International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 05 | May 2021 www.irjet.net p-ISSN: 2395-0072

An Efficient Blockchain-based Privacy-Preserving Collaborative Filtering Architecture

Kavery P Uthaman[1], Ajeesh.S[2], Dr. Smita C Thomas[3]

[1]M.Tech Student, APJ Abdul Kalam Technological University, Kadammanitta, Kerala, India
[2]Assnt.professor, Mount Zion College of Engineering, Kadammanitta, Kerala, India
[3]Associate Professor, Department of Computer Science and Engineering, Mount Zion College, Kadammanitta, Kerala, India

---

Abstract - *Information overload is a phenomenon of our days due to the unprecedented penetration of information and communication technologies (ICT) in our daily lives. As a result, people often end up with more options than they can process to choose from and therefore may opt for choices which do not fit best to their preferences. To address these issues, recommender systems (RSs) were proposed and have gained a lot of interest from the research community and industry. However, privacy is a big concern in these systems. While decentralized recommenders can protect privacy, they lack the needed efficiency to be widely adopted. In this article, we use blockchain as the backbone of a decentralized RS, managing to equip it with a broad set of features while simultaneously, preserving user's privacy. We introduce a new architecture, based on decentralized locality sensitive hashing classification as well as a set of recommendation methods, according to how data are managed by users. Extensive experimental results illustrate the performance and efficacy of our approach compared with state-of-the-art methods. In addition, a discussion about its benefits and opportunities provides ground for further research*

Key Words: Blockchain, collaborative filtering (CF) decentralized file storage, privacy, recommender systems (RSs).

1. INTRODUCTION

OUR society lives in an age where the eagerness for information has resulted in problems such as infobesity, especially after the arrival of Web 2.0. In this context, automatic systems such as recommenders are increasing their relevance, since they enhance data management and help to distinguish useful information from noise. In principle, recommender systems (RS) are data-driven algorithms that assist users in selecting item(s) from a larger set. RS play an active role on the Internet through the advances in data mining and artificial intelligence. Their role is considered critical in online business as they connect users with potential purchases, content, and other services. Collaborative filtering (CF)is a type of RS that comprises a large family of recommendation methods. CF aims to suggest/recommend items; e.g., books, films, routes, based on the preferences of users that have already acquired and/or rated some of those items.

Nowadays, data management is facing a paradigm change due to big data challenges, increased regulations and client demands as well as to its inherent scalability and security problems when real-time services are required. In this context, blockchain-based solutions seem to be the most promising opportunity to solve many of such issues and requirements due to their properties, such as security, verifiability, robustness, and availability. Blockchain is a distributed peer-to-peer linked structure, that was initially introduced by Satoshi Nakamoto back in 2008 [6] to maintain the order of transactions and avoid the double-spending problem in Bitcoin. This way, the blockchain structure manages to contain a robust and auditable registry of all transactions.

we may categorize blockchains in three generations. More concretely, Blockchain 1.0 includes applications that enable digital cryptocurrency transactions. Blockchain 2.0 includes smart contracts and a set of applications that are more extensive than simple cryptocurrency transactions. Finally, Blockchain 3.0 includes applications in areas beyond the previous two versions, such as big data management, governance, health, science, or Internet of Things (IoT).

1.1 Motivation

Traditional RS methods have inherent problems in their implementations, and therefore, current architectures do not allow them to reach their full potential. Privacy considerations are the most profound ones. Undoubtedly, there has been a lot of work in this direction. Nevertheless, the proposed solutions do not manage to solve many issues. For instance, many privacy solutions for RS depend on centralized schemes, in which data are stored, raising privacy concerns. In the case of decentralized schemes, availability is critical in decentralized schemes (as well as in centralized approaches, since the majority of computations are performed on the server-side), since intermediate data (e.g., similarities) are not stored, and the quality of recommendations highly depends on the number of participants. The latter means that in decentralized architectures, there is a tradeoff between communication costs and quality of recommendations, hindering scalability.

However, the recent introduction of blockchain technologies has paved the way to create a highly efficient distributed infrastructure capable of providing many features, which serve the backbone for various and heterogeneous applications ranging from supply chain management to identity management, finance and cryptocurrencies or IoT . In this regard, several authors have proposed architectures that use blockchain to enable secure and verifiable data sharing in contexts like health and the Internet of Things.

Following this train of thought, we extend the use of RS by utilizing blockchain technology. The distributed nature of blockchains and their efficiency may serve as the needed skeleton for a distributed architecture with well-known efficiency and efficacy. Nevertheless, as it will be discussed in the following sections, by adopting the blockchain technology one may significantly improve RS as it enables various features including timeliness, availability and improved privacy, among others. These theoretical features are then validated through our experiments that highlight that the proposed architecture is both practical and efficient. In addition to these advantages, blockchain paves the way for fully distributed, verifiable and auditable RSs, features that cannot be guaranteed in centralized nor decentralized architectures

1.2 Main Contributions

We aim to present an architecture that advances traditional decentralized systems in terms of communication and computational costs. Despite the contradictions that one may consider when discussing efficiency and blockchain, the overhead that blockchains imply is in the transactions and their verification. However, the actual overhead of RS is in the computations and the needed communications between peers to collect the data in the decentralized setting. To bridge both worlds, we exploit the combination of an efficient locality sensitive hashing (LSH)- based bucketization/clustering with blockchain, described in Section IV-B. Moreover, the addition of permanent decentralized storage systems such as InterPlanetary File System (IPFS), minimizes the information that needs to be stored and shared in the blockchain. Thus, blockchain enhances decentralized RSs as follows:

1) The hash function used to perform an LSH-based clustering of profiles can be shared on the blockchain. Thus, clustering is made off-line by each user and has a negligible cost.
2) We cluster profiles without communication costs, some- thing infeasible in the case of classical decentralized RS.
3) The information of users is stored off-chain, so that the transaction information is minimal, requiring only hashes to be shared.
4) Further computations are only performed between users of the same cluster, therefore avoiding all-to-all computations , such as in the case of well-known decentralized RSs. More on the efficiency of our proposal is described in Section VI-B.
5) We describe three methods with different characteristics to showcase the possible configurations and variants of our architecture, in terms of what information is shared in the blockchain.

2. RELATED WORK

The widespread use of RS on the Internet provides great opportunities and benefits for both companies and users, but there is a major drawback: the lack of users' privacy. Careless management of personal information, besides being against the legislation in most countries, could lead to serious consequences for both users, whose information is stored, as well as companies. Current legislation requires service providers to handle data properly and ensure users' privacy.

