Generating Private Recommendations Efficiently Using Homomorphic Encryption and Data Packing

Zekeriya Erkin, *Member, IEEE*, Thijs Veugen, Tomas Toft, and Reginald L. Lagendijk, *Fellow, IEEE*

***Abstract*—Recommender systems have become an important tool for personalization of online services. Generating recommendations in online services depends on privacy-sensitive data collected from the users. Traditional data protection mechanisms focus on access control and secure transmission, which provide security only against malicious third parties, but not the service provider. This creates a serious privacy risk for the users. In this paper, we aim to protect the private data against the service provider while preserving the functionality of the system. We propose encrypting private data and processing them under encryption to generate recommendations. By introducing a semitrusted third party and using data packing, we construct a highly efficient system that does not require the active participation of the user. We also present a comparison protocol, which is the first one to the best of our knowledge, that compares multiple values that are packed in one encryption. Conducted experiments show that this work opens a door to generate private recommendations in a privacy-preserving manner.**

***Index Terms*—Homomorphic encryption, privacy, recommender systems, secure multiparty computation.**

I. Introduction

MILLIONS of people are using online services for various daily activities [1], many of which require sharing personal information with the service provider. Consider the following online services:

Social Networks: People use social networks to get in touch with other people, and create and share content that includes personal information, images, and videos. The service providers have access to the content provided by their users and have the right to process collected data and distribute them to third parties. A very common service provided in social networks is to generate recommendations for finding new friends, groups, and events using collaborative filtering techniques [2]. The data required for the collaborative filtering algorithm is collected from various resources including users' profiles and behaviors.

Online Shopping: Online shopping services increase the likelihood of a purchase by providing personalized suggestions to their customers. To find services and products suitable to a particular customer, the service provider processes collected user data like user preferences and clicklogs.

IP-TV: A set-top box with high storage capacity and processing power takes its place almost in every household. The service providers use smart applications to monitor people's actions to get (statistical) information on people's watching habits, their likes and dislikes. Based on the information collected from the users, the service provider recommends personalized digital content like TV programs, movies, and products that a particular user may find interesting.

In all of the above services and in many others, recommender systems based on collaborative filtering techniques that collect and process personal user data constitute an essential part of the service. On one hand, people benefit from online services. On the other hand, direct access to private data by the service provider has potential privacy risks for the users since the data can be processed for other purposes, transferred to third parties without user consent, or even stolen [3]. Recent studies show that the privacy considerations in online services seem to be one of the most important factors that threaten the healthy growth of e-business [4]. Therefore, it is important to protect the privacy of the users of online services for the benefit of both individuals and business.

A. Previous Work

The need for privacy protection for online services, particularly those using collaborative filtering techniques, triggered research efforts in the past years. Among many different approaches, two main directions, which are based on data perturbation [5] and cryptography [6], have been investigated primarily in literature. Polat and Du in [7] and [8] suggest hiding the personal data statistically, which has been proven to be an insecure approach [9]. Shokri *et al.* present a recommender system that is built on distributed aggregation of user profiles, which suffers from the trade-off between privacy and accuracy [10]. McSherry and Mironov proposed a method using differential privacy, which has a similar trade-off between accuracy and privacy [11]. Cissée and Albayrak present an agent system where trusted software and secure environment are required [12]. Atallah *et al.* proposed privacy-preserving collaborative forecasting and benchmarking to increase the reliability of local

Manuscript received September 01, 2011; revised March 05, 2012; accepted March 06, 2012. Date of publication March 13, 2012; date of current version May 08, 2012. This work was supported by the Kindred Spirits Project, sponsored by the STWs Sentinels program in The Netherlands. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Alessandro Piva.

Z. Erkin and R. L. Lagendijk are with the Information Security and Privacy Laboratory, Department of Intelligent Systems, Delft University of Technology, 2628 CD, Delft, The Netherlands (e-mail: z.erkin@tudelft.nl; r.l.lagendijk@tudelft.nl).

T. Veugen is with Information Security and Privacy Laboratory, Department of Intelligent Systems, Delft University of Technology, 2628 CD, Delft, The Netherlands, and also with TNO, 2600 GB, Delft, The Netherlands (e-mail: p.j.m.veugen@tudelft.nl).

T. Toft is with the Computer Science Department, Aarhus University, Aarhus DK-8200, Denmark (e-mail: ttoft@cs.au.dk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2012.2190726

forecasts and data correlations using cryptographic techniques [13]. Canny also presented cryptographic protocols to generate recommendations, which suffer from a heavy computational and communication overhead [14], [15]. Erkin *et al.* also propose protocols based on cryptographic techniques, which are computationally more efficient than their counterparts in [16] and [17]. However, in their proposals the users are actively involved in the computations, which makes the overall construction more vulnerable to time-outs and latencies in the users' connections. Moreover, the computations that a single user has to perform involve encryptions and decryptions in the order of thousands, which makes the system expensive to run for the users.

B. Our Contribution

In this work, we consider a scenario where users of an online service receive personalized recommendations, which are generated using collaborative filtering techniques [2]. In this scenario, we aim at protecting the privacy of the users against the service provider by means of encrypting the private data, that is users' ratings, and to generate recommendations in the encrypted domain by running cryptographic protocols, which is an approach similar to [14]–[17]. The output of the cryptographic protocol, as well as the intermediate values in the algorithm, is also private and not accessible to the service provider. It is important to note that while generating recommendations by processing encrypted data is possible, the difficulty lies in realizing *efficient* privacy-preserving protocols. Our goal is to provide a more efficient privacy-preserving recommender system by improving the state-of-the-art further.

We achieve our goal of having an efficient recommender system as follows. We eliminate the need for active participation of users in the computations by introducing a semitrusted third party, namely the privacy service provider (PSP), who is trusted to perform the assigned tasks correctly, but is not allowed to observe private data. With this construction, the users, who use an applet or a browser plug-in for the service, upload their encrypted data to the service provider and the recommendations are generated by running a cryptographic protocol between the service provider and the PSP, without interacting with the users. As a consequence of this construction, the cryptographic protocol between the service provider and the PSP to generate recommendations has to work on encrypted data only, which makes it impossible to benefit from homomorphic operations as in [16] and [17]. Therefore, we employ alternative ways of processing encrypted data like secure multiplication and decryption protocols, which introduce a significant amount of additional computational overhead to the system. We reduce the computational and communication cost significantly by data packing, a construction similar to [18] and [19], in which several numerical values are packed in a compact way prior to encryption. We also present a cryptographic protocol that compares encrypted and packed data, which is, to the best of our knowledge, the only algorithm existing in the literature.

We analyze the performance of our proposal by conducting experiments on a dataset with 10 000 people and their ratings for 1000 items. We compare two versions of our proposal, with and without data packing, in terms of bandwidth and runtime.

C. Organization of the Paper

The rest of the paper is organized as follows. Section II describes the preliminaries and notation. Section III presents the privacy-preserving version of the recommender system without using data packing. Section IV explains how to use data packing to improve the privacy-preserving protocol further in terms of efficiency. Section V gives a security analysis of the proposed protocol, and Section VI presents the complexity analysis and experimental results. Section VII compares experimental results with existing work, and Section VIII concludes the paper with a discussion.

II. Preliminaries

We summarize the notation and security model in Section II-A, homomorphic encryption in Section II-B, and give a summary on the collaborative filtering technique that we build our protocol on in Section II-C.

A. Notation and Security Model

We denote a vector by capital, bold letters and its elements with small letters. Throughout the paper, the index i denotes the user and j denotes the index of the vector element. Thus, $v_{(i,j)}$ is the jth element of the ith user's vector, $\mathbf{V}_i$.

In this paper, we consider only the semihonest model where participants are assumed to be honest-but-curious [20]. Therefore, the privacy-preserving protocol we present in this paper can be proven secure in this model by using existing proofs for the basic building blocks, which it is composed of. In addition, in this model we assume that none of the involved parties, namely the service provider, the PSP, and the users, will collude. In particular, we assume that business driven parties, the service provider, and the PSP, do not collude since such an action will harm their business reputation. We refer readers to [21] for a discussion on the practicality of using third parties.

B. Homomorphic Encryption

A number of public-key cryptosystems are *additively homomorphic*, meaning that there exists an operation on cipher texts such that the result of that operation corresponds to a new cipher text whose decryption yields the sum of the messages. Given messages m_1 and m_2, the homomorphic property of an additively cryptosystem with multiplication being the homomorphic operation in the encrypted domain, works as follows:

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

where pk and sk are the public and secret key, respectively. As a consequence of the additive homomorphism, any cipher text $\mathcal{E}_{pk}(m)$ raised to the power a results in a new encryption of $m \cdot a$ as

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

In this paper, we use two additively homomorphic cryptosystems: Paillier [22] and Damgård, Geisler, and Krøigaard (DGK) [23], [24]. We use the Paillier cryptosystem to encrypt the privacy-sensitive data of the users. DGK is used only for a subprotocol as described in Appendix C. DGK is more efficient in

terms of encryption and decryption compared to Paillier due to its smaller message space of a few bits.

The Paillier scheme. The encryption of a message, $m \in \mathbb{Z}_n$, by using the Paillier scheme is defined as

$$\mathcal{E}_{pk}(m, r) = g^m \cdot r^n \bmod n^2 \tag{3}$$

where n is a product of two large prime numbers p, q, g generates a subgroup of order n, and r is a random number in $\mathbb{Z}_n^*$. The public key is (n, g) and the private key is (p, q). Note that the message space is $\mathbb{Z}_n$ and the cipher text space is $\mathbb{Z}_{n^2}^*$. For the decryption operation, we refer readers to [22].

