R]^{-1} = [T - R] = \left[\sum_{i=0, v_{A,i}==1}^{N-1} \mathcal{W}_i \right], \quad (9)$$

where $[R]^{-1}$ is the multiplicative inverse of $[R] \bmod n^2$. Then, the service provider sends $[T^{plain}]$ and $[C]$ to user $\mathcal{A}$. Note that the PSP cannot send $[C]$ or C directly to user $\mathcal{A}$ since he does not know (the identity of) that particular user.

4) Obtaining the Recommendations: In the last step of the protocol, user $\mathcal{A}$ runs a simple protocol to obtain the decrypted values T^{plain} and C. For this purpose, user $\mathcal{A}$ masks $[T^{plain}]$ and $[C]$ by adding random values S_1 and S_2 of size $M \cdot (\ell + \lceil \log_2(k) \rceil) + \kappa$ and $\lceil \log_2(k) \rceil + \kappa$, respectively.

$$\begin{aligned} \left[T^{masked}\right] &= [T^{plain}] \cdot [S_1] = [T^{plain} + S_1], \\ \left[C^{masked}\right] &= [C] \cdot [S_2] = [C + S_2], \end{aligned} \quad (10)$$

and sends $[T^{masked}]$ and $[C^{masked}]$ to the PSP via the service provider. The PSP decrypts and sends back the plain values. User A then removes the masks by subtracting random values S_1 and S_2 to obtain T^{plain} and C. Next user $\mathcal{A}$ splits T^{plain} into M compartments. Finally, the sum of ratings of each item is divided by C to obtain the average ratings as the recommendations.

C. Practical Details

In this section, we clarify the details of the cryptographic protocol that we omitted in the previous section for the sake of simplicity.

Evaluation of the Polynomial $\mathcal{P}_{(A,i)}$. Given that user A computed a polynomial of degree k, the server needs to compute the values of the polynomial $\mathcal{P}_{(A,i)}$ at point ID_i as in Eq. 3. This computation requires $\mathcal{O}(k)$ exponentiations. However, using Horner's rule as in [10], the computation can be done more efficiently. Moreover, notice that the bit length of the value $\mathcal{P}_A(ID_i)$ depends on the bit size of k and the ID values and it can be at most $k \cdot |ID|$ bits. Therefore, during the computation, there can be carry-overs modulo q when $q \leq ID^k$. However, $P_A(ID_i) \bmod q = 0$ implies that $(ID_i - ID_j) \bmod q = 0$ for some j since q is a prime number. In practice q has at least 512 bits and ID is much smaller, say 20 bits. Thus, $(ID_i - ID_j) \bmod q = 0$ is satisfied only when $ID_i = ID_j$. This shows that carry-overs modulo q are not a problem for the evaluation of polynomial P_A at point ID_i.

Data Packing. We also assumed for the sake of simplicity that packing all ratings in one encryption is possible. Given M ratings, each of which is ℓ-bit long, the number of ratings that can fit in one encryption is $\tau = (n - \kappa)/(\ell + \log_2(k))$. Therefore, the number of encryptions that a user needs to send to the server is not 1 but actually $\beta = \lceil M/\tau \rceil$.

VI. Complexity and Performance Analyses

In this section, we present the complexity analysis of our protocol and show experimental results. For testing purposes, we implemented our protocol in C++ using GMP Library version 4.2.1 on a single computer with Intel Xeon 2.33 GHz processor and 16 GB RAM. We use the Epinions data set (www.epinions.com) which has 49,290 users who rated a total of 139,738 different items in the range min=1 to max=5. The dataset comes with two files, namely a trust matrix and users' ratings, both of which are highly sparse, more than 99.9% [15]. As our aim is to show the performance of the recommendation generation in the encrypted domain, we choose a smaller subset of this dataset with 10,000 users and 1,000 items.

In our experiments, we have used 1,024 and 512-bit keys for the Paillier and the ElGamal cryptosystems, respectively. We set our security parameter κ to 40 bits. The bit length of ratings is $\ell = 3$ bits. We repeated our experiments 10 times with 1,000, 5,000 and 10,000 users, and 1,000 items in each case and the corresponding number of trusted users, k, is 409, 1136 and 1368, and the number of encryptions, β, is 82, 70, 70, respectively.

A. Round Complexity

The round complexity of the cryptographic protocol for generating recommendations is constant and 3 rounds: 1 round of communication between the service provider and the user, 1 round between the service provider and the PSP and 1 round between the PSP and the user.

B. Computational Complexity

Computational complexity is determined by the number of operations in the encrypted domain. These operations are encryption, decryption, multiplication and exponentiation. The overall computational complexity is summarized in Table I where N is the number of users in the system, k is the maximum number of trusted users, n is the message space of the Paillier cryptosystem and β is the number of encryption required to pack all of the ratings.

