Privacy-Preserving Recommender Systems in Dynamic Environments

Z. Erkin [#1], T. Veugen [#*2], R. L. Lagendijk [#3]

[#] *Information Security and Privacy Lab, Delft University of Technology, Mekelweg 4, 2628 CD, Delft, The Netherlands*
[1] z.erkin@tudelft.nl
[3] r.l.lagendijk@tudelft.nl

[*] *TNO, 2600 GB, Delft, The Netherlands*
[2] t.veugen@tno.nl

***Abstract*—Recommender systems play a crucial role today in on-line applications as they improve the customer satisfaction, and at the same time results in an increase in the profit for the service provider. However, there are serious privacy concerns as such systems rely on the personal data of the customers. There have been several proposals to provide privacy in recommender systems and, among many others, cryptographic techniques provide effective ways of protecting privacy-sensitive data of the customers. Unfortunately, existing methods only consider a static environment with constant number of customers in the system, which can be abused to extract more information on the customers when a cryptography based protocol is executed repeatedly. In this paper, we provide a privacy-preserving recommender system for a dynamic environment, which is more suitable for the real world applications.**

I. Introduction

E-commerce has exhibited phenomenal growth in the last decade [1]. According to analysts, the Internet presents a great place for customers to make a good deal as it is quite easy to compare prices from several retailers. To further increase their revenue, retailers have been successful in *personalizing* purchases by focusing on individuals rather than crowds. In particular, customer profiles are created and shopping patterns of customers are collected to be used in smart algorithms that generate a set of products, which are likely to be purchased by a target customer.

Among many algorithms, collaborative and content-based filtering techniques [2] have been proven effective in generating accurate recommendations for the customers. While collaborative filtering is based on similarity computations using ratings of multiple customers, content based filtering techniques are based on information on the items. In other words, a recommendation is generated for a particular customer by observing the characteristics of the previously purchased products [3]. Content-based recommendation is by far the most used recommendation system in practice and used by well-known providers such as Amazon.com, where each of user's purchased and rated items are matched to similar items rather than matching the user to similar customers [4].

WIFS' 2013, November 18-21, 2013, Guangzhou, China.
ISBN 978-1-4673-5593-3 ©*2013 IEEE.*

To improve the prediction accuracy, the retailers collect as much customer data as possible. While the benefits of personalized recommendations for the customers and the business are obvious, the collected data also create serious privacy risks for the individuals [5]. The service provider can easily identify and track individuals, especially when the collected data are combined with other publicly available resources, process the data for other purposes, transfer or sell them to third parties, or fail to provide adequate physical security. Consequences of either case will severely damage the privacy of the customers.

The privacy aspects of recommender systems in online services [5] have been investigated increasingly in the literature since [6]. Among many different approaches like data perturbation [7], distributed profiles [8], differential privacy [9], and agent-based approached [10], cryptography based techniques offer provable privacy by protecting the customer data, which can be in the form of profiles, ratings, and preferences, by means of encryption [11], [12]. While the data is kept secret from the service provider, it is still possible for the service provider to perform its usual tasks without accessing the private content [13].

The cryptographic methods proposed so far in the literature rely on different cryptographic tools like homomorphic encryption and multi-party computation techniques [14]. While the main consideration is the protection of privacy-sensitive user data from the malicious adversaries and the service provider, the main bottleneck of the proposals has been the efficiency; since the data is encrypted using a public-key cryptosystem, such as the Paillier scheme [15], there is an expansion in the size of the data, which introduces extra burden for the storage and the transmission of the data. Moreover, realizing the same service on the encrypted data is not trivial due to the limitations of the cryptosystems being used, and multi-party computation techniques are costly as they require interaction for operations, which are usually considered insignificant in the plain text domain [13].

Furthermore, the capabilities of the adversary play an important role in the design of the cryptography based protocols. A number of previous work is built on the semi-honest model, which assumes that computationally involved parties are honest enough to follow the described protocol [16].