However, there are many examples of errors that have led to the disclosure of private information and, hence, it is apparent that legislation alone cannot solve the problem [30]. Consequently, the task of finding and repairing the security and privacy weaknesses of RSs is an urgent need. This requires a multidisciplinary approach, which combines expertise in various fields, including privacy engineering, RSs, data mining, etc.

In the CF field, we need to tackle specific issues related to privacy. For instance, customers, who believe that their preferences/profiles may be exposed, would not rate an item or, rate it incorrectly or distorted. This user behavior, derived from the feeling of lack of privacy, results in a reduction of both the number of ratings as well as their quality. Another drawback is that companies can acquire data about the preferences of many users in a specific domain/market and obtain a significant advantage over new competitors if they decide to expand to other markets. Moreover, the existence of large monopolies on the Internet (e.g., Google and Amazon) is another clear disadvantage, so users' data could be transferred amongst different parties, which are managed by the same large companies, without the users' awareness. In the long term, this lack of security and privacy may strongly damage businesses as well as society as a whole. To address the privacy issues raised by the systematic collection of private information, which is required for the proper use of CF, current research focuses on privacy-preserving collaborative filtering (PPCF) methods . Such methods may be classified in centralized and decentralized as seen in. In terms of decentralized PPCF approaches, many examples can be found in the literature . Nowadays, the adoption of blockchain creates a new paradigm in RSs and CF literature, since they enable solutions that were infeasible in the past due to its inherent characteristics. Nevertheless, the symbiotic relationship between blockchains and RSs has yet to be explored. Only a few examples can be found in the literature, with the one of Frey et al being the most well-known. In such work, authors propose a blockchain-based approach to address privacy concerns in RSs utilizing the blockchain infrastructure. Authors claim that they provide cryptographic

guarantees to customer's privacy in RSs, since raw data are only accessible by the owner. Nevertheless, they use blockchain as a black box and do not perform experiments regarding the efficiency and accuracy of such architecture. In another work, Frey et al. propose an e-commerce solution based on a shopping system in which data and transactions use the framework schemed . Therefore, existing blockchain-based RSs mainly exploit the secure and verifiable storage of blockchain while maintaining communication and computation costs. On the contrary, our system goes one step beyond current state of the art and intro duces an architecture that enables efficient and fully distributed blockchain-based recommendation computation, outperforming traditional decentralized RSs. Moreover, we perform experiments in terms of efficiency and accuracy, as well as an extensive study of the different parameters of our methodology.

Finally, our system is enhanced by the inherent char-acteristics of blockchain, such as anonymity, security, availability,1 and verifiability.

We also want to stress that the efficiency of our proposal is unrelated to the underlying consensus mechanisms of the blockchain. The efficiency stems from the fact that with the use of the blockchain, we store and cluster the data so that they can then be used appropriately. Moreover, we have a stable and robust structure that nodes can trust and avoid costly communication data exchanges.

We assume that the blockchain enables the embodiment of enough information(about bytes inside each block or transaction) in a reasonable time. Note that RS databases are updated at fixed intervals of time (e.g., weekly, fortnightly) or when there is a high number of new user inputs in an attempt to find a reasonable tradeoff between accuracy and computational cost. Nevertheless, the transaction limitation size is overcome in this article through the use of decentralized permanent storage. In terms of scalability, recomputing all user-to-user similarities or applying dimensionality reduction methods like singular value decomposition (SVD) or principal component analysis (PCA) in real-time does not scale, especially for high-volume databases. On the contrary, our experiments and integration in a real blockchain show the feasibility of our approach and scale more efficiently than traditional RS systems.

2.1 Organization of this Work

The rest of this article is organized as follows. In the next section, we present the related work, mainly focusing on decentralized privacy-preserving approaches in CF. Then, we present our architecture and detail the different approaches that can be fostered and what features are gained in each case. Afterward, we present some experimental results and compare them with other relevant literature and discuss several aspects separately. Finally, the article concludes summarizing our contributions and discussing ideas for future work.

3. BACKGROUND

3.1 InterPlanetary File System

The InterPlanetary File System is a decentralized P2P system for retrieving and sharing IPFS objects. Instead of identifying objects by their location (e.g., HTTPS), IPFS uses a content-oriented addressing structure [i.e., a cryptographically authenticated Merkle directed acyclic graph (DAG)] in which objects are represented by their Base58 SHA-256 encoded hash.2 IPFS implements an improved and adaptable version of P2P data transfer protocol (i.e., compared with similar systems like BitTorrent), namely BitSwap, which enables built-in storage marketplaces like Filecoin.3 The use of a Merkle DAG structure allows the creation of a version control system (VCS). More concretely, IPFS stores the object history so that all versions are accessible throughout time. This permits configurable synchronisation of files since all users can edit them locally and later push the new files to IPFS. VCS is enhanced by the InterPlanetary Naming System (IPNS), which enables content linking, so that files can be accessed using the node ID address, allowing users to retrieve updated contents without knowing the new hashes of the files they request. In addition to these features, IPFS enables a series of properties described briefly in Table I. Nowadays, IPFS is being exploited in multiple contexts . Moreover, the symbiotic relationship between IPFS and blockchain is further enhancing the possibilities of such systems by enabling off-chain storage and anonymous file sharing.

Table -1: Main properties and characteristics of IPFS

Property	Description
Equality	Nodes have equal permissions and possibilities
Decentralization	A distributed network without central entities
Immutabilities	Content based addressing guarantees that each file resolves to a specific hash
Fault tolerance	Decentralization and peer participation guarantees the robustness of the network
Availability	Multiple nodes guarantees the availability of the network and not one single entity.
Resilient to attacks	Persistence is guaranteed by distributed manner
Unlimited resources	A high number of simultaneous usere sharing their assets
Scalability	Request are made to the closet peer instead to a central location
Market place monitisation	Storage incentivisation through systems like filecoine

3.2 Locality Sensitive Hashing

The primary goal of locality sensitive hashing is to represent data items with an arbitrary number of features/characteristics using constansized sets, namely signatures. Thus, each feature set is represented by a constant size signature that preserves the similarity between the data entries according to some estimator.