In our experiments, we ignore the network latency and assume that each user can encrypt the coefficients of the polynomials and the packed recommendations simultaneously. In Fig. 2 we present the total run time of the protocol for two different cases. In the first case, the protocol takes place

TABLE I
COMPUTATIONAL AND COMMUNICATION COMPLEXITY.

		service provider		PSP		User	
		Paillier	ElGamal	Paillier	ElGamal	Paillier	ElGamal
Encryption	$(c = E_{pk}(m))$	$\mathcal{O}(N\beta)$	-	$\mathcal{O}(N)$	-	$\mathcal{O}(\beta)$	$\mathcal{O}(k)$
Decryption	$(m = D_{sk}(c))$	-	-	$\mathcal{O}(1)$	-	-	-
Zero-check	$(E_{pk}(m) = E_{pk}(0))$	-	-	-	$\mathcal{O}(N)$	-	-
Multiplication	$(E_{pk}(m_1) \cdot E_{pk}(m_2))$	$\mathcal{O}(N\beta)$	$\mathcal{O}(Nk)$	$\mathcal{O}(k\beta)$	-	$\mathcal{O}(1)$	-
Exponentiation	$(E_{pk}(m)^z)$	$\mathcal{O}(n\beta)$	$\mathcal{O}(Nk)$	-	-	-	-
Data		$\mathcal{O}(N\beta)$	$\mathcal{O}(Nk)$	$\mathcal{O}(N\beta)$	$\mathcal{O}(N)$	$\mathcal{O}(\beta)$	$\mathcal{O}(k)$

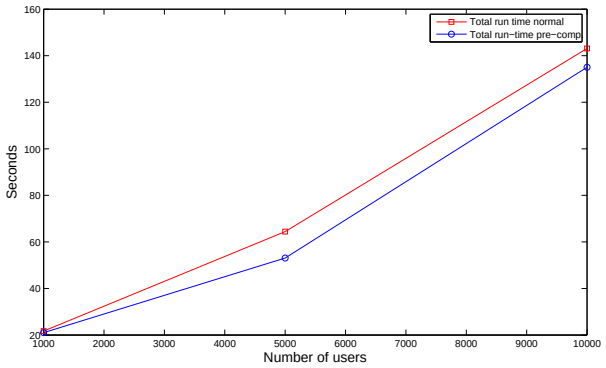


Fig. 2. Average run time of the protocol with and without pre-computation.

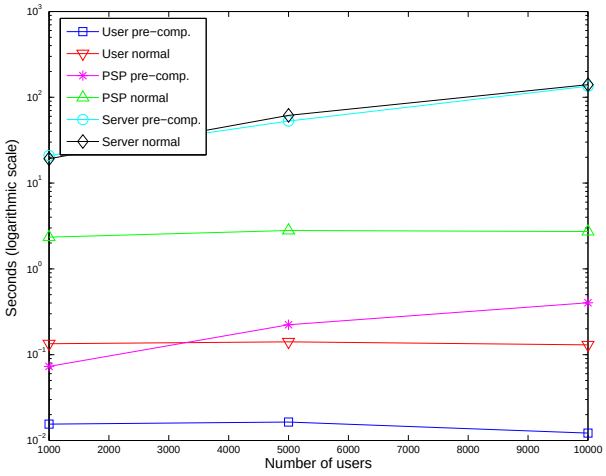


Fig. 3. Average run time for a user, the PSP and the service provider.

normally, while in the second case, we introduce a pre-computation step where we generate the random numbers required for the encryptions.

As can be seen in Fig. 2 the pre-processing step seems not to introduce any significant gain in total run time efficiency. This is because the total run time is heavily dependent on the operations by the service provider. As Fig. 4 shows the computation of the blinding factor R in Eq. 8 constitutes almost 97% of the total run time and this operation does not involve pre-computed values. On the other hand, Fig. 3 shows that the pre-computation introduces a speed-up by a factor of 8

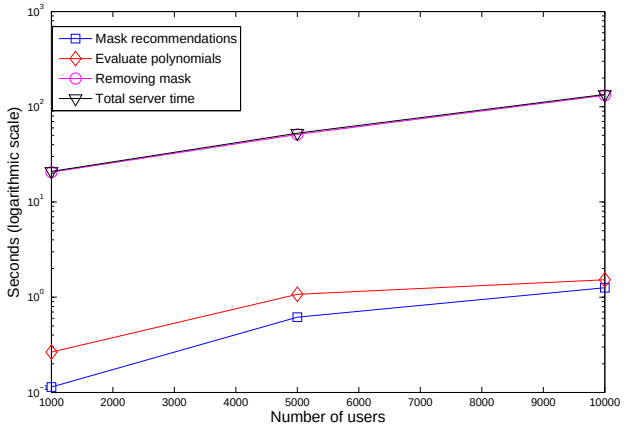


Fig. 4. Average run time for the service provider operations.

to 10 for the users and the PSP who encrypt their polynomial coefficients, packed recommendations, and γ_i's, respectively. Note that while user's actions are not dependent on the number of users N, the PSP encrypts seperate γ_i values for each user.