The homomorphic property of the Paillier cryptosystem can be easily verified as shown below:

$$\begin{aligned}\mathcal{E}_{pk}(m_1, r_1) \times \mathcal{E}_{pk}(m_2, r_2) &= \mathcal{E}_{pk}(m_1 + m_2, r_1 \cdot r_2) \\ &= g^{m_1} \cdot r_1^n \times g^{m_2} \cdot r_2^n \bmod n^2 \\ &= g^{m_1+m_2} \cdot (r_1 \cdot r_2)^n \bmod n^2.\end{aligned} \tag{4}$$

The DGK scheme. Let n be the product of two large prime numbers p, q such that there exists another pair of primes u, w, both of which divides $p - 1$ and $q - 1$. For random elements $g, h \in \mathbb{Z}_n^*$, where g has multiplicative order of uw and h has order equal to $w \bmod p$ and q, the encryption of a message $m \in \mathbb{Z}_u$ is

$$\mathcal{E}_{pk}(m, r) = g^m \cdot h^r \bmod n \tag{5}$$

where r is a randomly chosen integer [23]. The pubic key is (n, g, h, u) and the private key is (p, q, w). In practice, u is from a very small range, say 8-bit values, which results in a very small plain text space $\mathbb{Z}_u$, while the cipher text space is large: $\mathbb{Z}_n^*$. In order to decrypt a message, one needs to create a look-up table for all the values of $\mathbb{Z}_u$. As the message space is very small, this can be achieved efficiently. A key feature of DGK is that a ciphertext can be checked efficiently whether it is an encryption of a zero or not by raising the ciphertext c to the power of w since $c^w \bmod n = 1$ if and only if c encrypts 0. The DGK scheme is also additively homomorphic like Paillier. We refer readers to [23] for the details on generating parameters, decryption operation, and homomorphic property.

Paillier and DGK are also semantically secure. Informally, this means that one cannot distinguish between encryptions of known messages and random messages. This is achieved by having multiple possible cipher texts for each plain text, and choosing randomly between these. Semantic security is required as the messages to be encrypted in our recommender system have low entropy and could otherwise be recovered by brute-force guessing. Throughout this paper, we denote a Paillier encryption of a message m by $[m]$ and a DGK encryption by $[\![m]\!]$ for the sake of simplicity.

C. Collaborative Filtering

To generate recommendations for a particular user in a group of N users and M items, we use a system based on collaborative filtering, which has the following three steps [2].

1) Similarities are computed between that particular user and all others.
2) The L most similar users are selected by comparing their similarity values with a threshold.
3) The recommendations on all of the items are generated as the average rating of the L most similar users.

Each user in a recommender system is represented by a preference vector, which is usually formed of item ratings. Once the users have such a vector, finding a similar user is achieved by computing similarity values between these users' vectors. One common measure to find similar user is computing the Cosine similarity. For two users, A and B, with preference vectors $\mathbf{V}_A = (v_{(A,0)}, \ldots, v_{(A,M-1)})^T$ and $\mathbf{V}_B = (v_{(B,0)}, \ldots, v_{(B,M-1)})^T$, where M is the number of items and $v_{i,j}$ is a small, positive integer, the Cosine similarity is given by

$$\begin{aligned}\mathrm{sim}_{(A,B)} &= \frac{\sum_{m=0}^{M-1}(v_{(A,m)} \cdot v_{(B,m)})}{\sqrt{\sum_{m=0}^{M-1} v_{(A,m)}^2 \cdot \sum_{m=0}^{M-1} v_{(B,m)}^2}} \\ &= \sum_{m=0}^{M-1} \frac{v_{(A,m)}}{\sqrt{\sum_{m=0}^{M-1} v_{(A,m)}^2}} \cdot \frac{v_{(B,m)}}{\sqrt{\sum_{m=0}^{M-1} v_{(B,m)}^2}} \\ &= \sum_{m=0}^{M-1} \tilde{v}_{(A,m)} \cdot \tilde{v}_{(B,m)}\,.\end{aligned} \tag{6}$$

Note that in (6), $\tilde{v}_{(A,m)}$ and $\tilde{v}_{(B,m)}$ are the normalized vector elements of users A and B, respectively.

Once computed, the service provider compares the similarity values with a threshold δ. The ratings of those L users whose similarity to A exceeds δ are summed and divided by L. This result is presented as the recommendation.

In services that use collaborative filtering, the number of items offered to users is usually numbers in the hundreds of thousands. The similarity computation for each pair of users is performed for the whole item set. In practice, these vectors contain a vast amount of items that are not rated. Thus, the vectors are highly sparse and computing similarity using these highly sparse vectors leads to inaccurate results. One way to cope with the sparseness problem is to introduce a smaller set of items that are rated by most of the users [2]. In this paper, we assume there is a set of R items out of M that are rated by most of the users. Note that the selection of these R items, which is usually in the range of 10–50, is highly subjective and out of the scope of this paper. We refer readers to [25] for more information. We assume that the ith user's preference vector $\mathbf{V}_i$ is split into two parts: $\mathbf{V}_i^d = (v_{(i,0)}^d, \ldots, v_{(i,R-1)}^d)^T$ that consists of densely rated items, which will be used to compute the similarity measures, and the second part $\mathbf{V}_i^p = (v_{(i,0)}^p, \ldots, v_{(i,M-R-1)}^p)^T$ contains $M - R$ partly rated items that the user would like to get recommendations on [2].

III. Privacy-Preserving Collaborative Filtering

In this section, we introduce the privacy-preserving version of the recommender system described in Section II-C. In our setting, we have three roles:

- ***The Service Provider (SP)*** has a business interest in generating recommendations for his customers. He has resources for storage and processing.

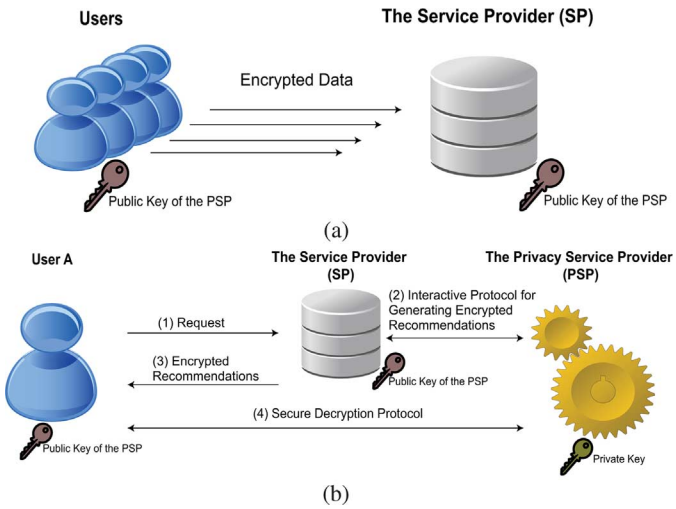

Fig. 1. System model of generating private recommendations. (a) Encrypted database construction; (b) generating private recommendations.

- ***The Privacy Service Provider (PSP)*** is a semitrusted third party who has a business interest in providing processing power and privacy functionality. The PSP has private keys for the Paillier and the DGK cryptosystems.
- ***Users*** are the customers of the service provider. Based on their preferences, in the form of ratings, the service provider generates recommendations for them.

The goal of our protocol is to hide any piece of information that may harm the privacy of users. That is, for a particular user, rating vectors, computed similarity values, results of comparing similarity values to a public threshold, and generated recommendations are all kept secret from the SP, the PSP, and all other users. Among these, only the generated recommendations and the number of users L, whose ratings are considered for generating recommendations, will be revealed to the user who asks for recommendations.

Our protocol consists of two phases as illustrated in Fig. 1: (a) construction of the encrypted database and (b) generating recommendations. To construct the encrypted database, the users encrypt their data before sending them to the service provider. To generate recommendations, the service provider and the PSP run a cryptographic protocol without interacting with the users. Notice that recommendations can be generated in a privacy-preserving way during the idle time of the service provider and the PSP even before any user asks for recommendations. This means that a user will receive recommendations shortly after his request without any delays. In the following sections, we present the details of database construction and generating recommendations and leave the performance analysis for Section VI.

A. Database Construction

To generate recommendations, we need two inputs from each user: the densely rated vector $\mathbf{V}_i^d$ to compute the similarity values between users, and the partly rated vector $\mathbf{V}_i^p$ to generate recommendations as the average rating of the top L most similar users. These vectors are highly privacy-sensitive and thus, they will be stored in the encrypted form by the service provider. Recall that the service provider does not have the decryption key, thus preventing him from accessing the users' private data. We now describe the procedure that each user has to follow prior to sending their encrypted data to the service provider.

The similarity measure given in (6) is computed over the whole item set. As explained before, we use a smaller subset of the items, which consists of R densely rated items, to compute the similarity values. Since the Paillier and the DGK cryptosystems work on integer values, the normalized vector elements given in (6) should also be scaled with a constant parameter s with enough precision and rounded to the nearest integer. After scaling and rounding, each user has a new normalized and scaled vector, $\tilde{\mathbf{V}}_i^d = (\tilde{v}_{(i,0)}^d, \ldots, \tilde{v}_{(i,R-1)}^d)^T$, whose elements are now k-bit integer values. Then the similarity value computed over R items becomes

$$\mathrm{sim}_{(A,B)} = \sum_{r=0}^{R-1} \tilde{v}_{(A,r)}^d \cdot \tilde{v}_{(B,r)}^d \,. \tag{7}$$

Note that each similarity value is now scaled by a factor of s^2. This factor increases the bit length of the similarity values, which makes the cryptographic protocols in later steps more costly. Therefore, it is important to determine the smallest scaling factor s with high precision.

Each user then element-wise encrypts his vectors $\tilde{\mathbf{V}}_i^d$ and $\mathbf{V}_i^p$ with the public Paillier key of the PSP to obtain $[\tilde{\mathbf{V}}_i^d] = ([\tilde{v}_{(i,0)}^d], \ldots, [\tilde{v}_{(i,R-1)}^d])^T$ and $[\mathbf{V}_i^p] = ([v_{(i,0)}^p], \ldots, [v_{(i,M-R-1)}^p])^T$. These encrypted vectors are then sent to and stored by the service provider.

B. Generating Recommendations

To generate recommendations for a specific user, let us say user A, the PSP and the service provider initiate a cryptographic protocol using only the encrypted vectors received by the users. This protocol has the following steps.

1) The service provider and the PSP compute the encrypted similarity values between user A and all other users.
2) The most similar users are found by comparing each similarity value with a publicly known threshold δ in the encrypted domain.
3) The service provider computes encrypted L, which is the number of users with a similarity value above the threshold. The service provider also computes the encrypted sums of the ratings of these L users for each item.
4) The service provider sends the encrypted L and the sums to user A.
5) After obtaining the number L and the sum in plain text by running a decryption protocol with the PSP, given in Appendix B, user A divides the sum of ratings by L.