Among typical LSH estimators (e.g., bit-sampling for Hamming and l1-norm, min- hash for Jaccard coefficient or other norms) there exist binary LSH schemes, which involve efficient and scalable bit- wise computations. Such binary solutions are suitable for large- scale applications such as pattern recognition, big data analytics or Rss. Our solution is based on the superbit LSH, which is a binary LSH method that guarantees an unbiased estimate of angular similarity. Moreover, unlike other LSH methods, hashes computed with the superbit LSH not only

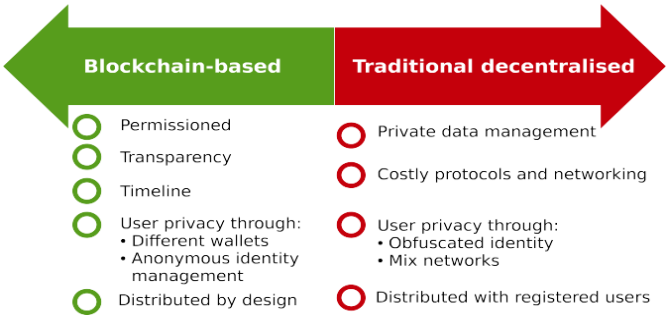

Fig -1: Comparison between main characteristics of blockchain-based and traditional decentralized architectures.

Permit a fast and accurate approximation of cosine similarity but also enable data clustering by design, since hashes can be easily mapped into a predefined number of buckets (i.e., bucketization). Therefore, similar entries are classified/clustered into the same bucket.

4. OUR APPROACH

As already discussed, the use of a private permissioned blockchain as the backbone of the RS implies that the derived RS will be distributed by design and highly efficient. Based on that, our approach comprises a two-step procedure. First, we use a bucketization step, which significantly enhances the efficiency of our methods when combined with blockchain. Second, the recommendation computation which has three variants, respectively. The main benefits and differences between traditional decentralized RSs and blockchain are summarized in Fig. 1. Moreover, private blockchains minimise the risk of fake profiles (and thus, sybil attacks) as well as shilling attacks, according to some participation rules.

4.1 Concept and Framework Definition

The core concept of our approach is summarized as follows. First, all the needed information that users must have is pushed in the genesis block of the blockchain. Therefore, the genesis block contains the definition of the LSH function that is going to be used to group users, and the parameters of the bucketization step, which are the hash function and the total number of buckets (i.e., the number of groups to which the hash function can map the inputs). The LSH function and the parameters used are described with details in Section IV-B. There are further parameters that can be considered, such as the number of stages, which is used to split the input data and process it independently. Such a technique increases the efficiency of similarity detection search processes. However, in our method, we are interested in correctly grouping users. Moreover, the use of more than one stage may have consequences in terms of privacy protection that are discussed in Section V.

Once the users have this information, they may apply it to their own ratings vector and find the bucket number or the

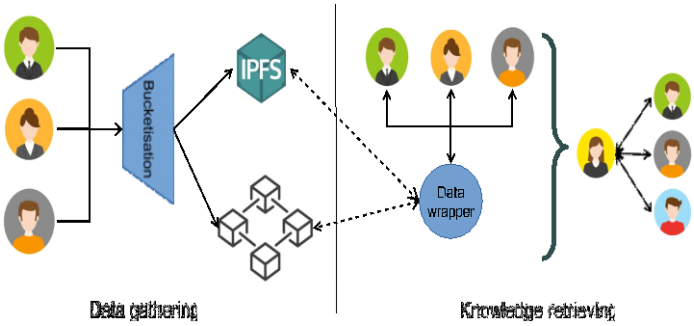

Fig -2: General overview of our framework.

cluster they belong to and update the blockchain accordingly. Instead of sending their complete profiles to the blockchain, users submit the hash of their profiles (pointing to an IPFS or IPNS location), the selected method (between the options later described in Section IV) and the corresponding bucket. This way, our platform manages to overcome the transaction size limitation. Thereafter, a data wrapper collects data pinned by users (i.e., uploaded and distributed over IPFS) so that we have an ordered and clean representation of the users. Note that all the information can be checked in the blockchain and therefore, data tampering can be easily and efficiently detected. Additional information can be uploaded on IPFS, specifying user-to-user communication details, according to each recommendation method. Finally, users can apply one of our three methods and compute recommendations using the information on the blockchain. A general overview of our framework is depicted in Fig. 2.

Once a user collects her data and performs the bucketization procedure, her results are stored in blockchain and IPFS. Later, the data wrapper collects all the so that users can compute recommendations according to each scenario. Our goal is to create a new architecture that preserves the benefits of blockchain and enhances the recommendations. Therefore, we aim toward an efficient and decentralized privacy- preserving RS.

Our work differentiates from the current state of the art in several ways. First, our scheme does not follow a centralized architecture; nevertheless, it provides higher efficiency and similar accuracy of such a system. Second, contrary to common distributed RSs, it is scalable and provides the inherent benefits of blockchain. To the best of our knowledge, this is the first blockchain-based approach that implements an efficient, scalable and accurate RS using blockchain infrastructure.

4.2 Superbit Bucketization Procedure

To predict whether an item would please a given user, CF methods rely on large databases with information regarding the relationships between sets of users and items. These data take the form of R→ p×q rating matrices, which are composed of p users and q items, and each matrix cell ru,i stores the rating of user u on item i. User's ratings belong to some finite set S such as {1, 2, 3, 4, 5}. Given a ratings matrix with empty cells, the first step is to fill them to accurately compute the LSH hashes and avoid biased bucketization. After considering well-known imputation techniques, in this article, we used the default voting,

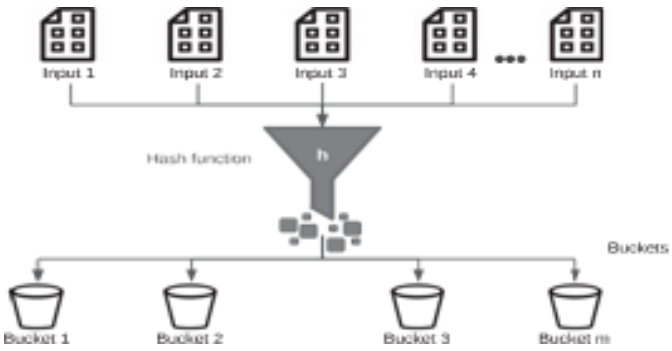

Fig -3: Bucketization procedure. A set of n input vectors are clustered in m buckets, according to their hash.

which is a classical imputation method that replaces missing values using a static value (e.g., the central value of the possible ratings). Due to the multiple possibilities and their complexity, a study of how different imputation techniques may affect our methods is left to future work.