Others also considered malicious model, which required to deploy more sophisticated cryptographic tools such as zero-knowledge proofs to prevent cheating [11].

Regardless of the cryptographic tools and the security model, existing works focus on a static environment, where the number of users do not change. While the protocols are proved to be secure, we argue in this paper that running the *privacy-preserving* protocol repeatedly on different number of users leaks information. This idea first addressed in [17] becomes very true for online services such as recommender systems since the number of users in the system is quite dynamic.

In this paper, we propose a recommender system in a setting, where the number of users change and the private data are protected from the adversaries and the service provider by means of encryption. To the best our knowledge, our proposal is the first recommender system in a dynamic environment. This is an important extension because it is very unlikely that in practice all users will be available during each run of the protocol. In particular, we also rely on homomorphic encryption and multi-party computation techniques, but in addition we propose a new approach to reduce the information that the service provider can extract about the users over a number of consecutive runs of the protocol.

We rely on a system with two servers and multiple users as in [12], which provides privacy for a static environment with constant number of users. The motivation for having a second server can be justified as follows: To provide services with privacy protection, the service provider might seek help from a semi-trusted external entity. This can be even required by law and regulations, if the service is based on medical data of the users (vital signs or medical record) such as self-helping groups for chronically ill or elderly people. Furthermore, having two servers delivering the recommendation service fits very well in a distributed server model.

The outline of the paper is as follows. We describe generating recommendation based on collaborative filtering, privacy requirements and assumptions in Section II. We present the privacy-preserving version of the recommender system in Section III along with the sub-protocols. We provide a complexity analysis and security discussion in Section IV. Finally, we draw conclusions in Section V.

II. Recommender Systems and Assumptions

A. Collaborative Filtering

A recommender system is a mean to personalize a service, where a specific user is given a number of recommendations that might be of interest to him or her. To generate such recommendations, different techniques exists, one of which is collaborative filtering [2]. In collaborative filtering, similar other users are found based on their preferences and weighted scores are computed among these similar users as the recommendations for the user who seeks recommendations. Let S be the number of user preference values used for computing the user similarities. More precisely, the similarity score between user A and user B is defined as follows:

$$sim(A,B) = \frac{\sum_{s=1}^{S} p_{A,s} \times p_{B,s}}{\sqrt{\sum_{s=1}^{S} p_{A,s}^2 \times \sum_{s=1}^{S} p_{B,s}^2}} , \qquad (1)$$

where $p_{A,s}$ and $p_{B,s}$ are the preference values of user A and B, respectively. It is assumed that the S dimensional vector consisting of the user's preferences is reflecting the taste of the users so that they can be used for the similarity score calculation. The recommendations for a set of M items are, in practice, generated from the rating of users, whose similarity score is above a certain threshold t. Let N be the total number of users. Suppose user A is requesting a recommendation. Then the recommendation for item m, $1 \leq m \leq M$, is computed as follows:

$$Rec_m = \frac{\sum_{n=1, sim(A,n)>t}^{N} r_{n,m}}{\sum_{n=1, sim(A,n)>t}^{N} 1} , \qquad (2)$$

where $r_{n,m}$ denotes the n-th user's rating for item m.

B. Privacy Requirements and Assumptions

In our setting, we aim for hiding users' preferences and ratings from the service provider, which consists of two servers. Furthermore, the similarity scores and the final recommendations are also kept secret from the service provider. The two server model paves the way for a natural business model in which one of the servers acts as the commercial service provider trying to recommend items to users. The other server acts as a privacy service provider which role could be fulfilled by a governmental organisation, but also by a commercial company selling privacy services. The two servers strenghten each other commercial or societal propositions, but their conflicting goals prohibit them from colluding.

We assume a semi-honest security model with two non-colluding servers. Each server has a key pair for an additively homomorphic cryptosystem like Paillier [15],which allows us to add plaintext messages by operating on their encryptions:

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(m_1) \times \mathcal{E}_{pk}(m_2)) = m_1 + m_2 . \qquad (3)$$

where m_1 and m_2 are the plaintext messages. Consequently, the following property also holds:

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)^c) = m \times c . \qquad (4)$$

The Paillier scheme is probabilistic as there is a fresh random parameter introduced in the encryption function. We refer readers to [15] for details on the Paillier cryptosystem.