In the following sections, these steps are explained in detail.

1) Computing the Encrypted Similarity Values: The similarity value computation between user A and any other user B is given in (7). Recall that the service provider has only the encrypted vectors and the PSP, who has the decryption key, is not trusted. To compute the encrypted similarity value, the service provider needs to obtain the encrypted products of vector

elements, $\tilde{v}^d_{(A,i)}$ and $\tilde{v}^d_{(A,j)}$, and add them up in the encrypted domain

$$[\mathrm{sim}_{(A,B)}] = \prod_{r=0}^{R-1} \left[\tilde{v}^d_{(A,r)} \cdot \tilde{v}^d_{(B,r)}\right] = \prod_{r=0}^{R-1} \left(\left[\tilde{v}^d_{(A,r)}\right] \otimes \left[\tilde{v}^d_{(B,r)}\right]\right) \quad (8)$$

where $\otimes$ denotes a cryptographic multiplication protocol that outputs the encrypted product of two ciphertexts as given in Appendix A. The secure multiplication protocol is invoked R times for a single similarity computation and the results are multiplied to obtain the encrypted sum. After computing $N-1$ similarity values, one for each user and excluding user A, the service provider will have a vector of $N-1$ encrypted similarity values computed for user A: $[\mathbf{SIM}_A] = ([\mathrm{sim}_{(A,0)}], \ldots, [\mathrm{sim}_{(A,N-2)}])^T$.

2) Finding the Similar Users: To find users similar to user A, each similarity value needs to be compared with a public threshold δ. Given the encrypted similarity value $[\mathrm{sim}_{(A,B)}]$ and public threshold δ, both of which are ℓ bits, the most significant bit of the value $d = 2^\ell + \mathrm{sim}_{(A,B)} - \delta$ is the outcome of the comparison: This bit is 1 if $\mathrm{sim}_{(A,i)} > \delta$ and 0, otherwise. The service provider can compute the encrypted d using the homomorphic operations but extracting the most significant bit of $[d]$ requires a cryptographic protocol based on [23], [26], and [27] as summarized below.

1) Service provider computes the encrypted d as follows:

$$[d] := [2^\ell] \cdot [\mathrm{sim}_{(A,B)}] \cdot [\delta]^{-1} = [2^\ell + \mathrm{sim}_{(A,B)} - \delta]\,. \quad (9)$$

2) Service provider masks $[d]$ by adding a random value r of size $\ell + 1 + \kappa$ bits, where κ is a security parameter

$$[z] := [d + r] = [d] \cdot [r] \quad (10)$$

and sends $[z]$ to the PSP.

3) PSP decrypts $[z]$ and sends back the encryption of $[z \div 2^\ell]$ to the service provider.

4) When the service provider receives $[z \div 2^\ell]$, he computes the most significant bit of d in the encrypted form as follows:

$$[\gamma_{(A,i)}] = [z \div 2^\ell - r \div 2^\ell - t] = [z \div 2^\ell] \cdot ([r \div 2^\ell] \cdot [t])^{-1} \quad (11)$$

where the term t is a single bit either 0 or 1 depending on the relation between $z \bmod 2^\ell$ and $r \bmod 2^\ell$. The term t is necessary because in the case where $z \div 2^\ell < r \div 2^\ell$, we need to adjust the computation by subtracting 1. Notice that comparing $\mathrm{sim}_{(A,B)}$ and δ is converted to a simpler problem of comparing $z \bmod 2^\ell$ and $r \bmod 2^\ell$, where the service provider has r and the PSP has z in plain text. To compare $z \bmod 2^\ell$ and $r \bmod 2^\ell$, we use a variant of the comparison protocol in [27]. This protocol, given in Appendix C, takes two private input values, $x = z \bmod 2^\ell$ and $y = r \bmod 2^\ell$, and outputs the result $[t]$ in encrypted form: if $x > y$, $t = 0$, and $t = 1$, otherwise.

Through the comparison protocol summarized above, the service provider and the PSP can compare each similarity value with the threshold. After running the cryptographic comparison protocol for $N-1$ similarity values, the service provider obtains the comparison results: $[\mathbf{\Gamma}_A] = ([\gamma_{(A,0)}], [\gamma_{(A,1)}] \ldots, [\gamma_{(A,N-2)}])^T$, where $[\gamma_{(A,i)}]$ is the result of comparing $\mathrm{sim}_{(A,i)}$ and threshold δ.

3) Computing the Number L and the Sum of Ratings of the Most Similar Users: The next step for the service provider is to compute the number of similar users who have a similarity value above the threshold and the sum of their ratings. Computing the number of most similar users is trivial since the service provider only needs to add all $\gamma_{(A,i)}$'s

$$[L] = \left[\sum_{i=0}^{N-2} \gamma_{(A,i)}\right] = \prod_{i=0}^{N-2} [\gamma_{(A,i)}]\,. \quad (12)$$

Computing the encrypted sum of the rating of these L users, on the other hand, requires a two-step procedure.

a) The service provider runs the secure multiplication protocol with the PSP to multiply $[\gamma_{(A,i)}]$ and the partly rated elements of user i, $[v^p_{i,j}]$, to obtain

$$[\mathbf{UR}_i] := ([\gamma_{(A,i)}] \otimes [v^p_{(i,0)}], \ldots, [\gamma_{(A,i)}] \otimes [v^p_{(i,M-R-1)}])^T = \left(\left[\gamma_{(A,i)} \cdot v^p_{(i,0)}\right], \ldots, \left[\gamma_{(A,i)} \cdot v^p_{(i,M-R-1)}\right]\right)^T = ([\mathrm{UR}_{(i,0)}], \ldots, [\mathrm{UR}_{(i,M-R-1)}])^T \quad (13)$$

where

$$[\mathbf{UR}_i] = \begin{cases} \left[\mathbf{V}^p_{(i,j)}\right], & \text{if } \gamma_{(A,i)} \text{ is } 1 \\ ([0], \ldots, [0])^T, & \text{if } \gamma_{(A,i)} \text{ is } 0\,. \end{cases} \quad (14)$$

Note that $\mathbf{UR}_i$ is computed specifically for user A. For other users to receive recommendations, the above computation should be repeated.

b) To obtain the sum of ratings of the L most similar users, the service provider multiplies all of the encrypted $\mathrm{UR}_{(i,j)}$ as follows:

$$[\mathbf{UR}_{\mathrm{sum}}] = \left([\mathrm{UR}_{(i,0)}], \ldots, \prod_{i=0}^{N-2} [\mathrm{UR}_{(i,M-R-1)}]\right)^T = \left(\left[\sum_{i=0}^{N-2} \mathrm{UR}_{(i,0)}\right], \ldots, \left[\sum_{i=0}^{N-2} \mathrm{UR}_{(i,M-R-1)}\right]\right)^T\,. \quad (15)$$

Even though the above summations for each item in the dataset involve inputs from $N-2$ users, only the ratings from the most similar users are accumulated.

Together with $[L]$, the service provider sends the $[\mathbf{UR}_{\mathrm{sum}}]$ to user A.

4) Computing the Recommendations: To obtain the recommendations, user A needs the decryptions of $[\mathbf{UR}_{\text{sum}}]$ and $[L]$. For this purpose, a simple decryption protocol is required. Depending on the existence of a direct channel between the user A and the PSP, user A performs one of the following:

1) **A channel exists**. User A runs the secure decryption protocol with the PSP, given in Appendix B, for decrypting L and each element of $\mathbf{UR}_{\text{sum}}$.
2) **A channel does not exist**. User A masks $[L]$ and each element of $\mathbf{UR}_{\text{sum}}$ with random values and sends the encryptions to the service provider. The service provider sends the masked encryptions to the PSP, who decrypts and sends the plain text values back to the service provider. After that, the service provider sends the decrypted values to the user A, who can subtract the random values to obtain L and $\text{UR}_{(i,j)}$ for j from 0 to $M-R-1$.

The recommendations are then computed by dividing each $\text{UR}_{(i,j)}$ by L. This step concludes the privacy-preserving protocol for recommender systems.

IV. Data Packing

The recommender system presented in Section III satisfies the privacy goals, but suffers from a vast amount of expensive operations on the encrypted data. There are two key observations that we can use to improve our protocol regarding efficiency:

1) The same operations (e.g., secure multiplication) are performed often.
2) The message space of the encryption scheme, which is typically 1024 bits, is not used completely as we only have relatively small messages, up to 10 bits, to encrypt.

Based on these observations, we employ data packing. Following a construction similar to [18] and [19], we pack the rating vectors, $\tilde{\mathbf{V}}_i^d$ and $\mathbf{V}_i^p$, by following a different strategy for each of them (due to the type of the follow-up operations) and then encrypt the packed values. As a result of packing, the amount of data to be transmitted between parties and the number of operations on the encrypted data, and thus the run time, will decrease linearly in relation to the amount of packing allowed. Note that for the sake of simplicity we present our protocol in this section assuming that packing all of the data in a single encryption is feasible. In reality, the message space of the cryptosytem allows only a limited number of values, namely T of them, to be packed in one encryption. In Section IV-F, we explain in detail how to determine the required number of encryptions.

A. Database Construction

Recall that each user has a vector of ratings $\mathbf{V}_i^p$ that is used for generating recommendations. The elements of the rating vector $\mathbf{V}_i^p$ for each user is to be multiplied with the same $\gamma_{(A,i)}$ and added up. Instead of encrypting each vector element individually, each user packs the vector elements of $\mathbf{V}_i^p$ to obtain a packed value V_i^{Packed} as follows:

$$V_i^{\text{Packed}} = \sum_{j=0}^{M-R-1} v_{(i,j)}^p \cdot (2^{t+\lceil \log(\hat{L}) \rceil})^j$$

where i is the user index, t is the number of bits used to represent the ratings, and $\hat{L}$ is the upper bound of the number of most similar users above a threshold δ in the system where $L < \hat{L}$. Notice that by packing in this way, each compartment is $\lceil \log(\hat{L}) \rceil$ bits larger than the bit length of the vector elements t. This additional space is required for the addition of such values, a maximum number of $\hat{L}$, in the consequent steps of the algorithm.