Thereafter, we generate an LSH structure with the superbit methodology and hash each user ratings profile pi (i.e., each row of the R→ p×q matrix to their corresponding bucket) so that similar users are classified/clustered into the same bucket.

An overview of the bucketization procedure is depicted in Fig. 3. The bucket number depends on data and provided configuration. The number of buckets must be selected carefully depending on the hash function and data distribution. The number of stages, which defines how many times we split pi to compute hashes, is selected to be one to avoid potential attacks (see Section V). The bucketization procedure is described in Algorithm 1.

We use a private permissioned blockchain to write all operations. As stated in Section IV, the genesis block (i.e., the first block of the blockchain) contains the initialisation information and the hash function as the number of stages, buckets, and dimensionality of data. Note that such information can be easily updated in the future by adding a new block with the required information in the blockchain. Moreover, users recompute their bucket if there are new ratings in their profile.

In this work, we developed three recommendation strategies:

1) public memory-based, in which users locally compute recommendations using the distorted information shared in the blockchain,
2) private average computation, in which users of the same bucket securely compute its average vector to obtain recommendations, and
3) private memory-based, in which users compute secure and private κ nearest neighbors (κNN) with the users of the same bucket.

The main characteristics of each recommendation strategy, as well as details of the information stored in the blockchain, are summarized in Table 2. In all cases, we assume that users always store their own ratings vector. This way, we prevent disclosure to third parties and dishonest behaviors of centralized schemes, which may entail security and economic issues.

4.3 Public Memory-Based Method

First, we consider the case where the users upload a modified version of their preferences to the blockchain.

Table -2: Main characteristics of the recommendation strategies.

property	Public Memory-based	Private Average computation	Private Memory-based
Information in blockchain	• Bucket • Hash	• Bucket • Hash	• Bucket • Hash
Method	Fully customisable	Secure average computation	Secure similarity computation
Computation cost	O((n/b)^2)	O(()n/b)^2	O((n/b)^2)
Privacy	Noise addition	K-anonymity(k>=3)	Fully private

Variable b denotes the number of valid buckets. Although theoretical, computational costs are similar. The secure multiparty computation protocols applied in the methods described in Sections 3.4 and 3.5 increase their security to the detriment of their communication costs.

Algorithm 1: Superbit LSH Bucketisation.

1: **function** HASHINPUTDATASET (DataSet D, Integer n_stages, Integer n_buckets, Integer data_dimensionality)
2: S = GenerateSuperbitStructure(n_stages, n_buckets, data_dimensionality);
3: **while** (DatasetHasRecords) **do**
4: pi = SelectTheNextRecord (D);
5: hi = ComputeHashesRecord (pi, S); D Compute Hash of each input
6: **end while**
7: H = MapSignatures (h1 ... hp, S);D Map hashes into buckets
8: return H D MappedSet H
9: **end Function**

this approach, namely Gaussian noise addition (GNA) approach is to perturb the numerical values of users' ratings vectors using a Gaussian distribution with zero mean and standard deviation σ (i.e., (0, σ)). Note that the higher the σ value, the greater the range of the generated values, (i.e., it is more likely to generate values close to the boundaries of the value range). Nevertheless, we propose a decentralized approach using blockchain, which increases the privacy and efficiency of computations compared with similar PPCF methods implemented using centralized or decentralized P2P architectures, as further discussed in Section VI-B2.
We may also use a discrete uniform distribution to substitute rare values or values with too few observations by other real values present in the dataset.

Laplacian noise is also widely used in the literature because of its interesting properties. A simple way to hide a number a is to add a random number r to it. Although we cannot do anything to a since it is distorted, we can perform certain computations if we are interested in the aggregated data, rather than in each individual data.

By adding random noise to the data, the data are perturbed/obfuscated so that certain computations can be performed while preserving users' privacy. Clearly, privacy is preserved as an adversary cannot tell whether a specific value is valid or noise. While information from each individual user is distorted, if the number of users is significantly large, the aggregated information of these users can be estimated with decent accuracy. Such property is useful for computations that are based on aggregate information. For instance, the scalar product and random sum are among such computations and are widely used in the literature. Despite that, our aim is to protect the real values of users while generating meaningful recommendations. Therefore, users store in the blockchain their pseudonym, an obfuscated version of their ratings vector and their bucket number. Note that the obfuscation value may be selected according to a parameter to modify the tradeoff between privacy and recommendation's quality. Despite the fact that users may generate a new pseudonym each time they publish their information to reduce the risk of profile linking attacks, this feature could enable other disadvantages such as Sybil attacks, discussed later in Section V. Therefore, we do not contemplate this option in our design. Notwithstanding, user's bucket changes will be used to keep track of their profile evolution.

Since all needed information is publicly available, users can use a plethora of methods to compute recommendations locally. For the sake of clarity, we selected a κ-NN approach using inner-bucket similarities.

4.4 Private Average Computation Method

this case, the information shared by users is their pseudonym and bucket number. A secure multiparty computation is performed between the members of the same bucket to compute the average vector, which will be used as a recommendation. Nevertheless, the use of blockchain increases the efficiency of computations, compared with other centralized and decentralized PPCF clustering approaches. For privacy reasons, users belonging to buckets with cardinality k < 3 (we consider k more than two since data will be disclosed if k 1 users collude) will be remapped into other buckets. In this sense, one of our future research lines is to explore higher cardinality values (i.e. k > 3) and study the bucket distributions as well as accuracy in such cases. Nevertheless, in this article, we establish k =3 as the lower bound.

To securely compute the average vectors, we use additive homomorphic encryption. For instance, a scheme with semantic security was given by Paillier. However, creating the common key and hiding the values of individuals without a semitrusted third party becomes an issue. Therefore, more specific and efficient private aggregation approaches can be used . Users may use the protocol described in so that they connect with a server by using a secure channel and compute the aggregate value of their evaluations. In fact, the existence of the blockchain facilitates such protocols in publishing their intermediate results. Therefore, users can also use blockchain-enabled secure multiparty computation systems like Enigma, which provides users with end-to-end decentralized apps using private contracts with which share their data with specific purposes (i.e., storing their rating profiles in a secure way but enabling secure multiparty computations) without disclosing them.

Once computations are finished, the average vector of each bucket is shared in the blockchain with a timestamp.

Therefore, each user stores her own ratings vector and only averages are revealed.