Experiments on a single machine show that for 10,000 users and 1,000 items, it takes only 135 seconds to generate recommendations. Here, it is important to note that 131 seconds of this total run time is consumed for the computation of the blinding factor R which can be parallelized for further efficiency. The run time of 135 seconds outperforms the previous work in [8] which takes 415 seconds for the same experimental setting. Note that [8] first computes Pearson correlation among users to find out the most similar users and then generates the recommendations. Compared to [3], our protocol outperforms as well. As Canny shows in his paper, the total communication and computational complexity per user is $\mathcal{O}(t \cdot m \log(N))$ where the approximation matrix in a lower dimension is of size $t \times m$, m being the number of items and N is the number of users. Note that this complexity is only for one iteration of that protocol. As Canny suggests, the protocol converges in 40-60 iterations. The run time estimation of Canny's works is about 15 hours on a 500 MHz machine [3].

C. Communication Complexity

The communication complexity depends on the amount of bits transmitted during the protocol. This amount is heavily

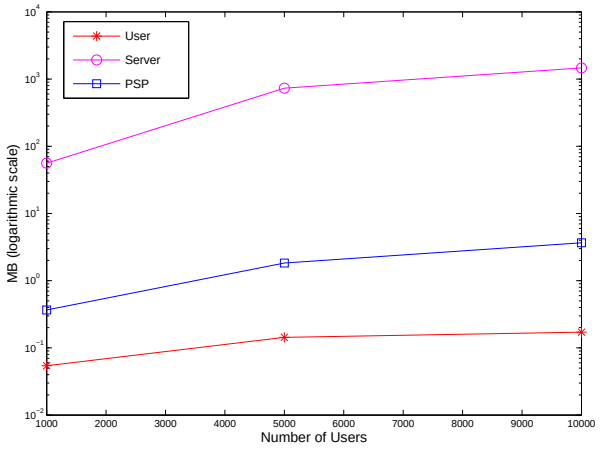


Fig. 5. The amount of data transferred.

influenced by the encryption scheme. For a modest key size of 1024 bits, the Paillier cryptosystem has a ciphertext size of 2048 bits. The ElGamal cryptosystem, on the other hand, uses a key size of 512 bits. The number of encryptions that each player sends and receives is given in Table I.

The overall data transmission for different number of users is given in Fig. 5. While a single user and the PSP have considerably less bandwidth requirement (170 KB and 3.6 MB, respectively), the service provider receives and sends 1.4 TB. This is mainly due to the first step of the protocol where N users send their $k+\beta$ encryptions. Considering that uploading these encryptions will spread in time and each user will only re-upload his encryptions if his trust vector or ratings change, we expect less bandwidth requirement for the service provider in reality. Compared to [3], the communicational complexity of our protocol is much better. With 10,000 users, Canny estimates a communication load of 4GB or 600 MB *per user* using either ElGamal cryptosystem or Elliptic Curve Cryptography, respectively.

D. Data Storage

The service provider stores $k+1$ encrypted polynomial coefficients in ElGamal cryptosystem and β Paillier encryptions for the packed recommendations for each user. The required space for storing these encryptions is 70 MB, 780 MB and 1850 MB for 1,000, 5,000 and 10,000 users, respectively. Note that cipher text size is 2,048 bits for the Paillier scheme and 1,024 bits for ElGamal.

REFERENCES

[1] S. Berkovsky, Y. Eytani, T. Kuflik, and F. Ricci. Enhancing privacy and preserving accuracy of a distributed collaborative filtering. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 9–16, New York, NY, USA, 2007. ACM.

[2] J. Camenisch and G. M. Zaverucha. Private intersection of certified sets. In *Financial Cryptography*, pages 108–127, 2009.

[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] R. Cissée and S. Albayrak. An agent-based approach for privacy-preserving recommender systems. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[6] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 280–299, London, UK, 2001. Springer-Verlag.

[7] I. Damgard, M. Jurik, and J. Nielsen. A generalization of paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6):371–385, 2010.

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

[9] C. Fontaine and F. Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, 2007, 2007.

[10] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, pages 1–19, 2004.

[11] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[12] J. Golbeck. Combining provenance with trust in social networks for semantic web content filtering. In *IPAW*, pages 101–108, 2006.

[13] O. Goldreich. *Foundations of Cryptography. Basic Applications*, volume 2. Cambridge University Press, first edition, May 2004. ISBN 0-521-83084-2.

[14] 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.

[15] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *Proceedings of Federated International Conference On The Move to Meaningful Internet: CoopIS, DOA, ODBASE*, pages 492–508, 2004.

[16] P. Massa and B. Bhattacharjee. Using trust in recommender systems: An experimental analysis. In *Proceedings of iTrust2004 International Conference*, pages 221–235, 2004.

[17] 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.

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

[19] 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.

[20] M. O. Rabin. How to exchange secrets with oblivious transfer, 1981. Harvard University Technical Report 81.

[21] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–62, 2001.

[22] R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. Hubaux. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 157–164, New York, NY, USA, 2009. ACM.

[23] F. E. Walter, S. Battiston, and F. Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16(1):57–74, 2008.

[24] 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.

[25] 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.