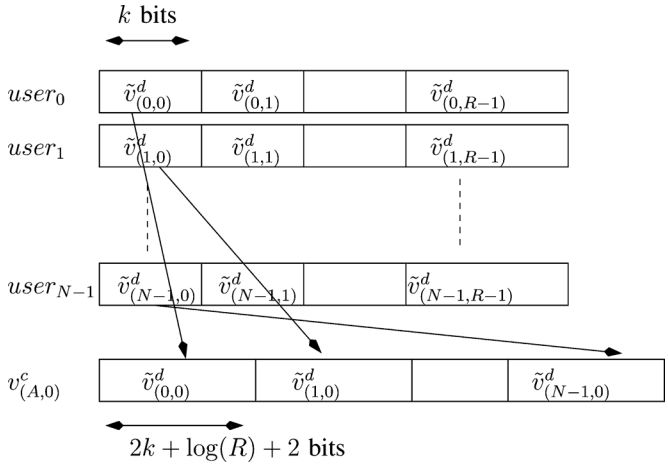

Fig. 2. Illustration of packing $\tilde{\mathbf{V}}_i^d$'s. The scaled ratings of all users with the same index j are packed in one encryption with a compartment size of $2k + \lceil \log(R) \rceil + 2$ bits to allow further processing.

Each user then encrypts the V_i^{Packed} by using the PSP's Paillier public key and sends it to the service provider with the encrypted vector $[\tilde{\mathbf{V}}_i^d]$, which is element-wise encrypted as explained in Section III-A. Note that users cannot employ data packing on $\tilde{\mathbf{V}}_i^d$ themselves because these vectors will be packed by the service provider using a different strategy that involves other users' rating vectors.

B. Similarity Computation

To generate recommendations for user A, the service provider needs to compute $N-1$ similarity values. For the similarity computations, the service provider and the PSP should run R secure multiplication protocol for each of N users, which is expensive in terms of computation and communication. Instead, the service provider packs the encrypted vector $\tilde{\mathbf{V}}_i^d$ in a particular way and invokes only R secure multiplication protocols.

The result of packing $\tilde{\mathbf{V}}_i^d$ is a new vector, $[\mathbf{V}_A^c] = ([v_{(A,0)}^c], \ldots, [v_{(A,R-1)}^c])^T$, whose terms are formed by the service provider as follows:

$$\begin{aligned}\left[v_{(A,j)}^c\right] &:= \left[\tilde{v}_{(0,j)}^d | \tilde{v}_{(1,j)}^d | \ldots | \tilde{v}_{(N-2,j)}^d\right] \\ &= \left[\sum_{i=0, i\neq A}^{N-2} 2 \cdot \tilde{v}_{(i,j)}^d \cdot (2^{2k+\lceil \log(R) \rceil + 2})^i\right] \\ &= \prod_{i=0, i\neq A}^{N-2} \left[\tilde{v}_{(i,j)}^d\right]^{2\cdot(2^{2k+\lceil \log(R) \rceil + 2})^i} \end{aligned} \tag{16}$$

for $j = 0$ to $R-1$. Notice that the content of each compartment belongs to a different user (see Fig. 2), therefore, only the service provider is capable of packing data in this way.

According to the similarity computation given in (6), the service provider computes the encrypted sum of products. As each vector element, $\tilde{v}_{(i,j)}^d$ is k bits, the product of two such vector

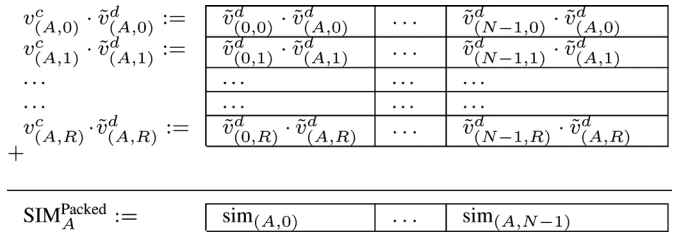

Fig. 3. Illustration of obtaining $\mathrm{SIM}_A^{\mathrm{Packed}}$ without encryption.

elements will be $2k$ bits long. The addition of R such vectors requires an extra $\lceil \log(R) \rceil$ bits. In addition to $2k + \lceil \log R \rceil$ bits, we also place two extra zero bits at the top and the bottom position of each compartment. These two additional zero bits are required for the comparison protocol that works with packed data as described in Section IV-C. In total each compartment is $2k + \lceil \log R \rceil + 2$ bits long.

Once the vector elements from all users are packed, the service provider computes the packed similarity values as follows (Fig. 3):

$$\begin{aligned}\left[\mathrm{SIM}_A^{\mathrm{Packed}}\right] &:= [\mathrm{sim}_{(A,0)}|\mathrm{sim}_{(A,1)}|\ldots|\mathrm{sim}_{(A,N-1)}] \\ &= \left[\sum_{j=0}^{R-1} v^c_{(A,j)} \cdot \tilde{v}^d_{(A,j)}\right] \\ &= \prod_{j=0}^{R-1} \left[v^c_{(A,j)} \cdot \tilde{v}^d_{(A,j)}\right] \\ &= \prod_{j=0}^{R-1} \left[v^c_{(A,j)}\right] \otimes \left[\tilde{v}^d_{(A,j)}\right]. \end{aligned} \tag{17}$$

C. Finding Similar Users

In Section III-B2, we compare the encrypted similarity values with a public threshold to find similar users by using a cryptographic protocol. With the change introduced by packing data, we now have an encryption of packed similarity values but not the separate encryptions of each similarity value

$$\left[\mathrm{SIM}_A^{\mathrm{Packed}}\right] = [\mathrm{sim}_{(A,0)}|\mathrm{sim}_{(A,1)}|\ldots|\mathrm{sim}_{(A,N-1)}]. \tag{18}$$

We need to determine the similarity values in $[\mathrm{SIM}_A^{\mathrm{Packed}}]$ that exceed a public threshold δ. In this section, we present a new version of the comparison protocol presented in Section III-B2 that works with packed data.

The desired outcome is a vector of encrypted bits, $[\mathbf{\Gamma}_A] = ([\gamma_{(A,0)}], \ldots, [\gamma_{(A,N-1)}])$, where $\gamma_{(A,i)}$ is 1 if and only if the $\mathrm{sim}_{(A,i)}$ is above the public threshold δ and 0 otherwise. Each similarity value $\mathrm{sim}_{(A,j)}$ is of size $\ell := 2k + \lceil \log(R) \rceil$ bits (and so is δ). The values were packed (Section III-B2) such that each of them is stored in the middle of an $(\ell+2)$-bit "compartment," with the top and bottom bits set to zero.

Focusing on a single comparison, the idea behind the present comparison protocol is to compute an encryption $[\tilde{d}^{(i)}] = [2^\ell + \mathrm{sim}_{(A,i)} - \delta]$ and determine the most significant bit $\tilde{d}^{(i)}_\ell$. This bit will be 1 when the similarity value matches or exceeds the threshold; naturally it must also remain secret.

Note that the computation of the encrypted $\tilde{d}^{(i)}$ can be performed in parallel on all similarity values in a single encryption. The service provider simply computes the packed list of the $\tilde{d}^{(i)}$

$$\begin{aligned}[D] :&= \left[\sum_{i=0}^{N-1} (2^{\ell+2})^i \cdot 2\tilde{d}^{(i)}\right] \\ &= \left[\mathrm{SIM}_A^{\mathrm{Packed}}\right] \cdot \left[\sum_{i=0}^{N-1} (2^{\ell+2})^i \cdot 2(2^\ell - \delta)\right]. \end{aligned} \tag{19}$$

Each of them is at most $\ell + 1$ bits long and, therefore, fits in a single compartment, as this was constructed with one bit of head room, i.e., an additional bit set to zero. We denote the $(\ell+2)$-bit value of the entire ith compartment $d^{(i)}$

$$d^{(i)} = 2\tilde{d}^{(i)} = 2^{\ell+1} + 2\mathrm{sim}_{(A,i)} - 2\delta$$

and specify the desired outcome as the $d^{(i)}_{\ell+1}$.

The service provider blinds $[D]$—and thus the $d^{(i)}$ contained therein—by adding a uniformly distributed random $(\kappa + (\ell + 2)N)$-bit number r

$$[z] := [D] \cdot [r] = [D + r] \tag{20}$$

where κ is a security parameter.

The service provider then computes $r^{(i)}$ such that $r \bmod 2^{N(\ell+2)} = \sum_{i=0}^{N-1} r^{(i)}(2^{\ell+2})^i$; each $r^{(i)}$ corresponds to the random value stored in compartment i. Notice that similar to the $d^{(i)}$, the integers $r^{(i)}$ are $\ell + 2$ bits long. At this point the service provider sends$[z]$ to the PSP.

The PSP decrypts the received $[z]$ and computes each $z^{(i)}$ value such that $z \bmod 2^{N(\ell+2)} = \sum_{i=0}^{N-1} z^{(i)}(2^{\ell+2})^i$, i.e., the unpacking of z in the plain domain. At this point each of the N compartments have been separated into values, $r^{(i)}$ and $z^{(i)}$ such that $z^{(i)} = r^{(i)} + d^{(i)} \bmod 2^{\ell+2}$, *except* that carry-overs may propagate from one compartment to the next.

As $z = D + r$, it is clear that for every bit-position j, $z_j = D_j \oplus r_j \oplus C_j$, where C_j is the jth carry-bit of the addition of D and r. Hence, we have

$$d^{(i)}_{\ell+1} = r^{(i)}_{\ell+1} \oplus z^{(i)}_{\ell+1} \oplus C_{i(\ell+2)+(\ell+1)}. \tag{21}$$

If additive sharings of these N carry-bits modulo 2 were given (i.e., if the PSP knew $C^{\mathrm{PSP}}_{i(\ell+2)+(\ell+1)}$ and the service provider knew $C^{\mathrm{SP}}_{i(\ell+2)+(\ell+1)}$, such that $C_{i(\ell+2)+(\ell+1)} = C^{\mathrm{PSP}}_{i(\ell+2)+(\ell+1)} \oplus C^{\mathrm{SP}}_{i(\ell+2)+(\ell+1)}$), then for each similarity value, the PSP could simply encrypt $d^{(i,\mathrm{PSP})}_{\ell+1} = z^{(i)}_{\ell+1} \oplus C^{\mathrm{PSP}}_{i(\ell+2)+(\ell+1)}$ and send it to the service provider.