4.5 Private Memory-Based Method

In this case, users store their pseudonyms and bucket numbers in the blockchain, as in the methodology described in Section. This time, however, we perform decentralized memory-based recommendations. Users of the same bucket use a secure multi- party protocol to compute similarities. Note that the number of computations is significantly lower when compared with the whole dataset, thanks to the LSH bucketization step described in Section IV-B. Therefore, by using LSH and blockchain, we can compute much more efficiently the recommendations than other state-of-the-art decentralized systems.

After computations are performed, users will have a list of their neighbors so that they can ask for recommendations on a given item. For instance, ua may ask her closest user ub for a recommendation on item ic. If ub does not have a real rating/recommendation for that item, ua will ask her next closest user till she finds a recommendation for that item. Note that the number of recommendations/queries to users will be limited to avoid users disclosing too much information. Furthermore, users may obfuscate their ratings locally to avoid privacy disclosure.

5. ATTACK VECTORS AND COUNTERMEASURES

In this section, we discuss possible attack vectors and the countermeasures implemented in our architecture. Therefore, we demonstrate the overall robustness and privacy protection of our architecture in addition to the concrete privacy level achieved by each of our implemented methods, described in Table II. In this regard, our methods obfuscate evaluation profiles in such a way that it is not possible to reidentify users corresponding to any particular record in the anonymized published dataset (i.e., identity disclosure) nor discover the value of a confidential attribute for a specific respondent (i.e., attribute disclosure). In this article, we consider all user's evaluations as confidential attributes.

Although this identity disclosure can be a configurable measure in the case of our first method (i.e., public memory-based) depending on the amount of Gaussian noise added to data (and, therefore, users can be reidentified if data are not properly obfuscated), the other two methods compute averages and recommendations in a private way. Therefore, the private average computation achieves k-anonymity, and thus protects attribute and identity disclosure. In the case of the private memory-based method, no data are exposed, so that we achieve even higher privacy protection than in the other cases.

To quantify the privacy offered by our methods, we use the disclosure risk (DR) as a privacy measure. The DR measures the probability of correctly relating a record of the anonymized dataset with a record of the original dataset. It is also known as the probability of reidentification, or the reidentification risk.

For an attacker, the reidentification procedure consists of computing the similarities (e.g., using the Euclidean distance) between a given anonymized record ra, and the target records rt, that could be obtained from third-party sources. In this case, we consider the worst-case scenario and assume that the attacker has access to the raw dataset. Therefore, the attacker will try to link each record from the anonymized dataset with the raw dataset.

Table -3: dr percentage for the analyzed datasets, parameters and methods

Method	Dataset	σ=0.25	σ=0.5	σ=1	σ=1.5	σ=2
B_kNN_GNA	Movielens 100K	100	95.31	53.18	33.29	24.39
B_kNN_GNA	Movielens 1M	100	90.13	38.97	25.97	21.25
B_AVG b=300	Movielens 100K			11.78		
B_AVG b=2000	Movielens 1M			13.08		

Hence, to compute the DR, we try to reidentify all records and then compute the percentage of correct reidentifications. In the first case, we compute the DR using different levels of noise addition, denoted by σ, ranging from 0.5 to 3. In the case of the average computation, we use the worst-case scenario implemented in our system (i.e., the one which presents more accuracy, since privacy will be lower in such case due to higher similarity between profiles), which corresponds to b = 300 in the case of Movielens 100 k and b = 2000 in the case of Movielens 1 M (see Section VI). The results of these computations are depicted in Table III.

In the case of B_KNN_GNA method, we observe that the values of DR decrease as the value of σ is increased. Nevertheless, the more obfuscation, the less accurate the recommendations would be. In regard to the B_AVG method, k-anonymity guarantees that the DR is upper bounded, that is DR 100/k. Moreover, the bucket cardinality distribution is also related to the level of privacy, as later described in Section VI.

The ease with which large third parties and companies collect, analyze and correlate data is undermining the user's privacy in multiple digital scenarios. These practices are usually paired with issues such as security breaches, data leakages and others, which might compromise users' privacy. Nevertheless, according to state of the art, there are several privacy-preserving proposals for public blockchain such as well as identity platforms such as uPort or Sovrin,

which propose enhanced decentralized ledgers that provide users with mechanisms to preserve their privacy in their digital transactions. In the case of private blockchains, there exist diverse ways to protect the privacy of the users. For instance, Hyperledger uses channels and private transactions to protect the confidentiality of the information available in the ledger. Moreover, it implements a zero-knowledge asset transfer, an advanced mixing mechanism, which is built on top of Identity Mixer, an anonymous authentication protocol. In the case of Ethereum, uPort enables the creation of segregated device keys to store the operations made by a certain user while protecting her privacy. In addition, a more generic framework for anonymous identity management in permissioned blockchains can be found in. In parallel, other privacy-preserving cryptographic systems are integrating self-sovereign identity in the blockchain through anonymous credential systems, as well as other solutions, which implement secure multiparty computation, and zero-knowledge proofs. Therefore, while there are mechanisms to protect the user's identity in blockchain, we focus our work on providing an efficient framework to enable recommendation computation between users, leaving this part of the system as a

Therefore, given a user ua in bucket bat an attacker may try to find the most similar user profile ub in bat so that he can infer that ua = ub. In this regard, several mechanisms prevent the attacker from obtaining such information:

1. The bucket information of a user will not be updated if there are no changes, avoiding that a user bucket profile information has the same value repeated in adjacent time intervals.
2. We can ensure that a user is always posting the same distorted vector if she did not make changes, to reduce DR.
3. GNA is applied at each iteration so that vectors are distorted in a different way and considering that users are clustered, the possibility of disclosure is vastly reduced.

Another similar attack can be performed with the profile evolution information of a user. This time, if we analyze ua bucket history, we may see that she was in bucket bat and then, after updating her profile, her bucket changed to bbt . An attacker may try to compute similarities between the user profiles in both buckets, to link the random pseudonyms generated at each iteration with the general pseudonym of a user and obtain information about her raw data. However, since LSH sends similar profiles to similar buckets, the possibility of disclosure is similar to the one of the previous attacks. In the case of the method described in Section 3.5, users could suffer from multiple query attacks. There are several countermeasures for that, such as limiting the maximum number of queries per user (or in a general setup) to avoid disclosure, as seen in other well-known approaches. Another option is obfuscating values after n responses, using well-known mechanisms. Such approaches can be implemented locally by the interested users, or we can use a variable (e.g., number of queries) paired with an ID in a mapping structure to perform a simple check. Stronger policies such as increasing the GAS cost (the internal pricing for running a transaction or contract in Ethereum) for users that perform repeated queries is also an option that can be adopted easily in blockchain, which can be used as a trust feature. Note that reputation systems are more difficult to implement in other decentralized designs since they require further computations to privately exchange information from users to keep track of a model representing the trust values between users. Therefore, the aforementioned measures can help to overcome shilling (biased profiles that may increase the popularity of an item) and Sybil (multiple user profiles to tamper with reputation and bias rating values). Note that our system partially deals with shilling by design, since the

Table -4: Main characteristics and features of the literature methods used in our comparison.