For simplicity, we use $[\cdot]_i$ to denote an encryption with the public key of Server i, $i = 1, 2$. Although all cipher text computations are performed modulo n^2, where n is the Paillier modulus, we omit the "$\bmod n^2$" addition after each operation to increase readability.

Let N be the total number of users although this number may vary over different recommendation requests. However, we assume that the variation in N over different executions of the recommendation protocol is limited in the sense at least half of the users do not change during a considered set of

executions (see Subsection IV-B for a formalization of this notion). We assume the users normalized their preferences and rounded them to the nearest integer after scaling [12].

Users are allowed to collude with one server which also incorporates the case of a malicious server that creates dummy users in an attempt to learn preference values or ratings of other users. Different users might collude with different servers. Besides protecting the preference values and user rating from both servers, we also want to avoid users (possibly colluding with a server) to learn preference values and ratings of other users, by comparing different recommendation results over time in a dynamic user setting.

We assume the similarity threshold t, which is explained hereafter, is chosen such that the amount of (randomly chosen) similar users is sufficient for computing a sensible recommendation value.

The table below depicts all variables that are used in our paper.

N	total number of users
n	user n, $1 \leq n \leq N$
A	the user requesting for a recommendation
S	number of preference values per users
$p_{n,s}$	preference value number s of user n
M	the number of recommended items
$r_{n,m}$	rating of user n for item m
Rec_m	recommendation value for item m
U	the number of (randomly chosen) users involved in the recommendation computation
U_i	ith users participating in the computation
u	participating user u, $1 \leq u \leq U$
$[\cdot]_i$	an encryption with the public key of server i, $i = 1, 2$
t	threshold used in the recommendation computation
$sim(A, n)$	the similarity value between user A and user number n
ℓ	bit-length of similarity values
δ_n	the comparison bit that indicates whether $sim(A, n) > t$ or not
π	the permutation chosen by server 1 which swaps the N users
e_n	a bit chosen by server 2 which determines the participation of user $\pi(n)$ in the current recommendation

III. Privacy-Preserving Recommender System

It is evident that even if a privacy-preserving protocol for generating recommendations is invoked repeatedly with the changing number of users, the difference in the recommendations can leak information. For example, assume one additional user entered the system, then a difference in the recommendation (requested by the same user) would reveal whether this new user is similar or not to the requesting user.

More formally, we consider a recommender system with different type of users: *static* users that are involved in every execution and *dynamic* users that are absent at least in one execution. Consider that the recommender system, even a privacy preserving version, is invoked several times, computing results $Rec_1, Rec_2, \ldots$. Clearly, the execution of any privacy-preserving protocol designed for a static setting reveals information on the private data of the dynamic users, as this information can be inferred from the set of participating users in those executions and the outcome. For example, after two executions with different number of users, by comparing Rec_1 and Rec_2, one can obtain information on $r_{n,m}$ (or even disclose that value). This scenario is defined as the *new group attack* in [17].

To overcome this problem, we propose to involve a subset of users from the set of static and dynamic users, that are randomly selected in each execution. Several approaches to choose a subset of entities randomly are studied considering different security models in [17] and we adopt this idea to realize a complete recommender system in this paper.

The main idea is to generate a secret binary value $e_n \in \{0, 1\}$ for every user n such that $\sum_{n=1}^{N} e_n = U$. This bit determines whether the n-th user will be participating in the computation or not. Determining the exact value of U depends on the system parameters, that is it should be large enough to generate accurate recommendations but smaller enough than N to protect the privacy of the users: A recent analysis shows that $U \approx N/2$ is optimal when the majority of users is static [17]. Note that in recommendation systems it is common practice not to involve all users in the similarity computation due to performance issues. Therefore, the reliability of the result as perceived by users is not decreased because the recommendation result is allowed to change over time.