The service provider could then compute the encryptions of the $d^{(i)}_{\ell+1}$, which is actually the comparsion result $\gamma_{(A,i)}$, by either retaining $[d^{(i,\mathrm{PSP})}_{\ell+1}]$ or flipping it (computing $[1] \cdot [d^{(i,\mathrm{PSP})}_{\ell+1}]^{-1} = [1 - d^{(i,\mathrm{PSP})}_{\ell+1}]$) depending on the value of $r^{(i)}_{\ell+1} \oplus C^{\mathrm{SP}}_{i(\ell+2)+(\ell+1)}$. The service provider and the PSP determine the shared carry-bits by running the protocol given in Appendix D.

After running the above comparison protocol, the service provider has $N-1$ encryptions, $[d_{\ell+1}^{(i)}] = [\gamma_{(A,i)}]$.

D. Computing the Number L and the Sum of Ratings of the Most Similar Users

In this part of the protocol, the number L is computed as before. To compute the encrypted sum of the ratings of the most similar users, we need to multiply the packed rating vector of each user $[V_i^{\text{Packed}}]$ and the comparison result $\gamma_{(A,i)}$. For this purpose, the service provider and the PSP run the secure multiplication protocol only once per user to obtain the encryption of the packed ratings

$$\begin{aligned}\left[\text{UR}_i^{\text{Packed}}\right] &= \left[V_i^{\text{Packed}} \cdot \gamma_{(A,i)}\right] \\ &= \left[V_i^{\text{Packed}}\right] \otimes [\gamma_{(A,i)}]\end{aligned} \tag{22}$$

where

$$\text{UR}_i = \begin{cases} V_i^{\text{Packed}}, & \text{if } \gamma_{(A,i)} = 1 \\ 0, & \text{if } \gamma_{(A,i)} = 0. \end{cases} \tag{23}$$

After the secure multiplication protocols finish, the service provider multiplies the results to obtain the encrypted sum of ratings from all other similar users

$$\begin{aligned}[\text{UR}^{\text{sum}}] : &= \left[\sum_{i=0,\gamma_{(A,i)}==1}^{N-2} V_i^{\text{Packed}}\right] \\ &= \left[\sum_{i=0}^{N-2} \text{UR}_i^{\text{Packed}}\right] = \prod_{i=0}^{N-2} \left[\text{UR}_i^{\text{Packed}}\right]\end{aligned} \tag{24}$$

where UR^{sum} is the packed sum of the ratings of the most similar users. The service provider sends $[\text{UR}^{\text{sum}}]$ and $[L]$ to user A.

Notice that by using data packing the service provider reduces the number of secure multiplication protocol invocations and also the number of multiplication in the encrypted domain from $(N-1)\cdot(M-R)$ to N.

The value of L, which will be different for every user, is only determined at the end of the protocol in the encrypted form. This encryption is only accessible to the user, who requested recommendations. As the value L is unknown to the service provider, L may exceed the upper bound $\hat{L}$. In such a case, intermediate values will not fit to the compartments and thus computations will be wrong. In order to check whether an error has occurred or not, user A compares the values L and $\hat{L}$. In the case where L is greater than $\hat{L}$, user A asks the service provider to repeat the protocol starting from the comparison step, only this time with a new threshold, larger than the previous one. In this way, there will be less users with a similarity above the new threshold and the encryptions sent by other users do not need to be repacked using a larger $\hat{L}$.

E. Computing the Recommendations

After receiving the encrypted UR^{sum} and L, user A runs the secure decryption protocol as explained in Section III-B4. The only difference is instead of $M-R+1$ invocation of the protocol, user A only runs the protocol twice due to data packing. The final recommendations are computed by first unpacking the decrypted UR^{sum} and dividing each compartment content with L.

F. The Number of Encryptions Needed for Packing

In the presentation of the protocol, for the sake of simplicity we assumed that packing all values in one encryption is possible. We clarify this point in this section. For the computation of V_i^{Packed}, the number of vector elements $v_{(i,j)}^p$ that can be packed into one encryption is $T_1 = \lfloor(\log(n))/(t+\lceil\log(\hat{L})\rceil)\rfloor$, where n is the message space of the Paillier cryptosystem and $t+\lceil\log(L)\rceil$ is the number of bits required for adding L ratings of size t bits. As a result, the number of V_i^{Packed} values that each user needs to send is $S_1 = \lceil(M-R)/(T_1)\rceil$.

Similarly, the service provider may need more than one encryption for packing $[\tilde{\mathbf{V}}_i^d]$. The number of v^c values required in total is $S_2 = \lceil(N)/(T_2)\rceil$, where $T_2 = \lfloor(\log(n)-\kappa)/(2k+\lceil\log(R)\rceil+2)\rfloor$, where κ is a security parameter. The value S_2 is also the total number of $\text{SIM}_A^{\text{Packed}}$ values.

V. Security Analysis

We assume that all participants in the recommender system, including users, service provider, and PSP, are honest-but-curious. They all follow the rules of the protocol, but will collect all their information and try to extrapolate information from this. We assume that the service provider and PSP do not collude—procedural, organizational, or legal steps should be taken to ensure this. Both may, however, collude with users, potentially including user A.

Intuitively, the inputs of honest parties and A's output are hidden from any attacker. The service provider only sees encryptions of the inputs under the PSP's public key, while the PSP only receives (encryptions of) masked values from which no information can be learned. We expand on this intuition below.

A. Users

With the exception of A, the users simply provide their input, i.e., encrypt it under the key of the PSP, and send it to the service provider. It is clear that they cannot learn any information about other user's inputs from this.

B. Service Provider

The service provider receives the encrypted inputs of all parties. Further, it receives encryptions of intermediate results from the PSP during the protocol, i.e., messages that are part of the multiplication and comparison protocols. However, all these messages are simply encryptions under one of the PSP's public keys. Since both the Paillier and DGK encryption schemes are semantically secure, all messages received from honest users and the honest PSP are computationally indistinguishable from encryptions of uniformly random values. It is therefore impossible for the service provider to extract any private information, even when colluding with a subset of the users (other than user A).

C. PSP

The initial state of the PSP consists of its public and private key pair. Because of the ability to decrypt, all messages sent to the PSP during the protocol are masked with a random value. Note that our security argument is independent of any curious users. A curious user reveals the decryptions of his messages and the randomness used to encrypt them, thus knowing also the inputs and randomness of any curious users provide no additional information.

During the secure multiplication protocol, the service provider adds random values that are κ (the security parameter) bits longer than the actual ones. These sums are statistically indistinguishable from random values of the same bit-length as the masks; hence, the messages received during the protocol are indistinguishable from fresh encryptions of random values. When determining the L users with highest similarity, the PSP obtains $[\mathrm{SIM}_A^{\mathrm{Packed}} + \mathrm{random\ value}]$, which also reveals no information by the same argument. This is also the case when running the secure decryption protocol at the end.

Security of the comparison protocol in Appendix C is analogous. During each invocation, the PSP obtains DGK encrypted values. Each encryption contains a masked bit t_i. The masking is more complicated than above, but the outcome is the same: the values are indistinguishable from random values containing no information.

D. User A

It remains to argue that user A does not learn any information other than the desired result, even if he colludes with the service provider or PSP. First, note that before the execution of the secure decryption protocol with the PSP in the end, user A is no different from any other user (except for the role of its input). In particular, user A has received no messages so he cannot know inputs of other users.

User A receives a fresh encryption of $\mathrm{UR}^{\mathrm{sum}}$ from the service provider. If user A colludes with the PSP, this can be decrypted directly; however, doing so leaks no additional information, as user A is supposed to learn $\mathrm{UR}^{\mathrm{sum}}$. Alternatively, if A colludes with the service provider, it simply receives a message that the adversary already knew. User A then computes an encryption of $\mathrm{UR}^{\mathrm{sum}} + r$, where r is a random mask. This is sent to the PSP for decryption, and the plaintext returned. Clearly—as user A knows r and should learn $\mathrm{UR}^{\mathrm{sum}}$—this reveals no information. Thus, a curious A cannot learn any additional information even if it colludes with either the service provider or PSP as well as a subset of the other users.

VI. Complexity Analysis and Performance Results

In this section, we present the complexity analysis and performance results of the two versions of the privacy-preserving recommender system, with and without data packing, to measure the performance of the cryptographic protocols in terms of bandwidth and runtime. For the experiments, we created a synthetic dataset with $N = 10\,000$ users and $M = 1000$ items. Each rating is randomly set to a value between 0 and 5. The protocol is implemented in C++, using the GMP Library version 4.2.1 and tested on a single computer with Intel Xeon 2.33-GHz processor and 16 GB of memory, ignoring network latency. The parameters used for the tests are given in Table I. These values are selected empirically. Variables of the protocol computed based on these parameters are given in Table II.

TABLE I
Parameter List Used for the Experiments

Parameter	Symbol	Value
Number of users	N	10,000
Number of items	M	1,000
Size of vector $\mathbf{V}^d$	R	20
Range of the rating	K	0 to 5, inclusive
Bit size of the ratings	t	3 bits
Scaling factor	s	1000
Bit size of the scaled ratings	k	10 bits
Upper bound for the number of most similar users	L	256
Threshold	δ	0.87
Message space of the cryptosystems	n	1024 bits
Security parameter	κ	40 bits

TABLE II
Variables and Their Values

Variable	Symbol	Value
Number of $v^p_{(i,j)}$ packed into one encryption	T_1	93
Number of UR_i values	S_1	11
Number of $\tilde{v}^d_{(i,j)}$ packed into one encryption	T_2	36
Number of $v^c_{(A,j)}$ values	S_2	278
Size of the similarity values	ℓ	25 bits

A. Bandwidth

The communication complexity of the privacy-preserving recommender system with and without data packing is given in Table III. In the case of using data packing, a single user sends and receives 11.5 KB of data during the whole protocol, based on the numbers given in Tables I and II. The service provider sends and receives 147 MB of data, 112 MB of which is received from 10 000 users as the encrypted ratings. Recall that receiving encrypted data from the users takes place only once, as long as their ratings do not change. The service provider exchanges 35 MB of data with the PSP during generating recommendations.