Method	Description	Parameters
kNN	Compute k most similar users and use their data to compute predictions	Number of neighbours k
SGD	Interactive method for minimizing an objective similarity function	reg=0.02 learning_rate=0.005,n_epochs=20
ALS	Alternative optimization for the 1.2 norms	λi=10, λu=15, n_epochs=10
NMF	Non negative low-rank matrix factorization	N_factors=15,n_epochs=50, λu=0.6, λi=0.6

LSH clustering will probably classify shilling/biased profiles into outlier clusters. Finally, we select only one stage for the Superbit structure to avoid brute force attacks. More precisely, since LSH functions are not cryptographically secure, having many stages would allow an adversary to brute force possible errors (known from the genesis block) to small vectors allowing her to recover the original values of the user. Having only one stage and a rating vector guarantees that this attack will not be possible. Moreover,

DATASET	BUCKET
Movielens 100k	50/100/150/200/250/300/350
Movielens 1m	250/500/750/1000/1500/2000 /2500

the more stages, the more distinguishable users are, since each user belongs to a particular and probably unique list of buckets, which increases the disclosure of the aforementioned brute force attack.

6. EXPERIMENTAL SETUP

To assess the quality of our methods, we use two well-known CF benchmarks. Movielens was developed by Grouplens, and is one of the reference sets in CF. The two most prevalent variants of such dataset are Movielens 100 k and Movielens 1 M. Movielens 100 k contains 100 000 ratings of 943 users on 1682 films. In the case of Movielens 1 M, such dataset contains 1 million ratings performed by 6040 users on 3900 films. In both cases, range values are comprised between 1 and 5. Both datasets are highly sparse since more than 90% of the fields are empty. First, we implement and test the methods B_KNN_GNA, B_AVG, and B_KNN proposed in Sections IV-C, IV-D, and IV-E, respectively. Next, we compare our approaches with a set of well-known state-of-the-art methods: 1) the κ-NN approach with κ = 1, 5 and 10, 2 two baseline regularization methods, namely stochastic gradient descent (SGD) and alternating least squares (ALS), the nonnegative matrix factorization approach (NMF). A summary of the state-of-the-art methods and their main characteristics is pro- vided in Table 4. The methods SGD, ALS, and NMF have been implemented using a Python library. In the case of our approaches, we follow a preprocessing step in which bench- mark datasets are filled with the center of their corresponding value range, to alleviate sparseness and to compute the superbit similarities accurately. Finally, the B_KNN_GNA approach uses (0, σ) to obfuscate data with σ = 0.25. We split our experiments into three sections. First, we assess the feasibility of our method and the implementation details for a real blockchain architecture in Section VI-A. Second, we perform an efficiency analysis of our LSH methodology for each benchmark dataset. Finally, we compute the accuracy of recommendations for our proposals and compare them with the aforementioned method.

Table -5: Bucket's values been computed according to the number of users present in each corresponding dataset.

6.1 Integration

In principle, storing data directly to the blockchain is not efficient. The reason is that blockchains have been designed to store transactions and not arbitrary data. Therefore, in typical scenarios, one would use something like IPFS4, Storj5, or TiesDB6 and push the hash identifier to the blockchain. This option allows one to store arbitrary data in the database efficiently.

In each transaction, users append some metadata which is the hash of their ratings, see Table II. Once a user wants to compute some recommendations, he collects the corresponding transactions and extracts the metadata. This allows the user to collect all the ratings stored off-chain and, based on the collected data. The user may apply one of the three proposed approaches detailed in Section IV. It has to be highlighted that the coin mining procedure can be set to a really low threshold. This allows users to commit their ratings with negligible cost.

In terms of performance, in our experimental setup running on an Intel i7-4710MQ with 8 GB RAM server on Ubuntu 16.04 LTS an Ethereum blockchain7 each user transaction took on average 5.158 ms. Moreover, data retrieval from IPFS was also performed in the order of milliseconds between different peers located in different countries. Bucket's values been computed according to the number of users present in each corresponding dataset.

6.2 Efficiency Analysis

We adapted the LSH superbit implementation of to our CF methods as an efficient similarity computation module. The number of stages was always 1 (as discussed in Section V) and the bucket configurations tested for each dataset are summarized in Table V. In Section VI-B1 we analyze the bucketization outcomes to observe how data are mapped into different buckets, we study the efficiency of neighborhood computations performed by our blockchain-based κNN methods, compared with the state of the art.

6.2.1 Bucketization Performance

Table VIa and b describe the LSH bucketization mapping outcomes for Movielens 100 k and

[4][Online]. Available: https://ipfs.io/

[5][Online]. Available: https://storj.io/

[6][Online]. Available: https://tiesdb.com/

[7][Online]. Available: https://geth.ethereum.org/

Table -6: Results of the bucketization step of our clustering method

a) Movielens 100K

Buckets	ABC	σ	Mapped	valid
50	48.78	54.73	58%	38%
100	25.91	40.33	80%	34%
150	14.49	37.95	82%	39.3%
200	6.16	5.02	96%	69.5%
250	8.75	10.39	60.4%	39.6%
300	4.91	2.07	91%	51.3%
350	5.63	4.13	82%	36%

ABC stands for average bucket cardinality.

b) Movielens 1M

Buckets	ABC	σ	Mapped	Valid
250	36.41	64.17	76.4%	66%
500	21.45	22.94	71%	55%
750	8.34	3.36	99.6%	95%
1000	6.59	2.81	99.4%	88.7%
1500	4.94	1.69	97.7%	73.66%
2000	4.31	1.41	93.65%	56.2%
2500	4.26	1.46	85%	41.32%

Table -7: Average results of the bucketisation step of our private and public kNN blockcahin-based methods for movielens 100 K