Based on the above approach, we now describe the protocol. We assume that a user, noted as A, seeks a recommendation for item m, $1 \leq m \leq M$.

A. Main Protocol

Initialization: All users n, $1 \leq n \leq N$, encrypt their preferences $p_{n,s}$ for $1 \leq s \leq S$ and their ratings $r_{n,m}$ for $1 \leq m \leq M$ using the public key of Server 2 and send them to Server 1.

1) Server 1 and 2 run a sub-protocol, defined in sub-protocol III.1, to compute the encrypted similarity scores between user A and all the other users in the system: $[sim(A, n)]_2$. Only Server 1 obtains the encrypted similarity scores, which are encrypted with the public key of Server 2.
2) Server 1 and 2 run a comparison protocol (sub-protocol III.2) that compares each similarity score with a threshold t. Before running the comparison protocol Server 1 hides the order of the users by a random permutation π. The protocol works in such a way that Server 2 obtains the (permuted) comparison bits $\delta_{\pi(n)}$ that are encrypted with the public key of Server 1, without either of them being able to access them.

3) Server 2 generates random bits, e_n, for each (permuted) user such that there will be U users involved in total (sub-protocol III.3).
4) Server 2 multiplies the permuted similarity bits and the random bits to obtain $[\delta_{\pi(n)} \cdot e_n]_1$, which equals either the encrypted comparison result bit if $e_n = 1$ or an encrypted zero, otherwise (sub-protocol III.4).
5) The encryption of $\delta_{\pi(n)} \cdot e_n$ held by Server 2 is transferred to Server 1 in a sub-protocol such that Server 1 obtains $[\delta_{\pi(n)} \cdot e_n]_2$, which is encrypted by the public key of Server 2 (sub-protocol III.5).
6) Server 1 and 2 run the secure multiplication protocol for each user n to compute $[\delta_{\pi(n)} \cdot e_n \cdot r_{\pi(n),m}]_2 := [\delta_{\pi(n)} \cdot e_i]_2 \otimes [r_{\pi(n),m}]_2$, where $\otimes$ is the secure multiplication protocol (sub-protocol III.6). Server 1 obtains the result.
7) Server 1 adds up all the computed values (sub-protocol III.7).
8) The recommendation Rec_m is given to the user through a decryption protocol [12] (sub-protocol III.8).

The above protocol achieves the goal of hiding preferences and ratings from the servers as discussed in Section IV-B. At the same time, similarity scores and the involved parties are kept secret as well. The protocol relies on a number of sub-protocols that we prefer to explain in the following section.

B. Sub-protocols

Protocol III.1 *Computing Similarities.* Server 1 has the encrypted preference values for all users and preference items: $[p_{n,s}]_2$ for $1 \leq n \leq N$ and $1 \leq s \leq S$. Server 1 and 2 compute

$$[sim(A,n)]_2 = \prod_{s=1}^{S} [p_{A,s} \cdot p_{n,s}]_2 = \prod_{s=1}^{S} [p_{A,s}]_2 \otimes [p_{n,s}]_2 \,, \quad (5)$$

for each n which involves S invocations of the secure multiplication protocol.

Protocol III.2 *Comparison Protocol.* In Step 2, the permuted similarity scores are compared to the threshold t. To achieve this, we rely on the protocol described in [18]. Server 1 inputs the encrypted similarity values $[sim(A, \pi(n))]_2$. The output bit $\delta_{\pi(n)}$ should be obtained by Server 2, encrypted with the public key of Server 1.

For the sake of clarity, we briefly summarize the protocol in [18]. Given two encrypted values $[a]$ and $[b]$ of length ℓ bits, the protocol outputs the most significant bit of the value $z = 2^\ell + a - b$ as the comparison result λ, where th $\lambda = 1$ if $a > b$ and $\lambda = 0$, otherwise. Since only encrypted values of a and b are in hand, computing the most significant bit of $[z]$ is achieved as follows:

1) Server 1 computes $[z] := [2^\ell] \cdot [a] \cdot [b]^{-1}$.
2) Server 1 generates a random number r that is at least κ bits larger than z and adds this value to z: $[c] := [z] \cdot [r]$. Server 1 sends c to Server 2.
3) Server 2 decrypts $[c]$.

