DocCloud: A document recommender system on cloud computing with plausible deniability

Juan Vera-del-Campo [a,⇑], Josep Pegueroles [a], Juan Hernández-Serrano [a], Miguel Soriano [a,b]

[a] Universitat Politècnica de Catalunya, Jordi Girona 1-3, C3, Barcelona, Spain
[b] Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), Av. Carl Friedrich Gauss 7, B4, Castelldefels, Barcelona, Spain

Keywords:
Recommender system
Doccloud
Social cloud
Plausible deniability

a b s t r a c t

Recommender systems select the most interesting products for costumers based on their interests. The move of a recommender system to a cloud faces many challenges from the perspective of the protection of the participants. Little work has been done regarding secure recommender systems or how to cope with the legal liability of the cloud provider and any virtual machine inside the cloud.

We propose DocCloud, a recommender system that focused on the protection of all participants against legal attacks. We present the architecture of DocCloud and analyze the security mechanisms that the system includes. Specifically, we study the properties of plausible deniability and anonymity of the recommenders and intermediate nodes. This way, nodes can recommend products to the customers while deny any knowledge about the product they are recommending or their participation in the recommendation process.

1. Introduction

Many of the most successful shops on the Internet provide suggestions to their customers. These suggestions are usually based on the previous experience of users with the shop, or the features of the products the customers are currently viewing.

A *recommender system* is an automatic system that, given a set of available products and a model that captures the interests of a user, outputs a list of products that the system estimates will be of interest to the user. We identify five different roles in a recommender system: *merchants* that provide documents and document descriptions, or *profiles*; *customers* that request recommendations according to their *user profiles*; *indexers* that reply to queries; *repositories* that provide access to final documents and *intermediate nodes* that route messages from merchants and customers to indexers and repositories.

We explore the recommendation of documents that can be distributed electronically and may show high peaks of activity followed by long periods of silence, such as video or audio content. The dynamic structure that cloud systems show makes them a convenient approach for providing the service of recommendation and distribution of multimedia content. However, regardless of the cloud paradigm that the recommender system follows, the participants of a cloud may be accused or prosecuted of copyright infraction.

In this paper, we will use the different roles that exist in a cloud system and other additional mechanisms to protect requests and recommendations, and minimize the legal responsibility that a recommender systems faces.

⇑ Corresponding author. Tel.: +34 699692582.
E-mail addresses: juanvi@entel.upc.edu (J. Vera-del-Campo), josep@entel.upc.edu (J. Pegueroles), jserrano@entel.upc.edu (J. Hernández-Serrano), soriano@entel.upc.edu (M. Soriano).

1.1. Attacks against a recommender system

Recently, a new kind of attack against intermediate nodes has appeared. A new international treaty on copyright protection known as the Anti-Counterfeiting Trade Agreement (ACTA) is being negotiated at the moment of writing this paper. After some initial secrecy, the consolidated text is now public [38]. Article 2.15 deals with the liability of legal persons, and states that ''the provisions of this section shall apply to *inciting, aiding and abetting* the offenses referred in article 2.14''[1]. The penalties proposed by this article ''include imprisonment as well as monetary fines''. Thus, not only is downloading or providing a protected document punishable under the ACTA, but so too is abetting downloading. ACTA was not ratified by the European Parliament in July, 2012 [24] However, it shows a trend among legislators throughout the world to make people who aid or even incite the downloading of copyrighted documents liable of copyright infringement. This is the case with the Stop Online Piracy Act (SOPA) and the Protect Intellectual Property Act (PIPA) in the United States, and the Sinde-Wert law in Spain. Most of these bills are currently under intense debate, but they have something in common with ACTA: they make companies liable for users' actions if the companies do not respond to a copyright infringement notification.

The reader should take note that the entity that makes documents available, the one that recommends documents, and the one that distributes them may be not the same. This way, a participant recommending or distributing a document may be unaware of its legal status. In this case, we believe that *recommending* is dangerously close to *inciting*, which is punishable according to ACTA. This situation is even worse for the recommender system if it includes mechanisms to upload and/or download the protected document.

Recently, the administrators of the direct download site Megaupload were arrested in New Zealand on behalf of the American FBI [40]. According to the FBI, this action ''directly targets the misuse of a public content storage and distribution site to commit *and facilitate* intellectual property crime''. In April, 2009, the administrators of The Pirate Bay, a popular document indexer that does not host any document, were found guilty of complicity in the provision of unauthorized access to copyrighted content and were sentenced to 1 year in jail and nearly 3 million euros in damages by a Swedish court [36]. The high cost of a lawsuit, the criminal charges against the defendants and the long wait time to get a ruling persuaded other site administrators (Filesonic, Fileserve, Uploaded.to, VideoBB, FileJungle, UploadStation, FilePost, UploadBox, x7.to, 4shared, etc.) either changed their policies or announced a voluntary shutdown [9].

Glorioso et al. [12] analyze some court decisions and conclude that, according to US jurisprudence, intermediate nodes in the communication channel, even at the network level such as ISPs, are threatened with legal attacks. The main argument for prosecutors is that the ''making available'' of copyrighted documents also includes the *indexers* of the system.