Buckets	Avg neighbours	σ	% Re-Mapped users	% Relative comparisons
50	107.35	85.04	0.53	11.40
100	82.16	55.05	3.18	8.73
150	102.43	114.72	4.24	10.88
200	8.49	9.26	2.22	0.89
250	18.54	24.60	2.96	1.97
300	3.99	2.68	5.51	0.43
350	5.91	6.33	9.33	0.63

Table -8: Average results of the bucketisation step of our private and public kNN blockchain-based methods for movielens 1M

Buckets	Avg.Neighbours	σ	%Re-Mapped users	%Relative comparisons
250	147.75	159.53	0.34	2.45
500	44.29	44.33	0.99	0.73
750	8.63	3.49	0.11	0.12
1000	6.60	3.26	0.44	0.11
1500	4.16	2.08	2.41	0.07
2000	3.17	1.88	5.09	0.06
200	2.93	2.02	9.10	0.05

Movielens 1 M, respectively. Tables contain the average of users per bucket and the corresponding standard deviation σ, the % of mapped buckets, which are buckets that have at least one user and the % of valid buckets, which are buckets that contain k => 3 users and thus form a cluster.

In general, as expected, the average bucket cardinality decreases when the number of buckets is increased. However, we may observe cases in which the % of valid buckets decreases, and thus, the cardinality of buckets is increased even when the number of buckets is increased (cf. Table VIa, b = 250).

For example, users that were mapped into different buckets when b = 200, may be mapped into the same bucket when b = 250, and users that belong to the same bucket when b = 200 may be mapped into different ones when b = 250, creating small clusters with different cardinalities.

Therefore, since the input vectors and the LSH parameters directly affect group creation, a previous methodology to identify proper bucketization parameters (e.g., optimal number of buckets and number of stages) depending on the dataset characteristics would be desirable and is left to future work.

6.2.2 Neighborhood Computation Performance

Tables VII and VIII show the results of the tested configurations and contain: 1) the average of neighbors of each user and its corresponding standard deviation, 2) the percentage of users that were mapped alone into a bucket, and 3) the relative percentage of similarity computations needed to create the neighborhoods compared with the ones needed between all users of the dataset. For instance, if the value of the relative comparison is 1%, this means that we create the neighborhoods avoiding/saving a 99% of computations and thus, the lower, the better. Note that users that are alone in a bucket are remapped to the closest nonempty bucket to enable the recommendation computation. Therefore,our method may also be useful for outlier detection. However, parameters such as the number of buckets must be tuned for each dataset and, thus, it is left for future work.

As observed in Section VI-B1, the data distribution and the number of buckets affect the neighborhood creation. Note that the average neighbors' value is weighted by buckets with high cardinality. Although the average neighbors decrease when the number of buckets is increased in the case of Movielens 1 M (cf. Table VIII), we may observe a different behavior in the case of Movielens 100 k (cf. Table VII). For example, we obtain higher cardinality with b = 150 than with b = 100, which means that a high number of users that belong to different buckets when b = 100 were mapped into the same bucket when b = 150. Moreover, the σ value when b = 150 is higher than the one obtained with a lower number of buckets and, thus, the cardinality of neighborhoods is less stable. Finally, we may also observe that the % of relative comparisons value is related to the average neighbors value, because the more users per bucket, the more similarity computations need to be performed. In all cases, such % of (especially when the number of buckets is high) to compute the user's neighborhoods compared with traditional methods.

6.3 Recommendation Computation

We compute the error between the original dataset values and the predicted values using the mean absolute error (MAE) metric, defined as follows:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |p_i - r_i|}{n}$$

Table -9: MAE outcomes of our approaches for the movielens dataset.

a) Movielens 100K

Number of	B_KNN		B_KNN_GNA		B_AVG	
Buckets	5-Fold	10-Fold	5-Fold	10-Fold	5-Fold	10-Fold
50	0.896	0.880	0.936	0.930	0.898	0.885
100	0.840	0.835	0.892	0.884	0.881	0.874
150	0.850	0.848	0.908	0.907	0.843	0.831
200	0.935	0.929	0.982	0.980	0.781	0.778
250	0.912	0.907	0.959	0.953	0.816	0.811
300	0.899	0.896	0.956	0.944	0.778	0.765
350	0.843	0.838	0.903	0.890	0.802	0.796

b) Movielens 1M

Number of	B_KNN		B_KNN_GNA		B_AVG	
Buckets	5-Fold	10-Fold	5-Fold	10-Fold	5-Fold	10-Fold
250	0.906	0.894	0.942	0.940	0.903	0.894
500	0.912	0.899	0.957	0.942	0.897	0.878
750	0.974	0.972	1.014	1.012	0.840	0.935
1000	0.970	0.963	1.015	1.005	0.812	0.809
1500	0.953	0.944	1.004	0.994	0.789	0.771
2000	0.921	0.910	0.977	0.966	0.769	0.766
2500	0.930	0.914	0.981	0.967	0.790	0.779

Best configurations are highlighted in blue.

Chart -1: MAE outcomes comparison for Movielens datasets. The lower, the better. (a) Movielens 100 k dataset. (b) Movielens 1 M dataset.

where n is the number of predicted elements, p_i is the predicted value over the element i, and r_i is the real value of i. The lower the MAE value, the better the accuracy of the predictions. Note that recommendations are computed only for the original voted values of each user. For each method, we conducted a fivefold and tenfold test, repeated them 100 times and computed the average of the outcomes. First, we depict the outcomes of our methods for the bucket configurations showed in Table 5. Therefore, Table 9a and b show, for each benchmark dataset, the MAE outcomes for B_KNN, B_KNN_GNA, and B_AVG methods. Next, we select the best configuration and compare our methods with well-known state-of-the-art methods. The results of such comparison are depicted in chart 1(a) and (b).

7. DISCUSSION

The outcomes of the bucketization step, as discussed in Section VI-B1, vary depending on data distribution and the LSH function. For instance, we observe that the percentage of buckets that have at least one user is not related to the number of buckets. In the case of Movielens 100 k (cf. Table 6a), the more mapped buckets, the more valid buckets, being b = 200 the configuration that achieves a higher percentage in both cases. In the case of Movielens 1 M (cf. Table 6b), we achieve a higher percentage of mapped and valid buckets than with Movielens 100 k. The best value is achieved when b = 750. However, a high percentage of valid buckets does not mean high-quality recommendations, being the cardinality of such buckets, a measure that has much more impact, as observed in Section VI-C. One of the aims of our proposal is to achieve high efficiency and scalability while maintaining the privacy of users in a fully decentralized context (i.e., in the blockchain). In that sense, the results discussed in Section VI-B2 shows that we only need a few computations to obtain the neighborhoods of users. Moreover, similarities computed using superbit hashes are far less costly than a standard cosine similarity computation. In the case of the B_AVG method, note that we only need to compute the average vector between the users of a bucket to obtain recommendations and thus, the computational cost can be considered almost negligible.