Note that the most significant bit of z is $\gamma = (z - z \bmod 2^\ell) \cdot 2^{-\ell}$. Server 1, now, can compute the most significant bit as follows: $[\gamma] := [z - (c \bmod 2^\ell - r \bmod 2^\ell) \bmod 2^\ell]$ since $z \bmod 2\ell = c \bmod 2^\ell - r \bmod 2^\ell$. However, the last reduction in not trivial to achieve in the encrypted domain: it requires to compare $c \bmod 2^\ell$ and $r \bmod 2^\ell$. For this, Server 1 and Server 2 continues with the following steps:

1) After decrypting $[c]$, Server 2 encrypts the first ℓ bits of c separately and sends them to Server 1.
2) Server 1 computes $[e_i]$ for $0 \leq i \leq \ell - 1$:

$$[e_i] := [s + c_i - r_i + 3 \sum_{j=i}^{\ell-1} c_j \oplus r_j] \,, \quad (6)$$

where $s \in \{-1, 1\}$ determines the direction of the comparison: $c > r$ or $c < r$.
3) Server 1 permutes the $[e_i]$ values and send them to Server 2.
4) Server 2 decrypts the values and check if there is a *zero* among them. Depending on that, it sets the value of γ to 0 or 1, and sends $[\gamma]$ to Server 1.
5) Server 1 corrects the direction of γ using s.

As the output bit of the protocol from [18] is obtained by the same server that holds the inputs, we need a small modification. In this protocol Server 1 chooses $s \in \{-1, 1\}$ indicating the type of comparison (larger or smaller than). Server 2 learns a bit γ that indicates whether one of the received encryptions is zero or not. Since the output bit is the exclusive-or of bit $(s + 1)/2$ and γ, Server 1 will be able to compute the output $[\delta_{\pi(n)}]_2$ from $[\gamma]_2$, which can be sent by Server 2. This modification preserves all privacy requirements.

Protocol III.3 *Choosing a random subset of users.* Server 2, who does not know which value belongs to which user, is allowed to choose a random subset of users of size U. This is done by generating N bits e_n such that $\sum_{n=1}^{N} e_n = U$. The preferred way to achieve this is to uniformly choose one of the $\binom{N}{U}$ subsets and consequently set the bits e_n according to the random choice.

Protocol III.4 *Limit the number of similarity bits.* Once the bits e_n are generated, Server 2 can easily multiply them with the previously received comparison bits $[\delta_{\pi(n)}]_2$. This is performed by replacing the encryption by a fresh encryption of 0 in case $e_n = 0$, and a rerandomization of the encryption (without altering the plain text value) otherwise.

Protocol III.5 *Switching the crypto system.* Server 2 obtained the encryptions $[\delta_{\pi(n)} \cdot e_n]_1$ which we need to transfer to Server 1, encrypted with the public key of Server 2. A small subprotocol is needed for this end.

1) Server 2 generates a large random number ρ_n to additively blind the value $\delta_{\pi(n)} \cdot e_n$.

2) Server 2 computes $[\delta_{\pi(n)} \cdot e_n + \rho_n]_1$ and sends it, together with $[-\rho_n]_2$, to Server 1.
3) Server 1 decrypts $[\delta_{\pi(n)} \cdot e_n + \rho_n]_1$, encrypts it with the public key of user 2, and use the homomorphic property to obtain $[\delta_{\pi(n)} \cdot e_n]_2 = [\delta_{\pi(n)} \cdot e_n + \rho_n]_2 \cdot [-\rho_n]_2$.

Protocol III.6 *Secure multiplication.* Server 1 and 2 run a secure multiplication protocol such that the encryptions $[\delta_{\pi(n)} \cdot e_i]_2$ and $[r_{\pi(n),m}]_2$ held by Server 1 are multiplied without leaking any information to either server. One such protocol can be found in [12].