The security service that protects indexers from legal prosecution is plausible deniability. In the common law context, plausible deniability [41] refers to circumstances where a denial of responsibility or knowledge of wrongdoing cannot be proved as true or untrue due to a lack of evidence proving the allegation. Hermoni et al. [13] defined deniability for the field of information security as the property that the user has if they can claim that their actions are legitimate. Ref. [13] understands deniability as an absolute property that actors have by mixing up legitimate and not legitimate documents in the indexers. We introduce *plausible deniability* as the property that an actor achieves if he cannot make a probabilistic distinction between legitimate and not legitimate documents, or the process to make this distinction is too long and complex.

We focus our efforts on the protection of all participants of the system against legal attacks. As such, we are interested in the provision of plausible server deniability and document deniability as defined in [13], and we will provide plausible deniability to intermediate nodes and the cloud provider.

1.2. Our contribution

In this paper, we introduce DocCloud, a document recommender system offered by a service provider using the SaaS paradigm of cloud computing. This service involves running small virtual machines (PaaS/IaaS) that are controlled by individual users to actually provide recommendations. We offer protection in two layers: from the SaaS point of view, it will not be possible for the system provider to be accused of complicity since said party will not be aware of the documents that individual virtual machines are recommending; from the PaaS/IaaS point of view, we will protect individual virtual machines and their owners by means of hiding the identity of the node that outputs a recommendation, and providing mechanisms to calculate affinities without leaking all the information in the user profile.

In our security model, we will consider that any user of the recommender system or the virtual machines comprising the cloud may be attackers of the system. We are trying to protect intermediate nodes, indexers and normal users. To so, we will consider them as *attackers* of the system and check how much information they can learn. As such, we limit their power to the power needed for a fair use of the system. We call this power ''reasonable efforts'' and ''fair use'' of the system.

If an attacker using only reasonable efforts is able to learn something about the profiles of the documents or users, or the identity of customers or indexers, we consider that the legal system may assume that these attacks are possible and intermediate nodes and indexers can be accused of not performing them. At the same time, showing that under a fair use of the system it is not possible to attack the system, then nodes cannot be prosecuted for not trying to obtain information about

[1] In this citation and others that follow, emphasis is added.

documents or users and they are safe against legal attacks. Hence, attackers are not malicious nodes in the network but nodes that try to learn something about the system using only reasonable efforts.

The attackers may be: (i) the customers, (ii) any intermediate node in the path between the customer and the indexer, (iii) a malicious indexer or repository or (iv) the cloud provider. If the attackers are the customers, they are successful if they can identify the specific indexer that provided a recommendation. Second, if the attacker is an intermediate node, they are successful if they can identify the customer, the specific indexer that answered the query or obtain any information about their profiles. Third, an attacker acting as an indexer is successful if they are able to access the contents of a query or the exact profiles of the documents that it indexes. Finally, the attacker that acts as the cloud provider is successful if they are able to identify the identity of the indexer that answered a query, the source of the query or the contents of any messages exchanged by the different participants.

There are at least four ways for intermediate nodes to learn information about the network and documents: (i) inspection of the messages they route, (ii) analysis of their links to other nodes, (iii) collusion with other nodes to get information about other parts of the network, and (iv) effectively using the services that the network offers. Document indexers have an additional way to attack the system: (v) they can analyze the profiles they store.

We will provide these security services:

- *Indexer plausible deniability*. Indexers have the property of plausible deniability if it is not reasonable to force them to run the process to identify the documents they are recommending. However, indexers still should be able to provide correct recommendations. This way, indexers are not aware of the document profiles they are serving with some probability. Using plausible deniability, an indexer cannot be prosecuted for complicity in finding copyrighted documents.
- *Cloud provider plausible deniability*. Similar to the indexer deniability property, the cloud provider should not be aware of the recommendations that the system is providing.
- *Oblivious routing*. Intermediate nodes should not be aware of the content of the messages they are routing from the point of view of plausible deniability. This way, nodes that assist in locating documents by means of routing queries cannot be accused of abetting copyright infringement.
- *Indexer anonymity*. Customers do know the query that they send to the system and the results of this query. If prosecutors acting as customers are able to identify the indexer that answers a query about a sensitive document, they may accuse the indexer of abetting the download of the document.

Powerful attackers may abuse the system and perform extra actions to leak information about the intermediate nodes or users. In our security model, even if these power attackers learn something, the only action that they can take is informing the intermediate nodes about the fact that they are serving offensive documents. In this case, intermediate nodes should remove immediately these links to be safe. We will design our system to prevent power attackers from learning information about the users, but our efforts during this paper will be focused on the protection of the intermediate nodes, and the provision of the deniability property.

It is often argued that our system aims to support illegal distribution of copyrighted documents. This is far from our intentions. We believe in sharing information, photographs, videos, artwork and other documents of this kind. There are many creative communities in the internet to share personal documents that are organized as a social network, such as Flickr, deviantART or Youtube. In addition, cloud storage systems where users upload their personal documents are widely used. We believe that these communities would benefit of a recommendation system. Too often, copyright defense is extensively claimed to close down sites that opt for a new business model, or control the freedom of speech of individuals. Most of the complaints against the ACTA treaty are along these lines. The high cost and long wait time that an individual user must face in order to defend his or her case are too overwhelming even if they are ultimately not found guilty. In some cases, users cannot afford them and end up reaching an agreement with the prosecutor. In addition, unattended databases and intermediate nodes of a distributed network will assist in the location and access of documents that were published by a third party. These documents may have protected or prohibited content without the knowledge and consent of the intermediate nodes and indexers. We do not support neither piracy nor offensive content. We support databases that help a recommender system to remain operative for a majority of lawful customers, even if that implies giving service to a few undesirables.