The recommender system *without* data packing is significantly more expensive: a single user needs to send 495 KB of data to the service provider. The service provider receives 2.3 GB of data from the users during the database construction phase. Running the comparison protocol without data packing requires the service provider and the PSP to exchange 2.3 MB more data than the one with data packing. For running the secure multiplication protocols, the service provider and the PSP exchange 7.15 GB of data.

B. Run-Time

The computational complexity is dependent on the cost of operations in the encrypted domain and can be categorized into four classes: encryptions, decryptions, multiplications, and exponentiations. We assume operations in the plaintext domain are free. In Table III, we give the complexity of the protocol in terms of operations in the Paillier and the DGK cryptosystems. One exception is the decryption operation for the DGK

TABLE III
COMMUNICATION AND COMPUTATIONAL COMPLEXITY FOR THE SERVICE PROVIDER (SP), INTERACTION WITH N USERS, THE PRIVACY SERVICE PROVIDER (PSP) AND THE USER, WHO RECEIVES THE RECOMMENDATIONS, FOR THE WHOLE PROTOCOL INCLUDING THE OFFLINE COMPUTATIONS

With Data Packing						
	SP		PSP		User	
	Paillier	DGK	Paillier	DGK	Paillier	DGK
Encryption	$\mathcal{O}(NS_1 + RS_2)$	-	$\mathcal{O}(NS_1 + RS_2)$	$\mathcal{O}(N\ell)$	$\mathcal{O}(R + S_1)$	-
Decryption	-	-	$\mathcal{O}(NS_1 + RS_2)$	$\mathcal{O}(N)$	-	-
Multiplication	$\mathcal{O}(NS_1 + RS_2)$	$\mathcal{O}(N\ell)$	-	-	$\mathcal{O}(S_1)$	-
Exponentiation	$\mathcal{O}(NR)$	-	-	-	-	-
Communication	$\mathcal{O}(N(R + S_1))$	$\mathcal{O}(N\ell)$	$\mathcal{O}(RS_2 + S_1)$	$\mathcal{O}(N\ell)$	$\mathcal{O}(R + S_1)$	-
Without Data Packing						
	SP		PSP		User	
	Paillier	DGK	Paillier	DGK	Paillier	DGK
Encryption	$\mathcal{O}(NM)$	-	$\mathcal{O}(NM)$	$\mathcal{O}(N\ell)$	$\mathcal{O}(M)$	-
Decryption	-	-	$\mathcal{O}(NM)$	$\mathcal{O}(N)$	-	-
Multiplication	$\mathcal{O}(NM)$	$\mathcal{O}(N\ell)$	-	-	$\mathcal{O}(M)$	-
Exponentiation	$\mathcal{O}(NM)$	-	-	-	-	-
Communication	$\mathcal{O}(NM)$	$\mathcal{O}(N\ell)$	$\mathcal{O}(NM)$	$\mathcal{O}(N\ell)$	$\mathcal{O}(M)$	-

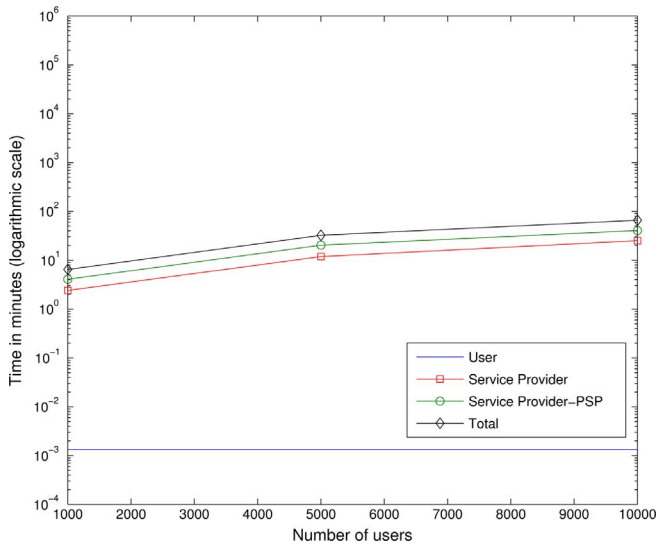

Fig. 4. Average runtime of the protocol steps with data packing. For 10 000 users, it takes 10 min for the SP to pack the data, 2.5 min for the SP and PSP to compute the similarity values, 35 s to compare them with the threshold, and roughly 38 min to generate the recommendations.

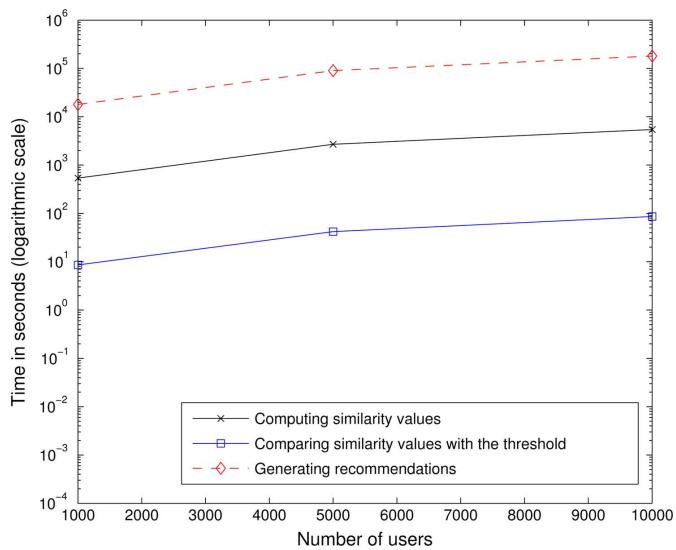

Fig. 5. Average runtime of the protocol steps without data packing. For 10 000 users, it takes approximately 90 min to compute the similarity values, 2 min to compare them with the public threshold, and 50 h to generate recommendations.

cryptosystem, which is actually a *zero-check* as mentioned in Section II-B which is a faster and less expensive operation compared to original DGK decryption.

As seen in Table III, the complexity for the offline computations between the service provider and the PSP is linear in the number of users N. However, by using data packing, the complexity for many operations is reduced from $\mathcal{O}(N\,M)$ to $\mathcal{O}(N\,S_1)$. A single user needs to encrypt only $R + 2S_1$ messages and run the secure decryption protocol for $S_1 + 1$ values. This is actually the cost of running the privacy-preserving recommender system from the user perspective: encrypting his input and running the secure decryption protocol to obtain the recommendations.

We present the experimental results for the average runtime of the protocol with and without data packing over 10 runs in Figs. 4 and 5, respectively. The overall runtime in either case is linear in the number of users in the system. Notice that while it takes 50 min for the service provider and the PSP to generate recommendations for a single user in a system with 10 000 users, it takes approximately 1 s for that user to encrypt his data and to finish the secure decryption protocols with data packing. We anticipate that with code optimization, parallelization, and dedicated hardware, the overall runtime can be reduced significantly.

The runtime of the protocol without data packing is considerably longer since the service provider and PSP should perform operations over encrypted data in large numbers. Analysis shows that the total time required for packing data, computing similarity values, and comparing them with the public threshold within the protocol that uses data packing takes much less time than only computing similarity values without data packing. As seen in Fig. 5, the operation required for generating recommendations, even with 1000 users, is completely infeasible without data packing due to the immense number of secure multiplications.

VII. Comparison to Related Work

Regarding other papers which are based on statistical techniques like perturbation of data [7], [8], offline profile hiding [10], and agents [12], our protocol is not comparable since we use cryptographic techniques to protect the privacy of the users. Given the complexity analysis and experimental results, our protocol outperforms the comparable work by Canny [14], which is also based on cryptographic techniques, regarding both communication and computational costs. 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 $T \times M$. Note that this complexity is only for one iteration of that protocol. As Canny suggests, the protocol converges in 40–60 iterations. With 10 000 users, Canny estimates a communication load of 4 GB or 600 MB per user using the ElGamal Cryptosystem [28] or Elliptic Curve Cryptography [29], respectively. The run time estimation for both cryptosystems is about 15 h on a 500-MHz machine. Notice that in our protocol the overall run time is roughly 50 min and the communication cost per user is only around 10 KB. Even the heavy communication load on the service provider side of our protocol is much smaller compared to the communication cost of a single user in [14].

Compared to [16], our protocol has the same communication complexity. However, the computational cost in total is increased as a result of using a third party, the PSP. Recall that the idea of adding the PSP to the protocol is to reduce the workload of the users. While we achieve this goal, an extra computational burden is assigned to the service provider and the PSP, which invoke a vast amount of secure multiplication protocols, as shown in Fig. 6. Within the same experimental setup, [16] is estimated to have a run time of approximately 8 min, which is much faster than our protocol. However, during these 8 min, all of the users have to be online and participate in the computations. In our work, a run time of 50 min is required for the computations between the service provider and the PSP without users' participation.

VIII. Discussion

We have presented a highly efficient, privacy-preserving cryptographic protocol for a crucial component of online services: recommender systems. Our construction with a semitrusted third party, the PSP, ensures a protocol where user participation in the heavy cryptographic operations is no longer needed. We also employ data packing to ease the computational and communication burden between the service provider and the PSP. A cryptographic protocol particularly developed for comparing packed and encrypted values, enables us to compare multiple encrypted data elements in a single operation.

We assume our recommender system is static, meaning that the two sets of items we consider in this work are fixed. Dynamic behaviors such as updating the set of R items that are used for similarity computations and removing items from the dataset will cause the service provider and the PSP to run the privacy-preserving protocol from the beginning with the new data. We leave extending our protocol to a dynamic recommender system as future work.

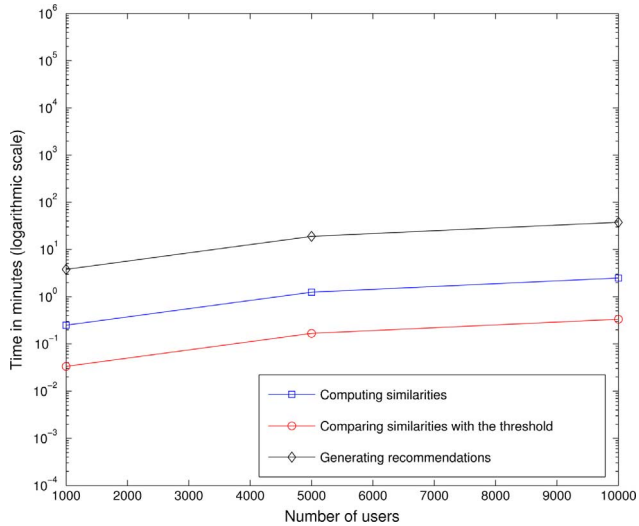

Fig. 6. Average runtime of interactive protocol steps between the service provider and the PSP. Generating recommendations makes up 76% of the computations due to large number of invocations of the secure multiplication protocol.