Protocol III.7 *Accumulation over all users.* Server 1 holds $[\delta_{\pi(n)} \cdot e_i \cdot r_{\pi(n),m}]_2$ and $[\delta_{\pi(n)} \cdot e_i]_2$ for each user n and can add them together due to the homomorphic property of the crypto system.

Protocol III.8 *Secure decryption.* User A should obtain its recommendation value Rec_m which is the division of $\sum_{n=1,\delta_{\pi(n)} \cdot e_i=1}^{N} r_{n,m}$ and $\sum_{n=1,\delta_{\pi(n)} \cdot e_i=1}^{N} 1$. Server 1 currently holds these encryptions. To decrypt these two values without either Server learning the results we need a small decryption subprotocol. To obtain $\sum_{n=1,\delta_{\pi(n)} \cdot e_i=1}^{N} 1$ for example the following steps are needed:

1) User A generated a large random number ρ_A, encrypts it with the public key of Server 2, and sends $[\rho_A]_2$ to Server 1.
2) Server 1 adds it to $[\sum_{n=1,\delta_{\pi(n)} \cdot e_i=1}^{N} 1]_2$ using the homomorphic property of the crypto system and sends it to Server 2.
3) Server 2 decrypts it and sends $\rho_A + \sum_{n=1,\delta_{\pi(n)} \cdot e_i=1}^{N} 1$ to Server 1, who passes it to user A.
4) User A subtracts its random number ρ_A and obtains the denominator of the recommendation value.

The numerator is similarly obtained by user A, who performs a divison in the plain domain to obtain Rec_m.

IV. Analysis

A. Complexity Analysis

In this section, we provide the complexity analysis of the proposed protocol in terms of operations on the encrypted data in Table I. These operations are encryption, decryption, multiplication and exponentiation, including reductions by the modulus of the cryptosystem, which is a large number in the case of the Paillier cryptosystem. Compared to operations on the encrypted data, plaint text operations have negligible costs, thus they are not considered in the analysis.

As it is clear from Table I, the computational burden on the users is not heavy: they only need to encrypt their private preferences and ratings once. On the other hand, Server 1 and Server 2 are engaged in heavy computations compared to the users. This is mostly due to the secure multiplication protocol and the comparison protocol. However, note that the encrypted values in the computations are mostly binary and therefore, the operations are not expensive. For example, ℓ is usually in the range of 15-20 bits. In terms of data transmission, the costs can be reduced further by applying data packing [19], [20].

TABLE I
Complexity of the proposed protocol.

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

Compared to [12], our proposal has a comparable complexity. Note that we have not considered optimizations like changing encryption schemes for a sub-protocol in comparisons and data packing. The difference in the complexity is due to the generation of a random vector e and additional secure multiplication protocols to multiply each user data with the corresponding random value e_i.

B. Security Analysis

The proof that our solution really fulfills the privacy requirements consists of two parts.

First of all, we have to show that the entire protocol, including all sub-protocols is secure in the semi-honest model. The main setting is similar to the one used in [12], where Server 1 plays the role of Service Provider and Server 2 is the Privacy Service Provider (PSP). The arguments given in [12] also explain why the protocol is secure in the semi-honest model. Most of our sub-protocols are similar to the ones used there, except for Sub-protocol III.5 where we transfer an encryption held by Server 2 to an encryption held by Server 1. However, the techniques used in this Sub-protocol, namely additive blinding and (homomorphic) encryption, ensure that the value $\delta_{\pi(n)} \cdot e_n$ never leaks to either server. The reader that is interested in formal security proof techniques is referred to [21]. The proof given there is easily extended to our protocols.

Secondly, we have to show that no personal information is leaked from the recommendation values of consecutive executions of our protocol with different users. Since a recommendation output unavoidable leaks *some* information about other user's preferences or ratings, a more precise statement is needed than "no personal information is leaked".