This paper is structured as follows: In Section 2, we describe the state of the art for security in recommender systems specifically in terms of the plausible deniability of its indexers. Section 3 presents the architecture and building blocks of our proposal. Then, Section 4 details the sequence that a user follows to obtain a recommendation from the system. Section 5 analyzes the proposal according to the requirements described in this introduction. In this section, we propose a metric for indexer anonymity and analyze the system from this point of view (Section 5.1). The paper ends by presenting conclusions and future work.

2. State of the art

Many different recommender systems have been proposed in the literature, either P2P [2,34,29] or cloud-based [17]. Most of them do not try to provide security to their users in any way, since their creators focused on offering the best

possible recommendation to the end user. Other recommender systems try to protect the privacy of users that query the system for recommendations. As far as we know, the recommender systems currently described in the literature do not include plausible deniability for document indexers and intermediate nodes, since the legal attacks introduced in Section 1 are relatively new. In this section, we analyze some of these proposals.

Canny [6] proposed one of the first recommender systems that deal with the protection of the system's participants. Users are organized in communities with similar interests and recommendations are stored in centralized indexers. The evaluations given by a user about a document are sent to the community which automatically adds random noise and finally stores the evaluation into an indexer. This way, the indexer is neither aware of the evaluation that a specific user gave about a document, nor the exact evaluation of a document. This approach only distorts evaluations and does not hide the profile of the document. In privacy-aware recommender systems, indexers do not know the source of an evaluation. Unfortunately, in the proposed systems, indexers may be prosecuted for providing access to copyrighted documents since they actually know the identity of a document even if they are not sure about the accuracy of the evaluation. Furthermore, nodes inside the community that help add noise to a document evaluation do know the query that they are managing and the answer that the user gets, and they can be prosecuted for abetting copyright infringement. Hence, [6] is only able to protect the users' privacy, but they do not grant plausible deniability to indexers. From this early attempt, dozens of different system proposals appeared in the literature to protect the privacy of the evaluators and readers [3,45,46,19] but their privacy-based approach has the same problem as [6] and they are not able to protect indexers of documents from legal attacks.

Hermoni et al. [13] propose deniability as the security service that protects nodes in a distributed file system against censorship. The proposed system does not provide any search or recommendation mechanism to the users, and therefore can only download documents that users know about in advance. As such, the provision of database deniability prevents databases from participating in the search process, and then recommendations are not possible. Ref. [20] proves that there is a trade-off for utility and privacy in recommender systems, as is usual when dealing with privacy. In this paper, we will introduce this trade-off as the *plausible deniability* concept. Hu et al. [15] assert that plausible deniability is a necessary yet often underestimated service for most cloud service providers. For instance, participants of a cloud system, especially its manager, might prefer not to retain a copy of the key that customers use to protect their documents. In this case, managers can claim that they are not aware of the documents they are managing.

Mortier et al. [25] propose the creation of ''dust clouds'', highly dynamic virtual machines on a cloud system in which users run nodes to create an anonymous routing protocol whenever they need. As an optional step during the creation of this structure, Mortimer et al. propose the use of fake virtual machines to provide deniability to their routing protocol. They take advantage of the monitoring services of cloud systems to let the user prove that they are not the source of any dubious message. In our proposal of a recommender system, we will use the concept of ''dusts'' to create virtual machines that act as recommenders or indexers, and will not impose any monitoring service in the cloud system. Instead, the application model we use in this paper is similar to the privacy manager for cloud systems proposed in [28]. In their work, the authors propose obfuscation of private data using a module that is local to the customer, and the cloud server only processes obfuscated data.

Belle and Waldvogel [3] study deniability using ''lies'' within the user's profile. The mechanism to achieve deniability is by means of Bloom filters. A query is mapped into a Bloom filter, and some bits of the filter are switched. In this case, it is possible to tweak the length of the filter and the lying probability to provide deniability to the user, since they could have picked up any possible query that matches a specific Bloom filter. However, authors of [3] do not provide any search service in their network, and users must meet one by one to calculate their affinity. Furthermore, since there is no indexer in their proposal, if a user leaves the network their documents are lost by the community.

Although they are not targeted to recommender systems, plausible deniability for databases has been studied in the available literature [8,30]. However, the authors aimed to protect users and not indexers, and their system was not designed with distributed environments in mind.

3. DocCloud system architecture

We call our recommender system DocCloud. DocCloud involves a social network of similar users, a cloud system of recommenders and a distributed secure file system.

We work on a social cloud [7] that represents a social network. Inside, there are N clusters of nodes that gather users who share similar interests and hence are *affine*. When users join the network, they identify their most suitable cluster according to their interests. Then, users inside the cluster link each other. There are several proposals in the literature to create this kind of social cloud according to the interests of the users in a decentralized fashion. The interested reader is referred to [29,2,43].