In our proposed system, the number of users L, who have a similarity value exceeding δ, is available to user A in plain text, which might leak information on the recommender system. In order to hide L from the user, it should be kept encrypted. In this case, the recommendations have to be computed by running a secure *division* protocol, which divides encrypted total by encrypted L as in [30]. Such an approach will introduce additional overhead on the user's side. Notice that even if secure division is chosen, the encrypted L has to be compared with $\hat{L}$, which introduces additional overhead again. Unfortunately, at the end of the comparison, both the service provider and user A will know the outcome of $L < \hat{L}$ since the service provider needs this information to validate the generated recommendations. Since in either scenario a certain amount of information is leaked, we permit users to know the number of similar users for efficiency reasons.

While the experimental results show a significant improvement of the state-of-the-art due to the new setting with a semitrusted third party, data packing, and fine-tuned cryptographic protocols, the scalability of the proposed system remains as a challenge. Analysis shows that the most time-consuming operations, approximately 76% of the entire computational effort, take place during the recommendation generation step, where encrypted ratings of each user need to be multiplied with the encrypted comparison results by using the secure multiplication protocol. Note that even if the comparison protocol given in Appendix C, which constitutes only 0.6% of the computations, is replaced with other constructions such as the ones in [31] and [32], the overall runtime of the protocol will still be dominated by the cost of the secure multiplication protocol. At this moment, a way to improve the secure multiplication protocol using existing cryptographic protocols remains as an open question.

APPENDIX

A. Secure Multiplication Protocol

The secure multiplication protocol in [33] and [26] can be adapted to a two-party protocol in which one party, Alice, has two encrypted values $[a]$ and $[b]$, and the other party, Bob, has the decryption key. The protocol outputs the encrypted value $[a \cdot b]$ to Alice without Bob learning a or b. Knowing that the encryption scheme is additively homomorphic,

1) Alice generates two uniformly distributed random numbers r_1 and r_2, and subtracts these numbers from the encryptions $[a]$ and $[b]$, respectively: $[\tilde{a}] := [a] \cdot [-r_1] = [a - r_1]$, $[\tilde{b}] := [b] \cdot [-r_2] = [b - r_2]$. Afterwards, she sends $[\tilde{a}]$ and $[\tilde{b}]$ to Bob.
2) Bob decrypts $[\tilde{a}]$ and $[\tilde{b}]$, multiplies them and sends the encrypted product $[\tilde{a} \cdot \tilde{b}]$ to Alice.
3) Alice removes the random values and obtains the encryption of the product of a and b as follows: $[a \cdot b] := [\tilde{a} \cdot \tilde{b}] \cdot [a]^{r_2} \cdot [b]^{r_1} \cdot [-r_1 \cdot r_2] = [\tilde{a} \cdot \tilde{b} + a \cdot r_2 + b \cdot r_1 - r_1 \cdot r_2]$.

B. Secure Decryption Protocol

Similar to the secure multiplication protocol, a secure decryption protocol can be designed based on [33]. In this protocol, Alice demands for the decryption of an encrypted value, $[a]$ without revealing it to Bob, the owner of the decryption key. Using an additively homomorphic cryptosystem, this protocol can be summarized as follows:

1) Alice generates a uniformly random number r and blinds the encryption with this number: $[\tilde{a}] := [a] \cdot [r] = [a + r]$. Then, Alice sends $[\tilde{a}]$ to Bob.
2) Bob decrypts $[\tilde{a}]$ and sends it back to Alice.
3) Alice obtains the decryption of $[a]$ by subtracting r from $\tilde{a}$: $a := \tilde{a} - r$.

C. Comparison Protocol

In order to compare $r^{(i)} \bmod 2^{\ell+1}$ and $z^{(i)} \bmod 2^{\ell+1}$, we use a comparison protocol based on the ideas in [27]. This protocol takes two private input values, $a = r^{(i)} \bmod 2^{\ell+1}$ and $b = z^{(i)} \bmod 2^{\ell+1}$, and outputs the result $[t]$ in encrypted form: if $a > b$, $t = 1$, and $t = 0$, otherwise. The comparison result $t = t_\ell$ is computed recursively as follows:

$$t_{i+1} = (1 - (a_i - b_i)^2)t_i + a_i(1 - b_i), \text{for } 0 \leq i < \ell + 1 \tag{25}$$

where $t_0 = 0$ and a_i and b_i are the ith bits of a and b, respectively. The recursive equation (25) can be computed efficiently by using the DGK scheme. Note that the result of the comparison should be converted to the Paillier scheme. This protocol is given in Protocol 1.

D. Obtaining the $\oplus$ Sharing of the Carry-Bits

In Section IV-C, the comparison of similarity values $\text{sim}_{(A,i)}$ to the threshold δ is reduced to that of obtaining N additive sharings modulo 2 of carry-bits of the addition of the secret D and the random r generated by the service provider. Each of these is computed by running a comparison protocol where the service provider knows one input and the PSP the other. This subprotocol is adapted from [23] and [26].

Protocol 1 Comparing private integers

Input SP	$a = a_{\ell-1} \ldots a_0$
Input PSP	$b = b_{\ell-1} \ldots b_0$ and the decryption key K
Output SP	bit t_{SP}
Output PSP	bit t_{PSP}
Relation	Shared bit $t = t_{SP} \oplus t_{PSP}$ is such that $(t = 1) \equiv (a > b)$

The PSP encrypts b_0 and sends $[\![b_0]\!]$ to SP
if $a_0 = 0$ **then**
  SP: $[\![t]\!] \leftarrow [\![0]\!]$
else
  SP: $[\![t]\!] \leftarrow [\![1]\!] \cdot [\![b_0]\!]^{-1} \bmod n$
end if
$\{(t = 1) \equiv (t_1 = 1) \equiv (a_0 > b_0)\}$
for $i \leftarrow 1, \ldots, \ell - 1$ **do**
  $\{t = t_i\}$
  SP blinds $t = t_i$ by tossing a fair coin $c \in \{0, 1\}$
  if $c = 0$ **then**
    SP: $[\![\tau]\!] \leftarrow [\![t]\!]$
  else
    SP: $[\![\tau]\!] \leftarrow [\![1]\!] \cdot [\![t]\!]^{-1} \bmod n$
  end if
  $\{\tau = c \oplus t_i\}$
  SP randomizes $[\![\tau]\!]$ and sends it to the PSP
  if $b_i = 0$ **then**
    PSP: $[\![tb]\!] \leftarrow [\![0]\!]$
  else
    PSP: $[\![tb]\!] \leftarrow [\![\tau]\!]$
  end if
  $\{tb = \tau \cdot b_i\}$
  The PSP encrypts b_i
  The PSP randomizes $[\![tb]\!]$
  The PSP sends $[\![tb]\!]$ and $[\![b_i]\!]$ to the SP
  $\{tb = (c \oplus t_i) \cdot b_i\}$
  if $c = 1$ **then**
    SP: $[\![tb]\!] \leftarrow [\![b_i]\!] \cdot [\![tb]\!]^{-1} \bmod n$
  end if
  $\{tb = t_i \cdot b_i\}$
  if $a_i = 0$ **then**
    SP: $[\![t]\!] \leftarrow [\![t]\!] \cdot [\![tb]\!]^{-1} \bmod n$
  else
    SP: $[\![t]\!] \leftarrow [\![tb]\!] \cdot [\![1]\!] \cdot [\![b_i]\!]^{-1} \bmod n$
  end if
  $\{t = t_{i+1}\}$
end for
$\{t = t_\ell\}$
SP blinds $t = t_\ell$ by tossing a fair coin $c \in \{0, 1\}$
if $c = 0$ **then**
  SP: $[\![\tau]\!] \leftarrow [\![t]\!]$
else
  SP: $[\![\tau]\!] \leftarrow [\![1]\!] \cdot [\![t]\!]^{-1} \bmod n$
end if
$\{\tau = c \oplus t_\ell\}$
SP randomizes $[\![\tau]\!]$ and sends it to the PSP
The PSP decrypts $[\![\tau]\!]$
$\{t_{PSP} = \tau$, $t_{SP} = c$, and $t_\ell = t_{PSP} \oplus t_{SP}\}$

The key observation is that if the desired carry-bit of compartment i is set, then $r^{(i)} \bmod 2^{\ell+1} > z^{(i)} \bmod 2^{\ell+1}$. Clearly the sum (modulo $2^{\ell+1}$) of $d^{(i)} \bmod 2^{\ell+1}$ and $r^{(i)} \bmod 2^{\ell+1}$ is bigger than $r^{(i)} \bmod 2^{\ell+1}$ except if an overflow occurred. An overflow from the compartment below will not change this. As $d^{(i)} = 2\tilde{d}^{(i)}$, it is guaranteed that $d_0^{(i)} = 0$. Hence, a propagated carry may simply be viewed as "part of $d^{(i)}$" in the above intuition.

At this point, all that is needed is a comparison of each of the $r^{(i)} \bmod 2^{\ell+1}$ and $z^{(i)} \bmod 2^{\ell+1}$, where the outcome is $\oplus$ shared. This happens just before termination in the comparison of [26]. However, instead of using the comparison protocol in [26], we use a variant of the protocol described in [27] which is computationally more efficient. We list the overall steps performed; for a full description see Appendix C.

1) The PSP and the service provider runs a cryptographic protocol that compares these two values in an iterative manner. In each iteration, the PSP sends DGK encryptions of the bits of each of the $z^{(i)} \bmod 2^{\ell+1}$; $N(\ell+1)$ encryptions in all.
2) Based on the bits of the $r^{(i)} \bmod 2^{\ell+1}$, the service provider computes encryptions containing only a masking of the desired result; the last one is then sent to the PSP. The main idea here is that the service provider picks a uniformly random bit $b^{(i)}$ for each iteration and specifies the goal as either $r^{(i)} \geq z^{(i)}$ or $z^{(i)} \geq r^{(i)}$ depending on these.
3) The PSP decrypts and determines the comparison results. However, as it does not know the $b^{(i)}$—i.e., the direction of the comparisons—no information is revealed (the $b^{(i)}$'s can be viewed as onetime pads).