The concept of choosing a random subset of involved users during each execution originates from a recent paper by Kononchuk *et al.* [17] concerning group services. The collaborative filtering based recommendation service is such a group service because its output is not affected by the order of the users. We adopt their definition of dynamic privacy by distinghuising between dynamic and static users. Let's consider n executions of the recommendation protocol. Then the static users are the users that are present (but not necessarily involved) in all n executions. The dynamic users are the remaining users. The set of n executions is considered private as long as, given the n recommendation values, the uncertainty (in terms of entropy) about the personal

information of each dynamic user is at least the uncertainty about the personal information of a static user. In [17] is shown that this condition holds as long as the majority of users is static. In our setting this is true by assumption, but it also seems quite realistic in a system with a large amount of users.

Given that (a set of) recommendation values reveals 'no' private information, it also follows that collusions between users and a single server do not pose extra threats to our system. Servers do not possess relevant additional information, because each execution is secure in the semi-honest model.

V. Conclusion

In this paper, we presented a cryptographic protocol to generate recommendations without revealing privacy-sensitive customer preferences and ratings. While our work is not the first one that tackles this problem using cryptographic techniques, it is the first one that addressed the dynamic nature of such systems, where the number of people to be included in the collaborative filtering procedure changes continuously. In particular, our proposal works on a random subset of users that are chosen in a secret way without degrading the performance of the recommender system. The building blocks for the proposal are designed to be as efficient as possible in terms of computational complexity. The analysis shows that within the 2-server model, the required computations are acceptable.

Acknowledgement

This publication is supported by the Dutch national program COMMIT.

References

[1] L. Indvik, "Forrester: E-commerce to reach nearly $300 billion in U.S. by 2015," http://mashable.com/2011/02/28/forrester-e-commerce/, February 28 2011, online.

[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 Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, June 2005.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *WWW '01: Proceedings of the 10th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2001, pp. 285–295.

[4] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76 – 80, jan/feb 2003.

[5] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis, "Privacy risks in recommender systems," *IEEE Internet Computing*, vol. 5, no. 6, pp. 54–62, 2001.

[6] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, vol. 29(2). ACM Press New York, NY, USA, 2000, pp. 439–450.

[7] H. Polat and W. Du, "Svd-based collaborative filtering with privacy," in *Proceedings of the 2005 ACM symposium on Applied computing*, 2005, pp. 791–795.

[8] 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*. New York, NY, USA: ACM, 2009, pp. 157–164.

[9] F. McSherry and I. Mironov, "Differentially private recommender systems: building privacy into the net," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2009, pp. 627–636.

[10] 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*. New York, NY, USA: ACM, 2007, pp. 1–8.

[11] J. Canny, "Collaborative filtering with privacy," in *IEEE Symposium on Security and Privacy*, 2002, pp. 45–57.

[12] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1053–1066, 2012.

[13] R. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection," *IEEE Signal Processing Magazine*, January 2013 2013.

[14] O. Goldreich, *Foundations of Cryptography I*. Cambridge University Press, 2001.

[15] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology, EUROCRYPT – 99*, 1999, pp. 223–238.

[16] M. Beye, Z. Erkin, and R. Lagendijk, "Efficient privacy preserving K-means clustering in a three-party setting," in *IEEE Workshop on Information Forensics and Security (WIFS'11)*, IEEE. Foz do Iguaçu, Brazil: IEEE, 11/2011 2011.

[17] D. Kononchuk, Z. Erkin, J. van der Lubbe, and R. L. Lagendijk, "Privacy-preserving user data oriented services for groups with dynamic participation," in *ESORICS*, ser. Lecture Notes in Computer Science. Springer, 2013, pp. 418–442.

[18] Z. Erkin, M. Franz, S. Katzenbeisser, J. Guajardo, R. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *9th Symposium on Privacy Enhanced Technologies (PETs)*, Seattle, USA, August 2009, pp. 235–253.

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

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

[21] T. Veugen, "Encrypted integer division," in *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, dec. 2010, pp. 1–6.