In order to simplify the system description, we assume that nodes in the network play a single role at any moment. In a real system, a node will play several of these roles simultaneously and readers should be aware of this simplification. We will use the term *users* as a placeholder for merchants and customers when they access indexers and their specific role is not important. To simplify the description of DocCloud, from here onward we will assume that a user only presents a single profile and then joins only one cluster of users. Actually, users may have several profiles and join different clusters, according to their own needs.

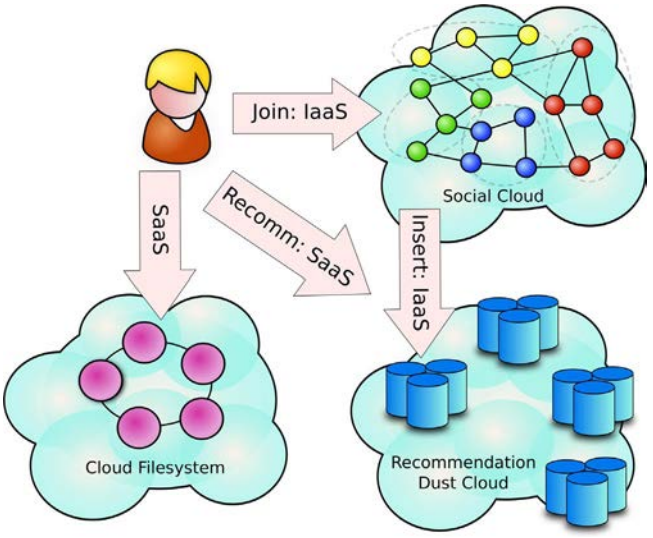

Fig. 1. DocCloud: system architecture.

We will also define a cloud system that includes all document indexers. This cloud is organized in a *dust cloud* in the sense of [25]. That is, each one of the virtual machines is controlled by a user that instantiates it as needed, and dismisses the machine when the job is done. The cloud provider maintains a list of machines included in each of the N clusters of the system. The users instantiate virtual machines that connect to one random node in the desired cluster and to other virtual machines whose users have similar profiles. Using this process, there are N different clusters of dust clouds inside the cloud. We will use these virtual machines as indexers of documents in the recommender system.

Finally, our recommender system will use another cloud system to store and distribute the real documents. We will assume that, given a URL, users are able to download a file from this file system in a private way. The details about the management of this file system fall outside the scope of this paper. The interested reader is referred to secure distributed file systems such as [42,5,35].

An overview of DocCloud is shown in Fig. 1.

3.1. Building blocks

In this section we explore the general structure of the system and its building blocks. In Section 4, we combine these blocks to build the system.

3.1.1. Key management

Customers will identify and join a cluster of customers with similar profiles. When a customer joins a cluster, a new key must be created and distributed among the group's members. We propose the key management algorithm of Hernández-Serrano et al. [14]. This is a Group Key Management (GKM) scheme, which manages the changes of the shared key during the life of a group. The main challenge of a GKM scheme is the secure update and distribution of the shared key among the group's members.

The output of the key management scheme produces a shared key that all members of the group know. This key is updated when a new member joins the group, or an existing member leaves. We assume that entities that are not group members will not actively seek the group key. That is to say, we assume attackers may have access to the group key, but legitimate nodes that are not group members will not be interested in owning this key.

3.1.2. User and document profiles

One of the most successful representations of a document in the field of information retrieval is the use of vectors [21]. In many cases, the components of these vectors are the frequency of appearance of certain common terms in the document under analysis. This same idea can be generalized to include ontologies or categories of terms. Indeed, it is possible to convert a vector of terms (*bag-of-words*) into a vector of ontologies or categories (*bag-of-concepts*). The definition of the ontology that classifies documents falls outside the scope of this paper. The interested reader is referred to recent works in the fields of information retrieval and artificial intelligence [21,37,22,39].

We work on a social network where users share a set of documents $D = \{d_1, d_2, \ldots, d_d\}$. According to [39], it is possible to define n different categories and assign each document d_i a vector $p\eth d_i\Þ \frac{1}{4} \eth c_1; c_2; \ldots; c_n\Þ$. We use the term *document profile* to

refer to this vector. In this paper, we establish that each component c_i of $p$ðdP is an integer from 0 (cannot be classified in this category) to M (completely classified in the n category).

We use the term *social space* P ¼ ½0; M]^n to refer to the set of different profiles in the network. We assign each user u a *user's profile* in the social space P based on their interests. We will not make any assumptions about how these users' profiles are assigned. In any case, we strongly believe that the algorithm that assigns profiles to users must (i) take as an input the user's past behavior and the documents that they shares, and (ii) create equally likely profiles.

Searches of documents in the network are performed through queries q. Since these queries will be compared with document and user profiles, they must be in the social space q 2 P. Hence, queries are assigned a *query profile*. We establish that customers do not issue random queries and their queries are always related to the user's profiles.