This concludes the computation, as the $b^{(i)}$'s and the outcomes of the comparisons are exactly $\oplus$ sharings of the results. The PSP and the service provider then use the shared results they have obtained here to provide the service provider with an encryption of each of the results of the comparisons of the $\mathrm{sim}_{(i,j)}$ and δ as described in Section IV-C.

REFERENCES

[1] List of Social Networking Websites 2009 [Online]. Available: http://en.wikipedia.org/wiki/List_of_social_networking_websites

[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. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[3] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis, "Privacy risks in recommender systems," *IEEE Internet Comput.*, vol. 5, no. 6, pp. 54–63, Nov./Dec. 2001.

[4] N. Kroes, Digital agenda, Brussels, May 19, 2011.

[5] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proc. SIGMOD Rec.*, May 2000, vol. 29, pp. 439–450.

[6] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *J. Cryptol.*, pp. 36–54, 2000, Springer-Verlag.

[7] H. Polat and W. Du, "Privacy-preserving collaborative filtering using randomized perturbation techniques.," in *Proc. ICDM*, 2003, pp. 625–628.

[8] H. Polat and W. Du, "SVD-based collaborative filtering with privacy," in *Proc. 2005 ACM Symp. Applied Computing (SAC'05)*, New York, NY, 2005, pp. 791–795, ACM Press.

[9] S. Zhang, J. Ford, and F. Makedon, "Deriving private information from randomly perturbed ratings," in *Proc. Sixth SIAM Int. Conf. Data Mining*, 2006, pp. 59–69.

[10] R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. Hubaux, "Preserving privacy in collaborative filtering through distributed aggregation of offline profiles," in *Proc. Third ACM Conf. Recommender Systems (RecSys'09)*, New York, NY, 2009, pp. 157–164, ACM.

[11] F. McSherry and I. Mironov, "Differentially private recommender systems: Building privacy into the net," in *Proc. 15th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'09)*, New York, NY, 2009, pp. 627–636, ACM.

[12] R. Cissée and S. Albayrak, "An agent-based approach for privacy-preserving recommender systems," in *Proc. 6th Int. Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS'07)*, New York, NY, 2007, pp. 1–8, ACM.

[13] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara, "Private collaborative forecasting and benchmarking," in *Proc. 2004 ACM Workshop on Privacy in the Electronic Society (WPES'04)*, New York, NY, 2004, pp. 103–114, ACM.

[14] J. F. Canny, "Collaborative filtering with privacy.," in *IEEE Symp. Security and Privacy*, 2002, pp. 45–57.

[15] J. F. Canny, "Collaborative filtering with privacy via factor analysis," in *SIGIR*. New York, NY: ACM Press, 2002, pp. 238–245.

[16] Z. Erkin, M. Beye, T. Veugen, and R. L. Lagendijk, "Privacy enhanced recommender system," in *Proc. Thirty-First Symp. Information Theory in the Benelux*, Rotterdam, 2010, pp. 35–42.

[17] Z. Erkin, M. Beye, T. Veugen, and R. L. Lagendijk, "Efficiently computing private recommendations," in *Proc. Int. Conf. Acoustic, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 5864–5867, 2011.

[18] J. R. Troncoso-Pastoriza, S. Katzenbeisser, M. U. Celik, and A. N. Lemma, "A secure multidimensional point inclusion protocol," in *Proc. ACM Workshop on Multimedia and Security*, 2007, pp. 109–120.

[19] T. Bianchi, A. Piva, and M. Barni, "Composite signal representation for fast and storage-efficient processing of encrypted signals," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 1, pp. 180–187, Mar. 2010.

[20] O. Goldreich, *Foundations of Cryptography. Basic Applications*, 1st ed. Cambridge, U.K.: Cambridge Univ. Press, May 2004, vol. 2, ISBN 0-521-83084-2.

[21] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft, "Secure multiparty computation goes live," in *Proc. Financial Cryptography*, 2009, pp. 325–343.

[22] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Advances in Cryptology (EUROCRYPT'99), ser. LNCS*, J. Stern, Ed., May 2–6, 1999, vol. 1592, pp. 223–238, Springer.

[23] I. Damgård, M. Geisler, and M. Krøigaard, "Efficient and secure comparison for on-line auctions," in *Proc. Australasian Conf. Information Security and Privacy (ACSIP 2007), ser. LNCS*, J. Pieprzyk, H. Ghodosi, and E. Dawson, Eds., Jul. 2–4, 2007, vol. 4586, pp. 416–430, Springer.

[24] I. Damgård, M. Geisler, and M. Krøigaard, A Correction to "Efficient and Secure Comparison for On-Line Auctions" Cryptology ePrint Archive, Report 2008/321, 2008 [Online]. Available: http://eprint.iacr.org/

[25] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proc. 22nd Ann. Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR'99)*, New York, NY, 1999, pp. 230–237, ACM.

[26] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, R. L. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Proc. Privacy Enhancing Technologies Symp.*, Seattle, WA, 2009, pp. 235–253.

[27] B. Schoenmakers and P. Tuyls, "Practical two-party computation based on the conditional gate," in *Proc. Advances in Cryptology*, 2004, vol. 3329, pp. 119–136, Springer-Verlag.

[28] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.

[29] I. F. Blake, G. Seroussi, and N. P. Smart, *Elliptic Curves in Cryptography*. New York, NY: Cambridge Univ. Press, 1999.

[30] R. Lazzeretti and M. Barni, "Division between encrypted integers by means of garbled circuits," in *Proc. 3rd IEEE Int. Workshop on Information Forensics and Security*, Foz do Iguacu, Brazil, 2011.

[31] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *Proc. 12th Int. Conf. Information Security and Cryptology (ICISC'09)*, Berlin, Heidelberg, Germany, 2010, pp. 229–244, Springer-Verlag.

[32] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider, "Modular design of efficient secure function evaluation protocols," in *IACR Eprint Archive*, 2010.

[33] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," in *Proc. Int. Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT'01)*, London, U.K., 2001, pp. 280–299, Springer-Verlag.

Zekeriya Erkin (M'07) received B.Sc. and M.Sc. degrees in computer engineering from Istanbul Technical University, Istanbul, Turkey, in 2002 and 2005, respectively, and the Ph.D. degree in secure signal processing from Delft University of Technology, Delft, The Netherlands, in 2010.

He participated in the EC-funded FP6 FET project Signal Processing in the Encrypted Domain (SPEED). He was a short-term visiting researcher with Aarhus University and University of California Irvine in 2009 and 2011, respectively. His research interests involve watermarking, steganography, and privacy protection in online social networks, trusted healthcare systems, and smart metering systems. He is currently a postdoctoral researcher in the Information Security and Privacy Laboratory, Delft University of Technology.

Thijs Veugen received the M.Sc. degree from the Centre for Mathematics and Computer Science (cryptographic protocols for electronic cash under the supervision of David Chaum), and the Ph.D. degree in the field of information theory from Eindhoven University of Technology.

After that he worked as a scientific software engineer at Statistics Netherlands. He is a senior scientist in the Information Security research group of the Technical Sciences Department of TNO, Delft, The Netherlands, and senior researcher in the Multimedia Signal Processing group of Delft University of Technology. Since 1999, he has worked at TNO in information security, and specialized in the field of applications of cryptography. He published many scientific papers and patents, and acts frequently as member of the program committee of conferences and workshops. Within TNO, he is responsible for the knowledge strategy on information security, and works on privacy and identity management and computing with encrypted data.

Tomas Toft received the Ph.D. degree in 2007 from Aarhus University for the dissertation "Primitives and Applications for Multiparty Computation."

Since then, he has worked at CWI and Technische Universiteit Eindhoven, The Netherlands. He is currently employed at Aarhus University where he holds a postdoctoral position. His work focuses on the high-level aspects of secure multiparty computation, both theoretical and practical. In particular, he focuses on constructing specialized protocols for well-motivated applications as well as primitives easing the constructions of those applications.

Reginald L. Lagendijk (S'87–M'90–SM'97–F'07) received the M.Sc. and Ph.D. degrees in electrical engineering from Delft University of Technology, Delft, The Netherlands, in 1985 and 1990, respectively.

He was a Visiting Scientist in the Electronic Image Processing Laboratories, Eastman Kodak Research, Rochester, NY, in 1991 and Visiting Professor at Microsoft Research and Tsinghua University, Beijing, China, in 2000 and 2003, respectively. He has been a consultant with Philips Research Eindhoven from 2002 to 2005. Since 1999, he has been full professor at Delft University of Technology in the field of multimedia signal processing, where he holds the chair of Information and Communication Theory. The fundamental question that he is interested in is how multimedia information (images, video, audio) can be represented such that it is not only efficient in communication bandwidth or storage capacity, but that it is also easily identified when stored in large volumes (video libraries, internet) or transmitted over networks (e.g., P2P networks), that it is robust against errors when transmitted, that it can be protected against unauthorized usage, and that it has a good (audio-visual) quality. Research projects he is currently involved in cover subjects such as multimedia security (fingerprinting, watermarking, secure signal processing), multimedia information retrieval, and distributed signal processing. In the past he was involved in research on image sequence restoration and enhancement, 3-D video, and video compression. He is author of *Iterative Identification and Restoration of Images* (Kluwer, 1991) and a coauthor of *Motion Analysis and Image Sequence Processing* (Kluwer, 1993) and *Image and Video Databases: Restoration, Watermarking, and Retrieval* (Elsevier, 2000). He has been involved in the conference organizing committees of ICIP2001, 2003, 2006, and 2011.

Prof. Lagendijk was a member of the IEEE Signal Processing Society's Technical Committee on Image and Multidimensional Signal Processing. He was Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON SIGNAL PROCESSING'S SUPPLEMENT ON SECURE DIGITAL MEDIA, and IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY. Professor Lagendijk is elected member of the Royal Netherlands Academy of Arts and Sciences (KNAW).