The blockchain-based recommendation methods proposed have different characteristics in terms of data obfuscation as well as computational complexity. Nevertheless, such method selection benefits user personalization. Moreover, users can apply further configuration policies, for example, in the case of the B_KNN_GNA method our blockchain-based recommendation methods are depicted in Table IXa and b. In general, the results of B_KNN and B_KNN_GNA methods are better when the number of buckets is low, because the average cardinality of neighborhoods is higher, increasing the number of referrals. In the case of the B_AVG method, the lower the cardinality, the lower the error, which is normal behavior.

Finally, we compared our methods with other well-known approaches and depicted the outcomes in chart 1(a) and (b). In all cases, the highest error is obtained by κNN with κ = 1, which acts as an upper bound. In the case of Movielens 100 k, the outcomes of our B_KNN approach are close to the ones obtained by κNN with κ = 10, which highlights the quality of our approach [cf.chart 1(a)]. Moreover, we achieve such results with only an 8.73% of relative comparisons, in a decentralized privacy-preserving way. The MAE obtained by our clustering procedure B_AVG is slightly higher than the one obtained by the literature methods but better than all memory-based approaches. Moreover, note that the computational cost of the B_AVG is much lower than the cost of B_KNN.

The outcomes obtained for Movielens 1 M are depicted

in chart 1(b). The error obtained of our B_KNN approach is slightly higher than the one obtained with κNN with κ = 5. Despite that, it is much lower than the one obtained by κNN with κ = 1. In the case of the B_AVG method, the MAE value is slightly higher than the other literature methods, as with Movielens 100 k.

The MAE obtained by B_KNN_GNA is slightly worse than the one obtained by B_KNN in all cases. However, such error is slightly higher than the one obtained by the κNN method with κ =5 and far better than the one with κ =1 in both datasets. Moreover, the B_KNN_GNA method enables a totally customizable recommendation computation. For the sake of clarity, we used the κNN method, but any other state-of-the-art approach could be used to improve the outcomes. Therefore, more experiments using recommendation methods with obfuscated user vectors are left as future work.

As previously stated, the outcomes achieved by our blockchain-based approaches, especially the B_AVG method where they can obfuscate their data according to their privacy requirements. The outcomes of, are computed much more efficiently than its peers, even if they are implemented in a centralized setting. Therefore, our methods are more efficient (cf. Tables 7 and 8), preserve the privacy of users (cf. Table 2) as well as integrate the inherent benefits of blockchain architecture in a decentralized permanent storage, features which are not offered by the rest of state-of-the-art methods. In addition, blockchain's direction toward the use of sovereign identities and advanced identity management standards enables the use of smart contracts to provide anonymous-but-personalized services, which in the case of RSs may go from directed marketing to reward mechanisms, as well as further services delivered by third parties.

8. CONCLUSIONS

In this article, we introduced a new blockchain-based RS architecture. We use an accurate and efficient LSH implementation to map users into buckets and share such information in the blockchain. Next, we present three new privacy-preserving blockchain-based methods and compare them with other state- of-the-art methods. The results showed that we achieve almost the same accuracy than centralized approaches, but with much higher efficiency and using the fully decentralized architecture of a blockchain. Moreover, our approaches preserve the privacy of users, a feature not offered by the rest of the compared methods. Finally, we observed that the LSH function and the data distribution have a strong impact on our blockchain-based methods. Future work will focus on studying other LSH implementations, as well as finding a methodology to properly select parameters such as the number of buckets depending on the recommendation method and the imputation technique used to overcome sparseness. Moreover, we will study the addition of randomized response techniques and the enhancement of our obfuscation step using Laplacian noise, to achieve differential privacy. Finally, we will investigate more sophisticated and configurable architectures using smart contracts to enhance the possibilities of the framework.

REFERENCES

[1] P. Resnick and H. R. Varian, "Recommender systems," Commun. ACM, vol. 40, no. 3, pp. 56–58, Mar. 1997.
[2] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," Commun. ACM, vol. 35, no. 12, pp. 61–70, Dec. 1992.
[3] J. L. Zhao, S. Fan, and J. Yan, "Overview of business innovations and research opportunities in blockchain and introduction to the special issue," Financial Innov., vol. 2, no. 1, Dec. 2016, Art. no. 28.
[4] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: current status, classification and open issues," Telematics Inform., vol. 28, pp. 55–81, 2018.
[5] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," Future Gener. Comput. Syst., 2017.
[6] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[7] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn., Springer, 2015, pp. 281–310.
[8] E. Karydi and K. Margaritis, "Parallel and distributed collaborative filtering: A survey," ACM Comput. Surv., vol. 49, no. 2, pp. 37:1–37:41, Aug. 2016.
[9] A. Friedman, B. P. Knijnenburg, K. Vanhecke, L. Martens, and S. Berkovsky, Privacy Aspects of Recommender Systems. Berlin, Germany: Springer, 2015, pp. 649–688.
[10] K. Xu and Z. Yan, "Privacy protection in mobile recommender systems: A survey," in Security, Privacy, and Anonymity in Computation, Communication, and Storage, G. Wang, I. Ray, J. M. Alcaraz Calero, and S. M. Thampi, Eds. Berlin, Germany: Springer, 2016, pp. 305–318.
[11] A. Ozturk and H. Polat, "From existing trends to future trends in privacy- preserving collaborative filtering," Wiley Interdisciplinary Rev.: Data Mining Knowl. Discov., vol. 5, no. 6, pp. 276–291, 2015.
[12] F. Casino, C. Patsakis, D. Puig, and A. Solanas, "On privacy preserving collaborative filtering: Current trends, open problems, and new issues," in Proc. IEEE 10th Int. Conf. e-Bus. Eng., 2013, pp. 244–249.

[13] I. Mohallick, K. De Moor, Ö. Özgöbek, and J. A. Gulla, "Towards new privacy regulations in europe: Users' privacy perception in recommender systems," in Security, Privacy, and Anonymity in Computation, Communication, and Storage, G. Wang, J. Chen, and L. T. Yang, Eds. Berlin, Germany: Springer, 2018, pp. 319–330..