In the social space, a metric for profiles d : P × P ! R can be defined. This paper uses the cosine distance [33], which was successfully used in other proposals to compare document profiles [33,21,17]. The original proposal of the cosine distance uses vectors of real numbers, but the reader will notice that it can be safely used with vectors of integers as well. Given two profiles a and b in P, the similarity between them is calculated according to Eq. (1).

$$sim(a, b) = \frac{a \cdot b}{\|a\| \|b\|} = \frac{\sum a_i b_i}{\sqrt{\sum a_i^2} \sqrt{\sum b_i^2}} \qquad (1)$$

Using this definition for similarity, $sim(a, b) = 0$ shows that profiles a and b are completely dissimilar, and $sim(a, b) = 1$ shows that a and b are completely similar. We define that two profiles a and b are *affine* under a real number k if their profiles show a similarity $sim(a, b) > k$. The objective of a recommender system is to find those documents in the social network with an profile p affine to a query q.

The cosine distance, as defined in Eq. (1), is not the most optimal way of computing similarity between documents [23]. However, the cosine distance is simple enough to be computed using only additions and multiplications; it will then be possible to use special cryptosystems to protect profiles. We establish that a protected profile ½a] = {½e_1]; ½e_2]; . . . ; ½e_n]} is a profile where each component was encrypted using an additive-homomorphic cryptosystem such as Paillier's [27]. Only the user that knows the private key can decrypt protected profiles. We will represent the decryption of this profile as ½a]$^{-1}$ = a. Given a profile in clear a with a known norm kak and a protected profile ½a], it is possible to make the calculation of a new parameter E(a; ½e]) according to Eq. (2).

$$E(a, [e]) = \sum_{i=0}^{n} a_i [e_i] \qquad (2)$$

The reader should be aware that operators of Eq. (2) are on the space of the encrypted text, not on the set of integers, and these operations will be calculated by an entity that does not know the private key of the encryption of ½a]. Under these circumstances, given E(a; ½e]), only the entity that owns the private key of ½a] is able to calculate the real similarity as in Eq. (3):

$$sim(a, e) = \frac{[E(a, [e])]^{-1}}{\|a\| \|e\|} \qquad (3)$$

We use this block as follows. Databases index profiles in the form p = {p_1; . . . ; p_m}. A customer creates a secret key k. Then, the customer sends to the database queries ½q] that are encrypted under k. Using an homomorphic crypto-system as the one described before, the database can calculate the value of the parameter E(p; ½q]) for each document profile that it indexes, even if the database does not know k and q. Finally, the E(p; ½q]) of each document profile is sent back to the user, which calculates the similarity using Eq. (3).

3.1.3. Profile projections

As we will see, merchants in a cluster A insert the profiles of the documents they share into the indexers of a different cluster B. Since either these profiles include private information or private data may be inferred from them, profiles cannot be inserted directly into the system. These profiles should be modified in such a way that the personal information is protected while the calculation of similarities between profiles is still possible.

Our proposal is projecting profiles from P onto a new social space with fewer dimensions. Original profiles p_n are in a social space of n categories and are projected into a social space of m categories using a projection matrix $M_{m \times n}$. Thus, given $y_m = M_{m \times n} p_n$, an attacker is only able to calculate a class of original profiles $\hat{p}_n$ that solves the undetermined system:

$$\hat{p}_n = p_p + c_1 u_1 + \cdots + c_{n-m} u_{n-m}; \qquad (4)$$

where p_p is any profile that verifies $y_m = M_{m \times n} \cdot p_p$; c_1; . . . ; c_{n-m} are arbitrary real numbers and u_1; . . . ; u_{n-m} are a basis of the kernel of $M_{m \times n} \cdot p = 0$. As a consequence, the set of profiles that could be projected onto y_m is a subspace of dimension $n - m$, and attackers learn that $p_n \in \hat{p}$. If all profiles in the social space are equally likely, as the social model must enforce, then all profiles in $\hat{p}$ are equally likely and the attacker cannot learn any additional information from the projected profiles.

Two different problems arise in this scenario: (i) whether comparison of profiles makes sense in the projected space and (ii) the amount of information of the original profile that is preserved after the projection.

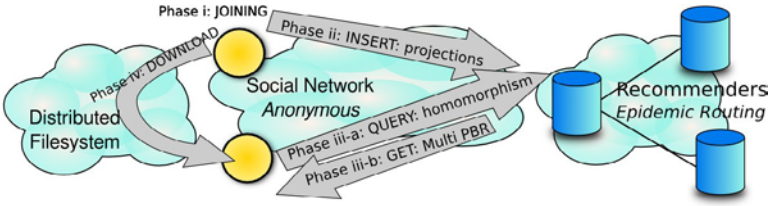

Fig. 2. An overview of the mechanisms.

We will draw on two lemmas. First, the Johnson and Lindestrauss' [1]. According to this lemma, given a set of vectors of dimension n, it is possible to calculate a projection onto a metric space of dimension $m < n$ while limiting the error of the distances between projected vectors. Hence, after projections, two users that are affine in the original social space are also affine in the projected space with high probability. The second lemma we use for building our system is the undecomposability of random matrices [18]. According to this lemma, not only is the calculation of the exact original description not possible after projections, but also a malicious user will not be able to calculate a single component of the original profile if $n \geq 2m - 1$.

By means of these two lemmas, if we use a specially crafted matrix M to project a profile $p \in \mathbb{P}^n$ onto a profile $y = Mp^t \in \mathbb{P}^m$ where $n > 2m - 1$, then given y with high probability it is not possible to recover any component of p and the distances in the projected space are still related to the distances in the original space.

The creation of the projection matrices is an extremely interesting issue that, due to its complexity, should be addressed in a future work. From this point forward, we will assume that a projection matrix with the necessary properties is already defined. By means of the referred lemmas, an attacker with access to the projected profiles cannot make any good guess about the original profiles or extract any relevant information from the projections. At the same time, two profiles that are affine in the original space will be affine in the projected space with high probability.

The reader should be aware that a projected profile can be encrypted using the mechanisms presented in Section 3.1.2.

4. Recommender system operation

In DocCloud, we identify four different phases to get a document recommendation: (i) the creation of a social cloud, (ii) the insertion of document profiles into the indexers, (iii) the search of recommendations by the customers and (iv) downloading of the recommended document. Fig. 2 shows these steps and the mechanisms used in them.

The network at the center of Fig. 2 represents the social network that users join according to their preferences. This network is used to link users with similar interests and improve the recommendation process. However, we will not use this network to provide recommendations. The document profiles are indexed in a different network, which is the recommenders' network of Fig. 2. This network is the main object of analysis in this paper. Finally, documents are distributed from a distributed file system that is depicted on the left side of Fig. 2.

4.1. JOINING: Creation of the social cloud

Users of DocCloud are arranged in clusters according to their interests. How new users discover the most suitable cluster of interest and the management of these clusters fall outside the scope of this document. We refer the interested reader to [29,2,43] for the creation of a social network of similar users. Epidemic routing is a convenient method to distribute messages when data must arrive to as many nodes with shared features as possible. From here onward, we assume there is an epidemic routing algorithm inside the different clusters of networks to distribute messages and support the discovering of other affine users. Other proposals that use epidemics routing in a way similar to ours are [10,34,44].

After joining the most suitable cluster, nodes in a cluster A will agree on a secret key that includes two items, $K_A = (B, M_A)$, using the block introduced in Section 3.1.1. The first part of the key is a reference to another cluster of nodes B that will be used to store the profiles of the documents shared by nodes in A. The reader should note that nodes in B will not need to be aware of the identity of the cluster A. The second part of K_A is a random matrix M_A that we will use to protect profiles in A, as introduced in Section 3.1.3.

An important point to consider is that since users are not authenticated when they join group A, we cannot assume that attackers are unable to get K_A. We do assume that lawful indexers in B are not going to actively look for K_A [15], and we will soon see that even if they are handed over K_A, they will not be able to calculate the original document profiles.

Nodes in B are not free contributors. They accept index documents for nodes in A because they insert the profiles of their own documents in a different cluster $C \in N$. That is to say, in a real network nodes are customers, merchants and indexers. For the sake of clarity, during the rest of this article we will consider only the merchant/customer roles of nodes in A and the indexer role of nodes in B.

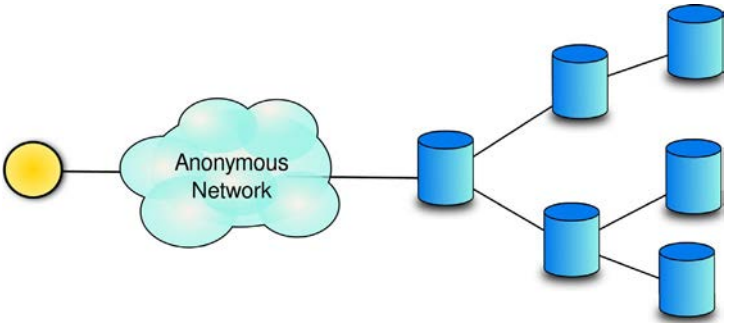

Fig. 3. A tree of indexers answers a query.

4.2. INSERT: Inserting document profiles into indexers

After users join a specific cluster of the social cloud, they will share some documents with the community. During this phase, user $a \in A$ plays the role of *merchant*. For a document d_i, a assigns a profile $p(d)$ using the mechanisms discussed in Section 3.1.2 and publishes the document d under $URL(d)$ in the file system in the cloud, as described in Section 3. Finally, a inserts the pair $\langle p(d); URL(d)\rangle$ into a random indexer $b \in B$.

The reader will notice that if indexers and intermediate nodes are able to access these profiles in clear, they will know exactly what kind of documents they are providing access to. In addition, they can even estimate the user profile by simply collecting enough document profiles from the same source. Indexers and intermediate nodes need to use these profiles to route and answer queries according to the affinity of the user to the document descriptions that they index.

In order to provide deniability for DocCloud, we need to devise a mechanism that hides some of the information in the descriptions but is still useful to calculate affinities between elements in P. We explored this mechanism in Section 3.1.3. When a user $a \in A$ inserts the description of a document d into an indexer $b \in B$, they send the pair $\left(M_A p_n(d)^t; URL(d)\right)$, where the first component is the projection of the document profile and the second component is the URL of the document.

The identity of the merchant reveals information about the contents of a document, since we assumed that users' profiles are related to the documents they own, as discussed in Section 3.1.2. In this regard, nodes in A should not publish the pair on their own in node b, but they must hide inside an anonymous cloud. We do not enforce any particular anonymous network, but for the rest of this paper we assume the existence of a distributed anonymous network such as crowds [32] or dust cloud [25].

Next, an epidemic routing protocol occurs to distribute information inside the set of indexers from B. The objective of this epidemic protocol is to randomly spread the information in the documents through many different indexers. This way, (i) the availability of the document profiles increases, (ii) nodes in A may contact a random node $b \in B$ to perform queries without compromising the efficiency of the results; and (iii) the possible liability of providing access to a document is shared among different nodes. There are many existing proposals of an epidemic protocol for locating documents in distributed systems [10,29,2,43]. Indeed, nodes in B save the document description of nodes in A, but since they have a user profile as well, it is possible to use this profile to organize nodes in B according to their interest, as in [43]. Thus, nodes in B can take advantage of the epidemic algorithms proposed in the literature, but they store and replicate the description of documents owned by nodes in A instead of the profiles of their own documents. These same epidemic routing mechanisms will be used to distribute query messages inside B.

4.3. QUERY and GET: Recommending documents

Our security goal during this phase of the recommender system is to provide indexer anonymity, i.e., to make it impossible for an attacker to distinguish which indexer stored a particular answer to a query.

During this phase, the participants view the system as Fig. 3 shows. Although it is not stated in this section, the reader must take into account that all communications between customers and indexers use the anonymous routing introduced in the previous phase. Hence, indexers cannot identify the customer that issued the query they are currently managing.

We describe the process in several steps. First, the customer issues a query to the indexers using an anonymous channel. Then, the indexers are automatically organized as a tree structure and calculate the parameter $E(\vec{d}; \frac{1}{2}q])$ between the documents d they index and the query q. As a result, the customer gets a vector of encrypted distances, decrypts these distances using Eq. (2) and chooses the indexes of documents that are more similar to the query. Next, the customer downloads the URLs of the selected documents using a PBR scheme. Finally, the customer uses these URLs to download the recommended document from an external, distributed file system. In this section, we look at these steps in detail.

4.3.1. QUERY document profiles

A user $a \in A$ that searches for a document in an indexer $b \in B$ builds a query q_n and chooses a private key K_a. This key is only known by a, which projects and encrypts the query as stated in Section 3.1.3 to create $\frac{1}{2}q]$. Next, a anonymously sends $\frac{1}{2}q]$

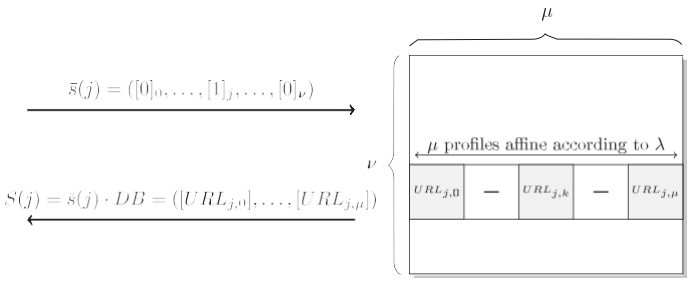

Fig. 4. A PBR scheme in the multi-indexer scenario.

to a random indexer $b \in B$, which calculates the parameter $e_j(p_j, r_q)$ of every document profile that it indexes using Eq. (2), and then creates a vector E_b that contains these parameters e_j. At the same time, the query is distributed to other indexers of the cluster B using an epidemic algorithm that creates a tree-shaped organization of random indexers. All the indexers in the tree answer the query, and all answers are joined together. Finally, a will receive an ordered set $E = [E_b$ where each component is the parameter $E(p, r_q)$ of the documents that the nodes in B indexed. Next, a decrypts and calculates the distances to the documents, and selects those that are affine according to the threshold k.

An epidemic algorithm without loops inside the cluster B, as we discussed earlier, can create the tree of random indexers as shown in Fig. 3. For the purposes of this document, the tree structure has a disadvantage: answers provided by nodes in the inner branches of the tree will be located in the last positions of the joint vector E. Since we want to provide indexer anonymity, it is necessary for each node to locally permute the vector E. This way, the customer cannot identify the position of an indexer in the tree according to the position of its answer within the vector E. In addition, during the GET sub-phase, indexers must locally undo these permutations. The permutation that each indexer apply must be a secret that should not be made public. This mechanism can be refined to avoid duplicate items by using Bloom filters.

4.3.2. GET URLs: Private Block Retrieval Protocol

Finally, a private block retrieval (PBR) scheme takes place in order to download the $URL(r)$ associated with the document of their interest without leaking which specific $URL(r)$ the customer is obtaining.

The PBR hides the index of the item that the customer a is retrieving, and ensures that no one in the path $a \rightsquigarrow b$ knows which item a is interested in, not even the indexer b. One of the PBR schemes suitable for our work is presented in [11]. The complexity of this algorithm is $O(k + j)$, being $k > \log(n)$ a security parameter and n the size of the database in bits. The authors of this scheme state that it has the lowest asymptotic communication complexity of the current proposals, and their statement seems not to be challenged by recent literature. For the sake of completeness, we now describe a simple yet useful PBR scheme to get a URL from the indexer without leaking which URL the user is requesting. This PBR system uses the Paillier cryptosystem [27].

To enhance the efficiency of the communication in terms of the number of exchanged bits, we will use a two-dimensional view of the database. Hence, the customer will not receive a single *URL*, but instead a set of l related, affine *URLs*. First, the database needs to be organized as follows: An indexer creates m sets that contain at most l URLs each, such that the URLs inside a set are affine according to a pre-shared k, being k the affinity threshold. These sets are then organized as a matrix of m rows and l columns. We assume that the database used an optimization mechanism to calculate a representative of each set [31], and the distance to this representative profile was sent to the user during the QUERY sub-phase.

Then, the user requests for the jth row of the matrix, where j was the output of the previous sub-phase. To do this, the customer constructs the selection vector $s(j)$ of Eqs. (5) and (6), where all components are the Paillier encryption of 0 except the jth component that stores the encryption of 1. The reader should note that the Paillier encryption is probabilistic, and an observer cannot identify the components of this vector.

$$s(j) = (s_0, s_1, \ldots, s_k) \tag{5}$$

$$\text{where } s_i = \begin{cases} [1] & \text{if } i = j \\ [0] & \text{if } i \neq j \end{cases} \tag{6}$$

Then, the indexer multiplies each component of the selection vector for each row of the URL matrix, and adds the results. These operations are shown in Eq. (10), which takes advantage of the homomorphic properties of the Paillier cryptosystem. As Fig. 4 shows, this process results in a vector with the URLs in the ith row.

$$S(j) = \sum s_i \cdot row_i \tag{7}$$

$$= [0] \cdot row_1(p); \cdots; (p[1] \cdot row_j(p); \cdots; [0] \cdot row_m \tag{8}$$

$$= [row_j] \tag{9}$$

$$= ([URL_{j1}]; \ldots; [URL_{jl}]) \tag{10}$$

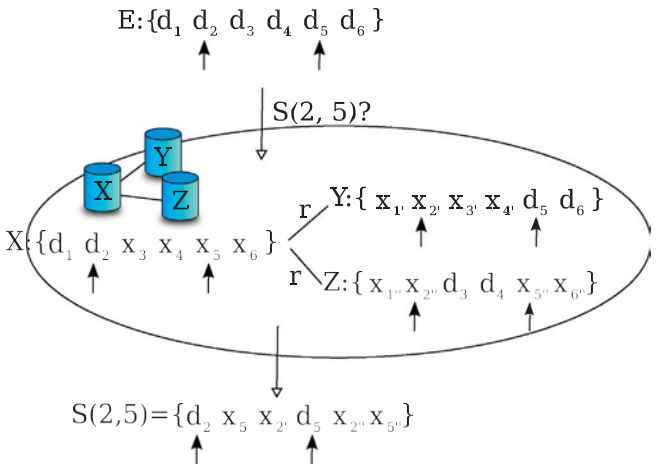

Fig. 5. The two dimensional PBR system.

Further analysis of possible PBR systems can be found in [26]. Even if there are other, more efficient proposals than this one, the reader should note that the PBR system described is very suitable for our proposal, since a user gets a complete row of affine profiles from the indexer with a single query and most of them would be of interest to them.

We will clarify the complete recommendation process, including the indexer tree, using an example. Fig. 5 shows a tree of three indexers that returned an array E with 6 distances. Indexer x contributed with $\{d_1, d_2\}$ to the array, indexer y contributed with $\{d_5, d_6\}$ and indexer z with $\{d_3, d_4\}$. When the user receives the vector of encrypted distances, they choose the elements 2 and 5, and start a PBR process. Indexers do not know the position of the elements that a user wants to download, so each x,y and z builds a virtual database containing the documents they store in the same position they answered in E, using empty elements for the other positions. When x receives a request for d_2 and d_5, it is able to answer d_2 and the 5th position is empty, x_5. z cannot answer with any valid document and only returns empty positions, while y answers an empty item and a valid *URL* d_5. The reader will notice that indexers are unaware of whether they are returning valid *URLs* or just empty items. Then, the user receives the answer array, undoes the PBR scheme and finally discards the empty items.

Using this scheme, it is clear that neither indexers nor an observer in the middle of the path can learn whether or not a particular document comes from a specific indexer. Indexers are unaware even if they gave a valid answer to the user. With the additional permutation introduced in Section 4.3.1 (which was not shown in this example in sake of simplicity), the customer cannot reconstruct the original positions of the indexers within the tree.

4.4. DOWNLOAD the document

Finally, the customer will download the desired document from the distributed file system. A secure distributed file system must take these issues into account. We assume the existence of a secure distributed file system that (i) makes not possible to learn any information about þðdÞÞ from *URL*(d) of a document, and (ii) it is not possible to access the *URL*(d) from any node not in A without the group key K_A.

The interested reader is referred to [42,5,35] for secure distributed file systems that meet these requirements.

5. Analysis of DocCloud

This section revisits the system requirements introduced in Section 1.2. These requirements are analyzed according to the security mechanisms and procedures described in the last section. In addition, Section 5.1 explores the indexer anonymity requirement in depth.

Indexer plausible deniability. Indexers store document profiles "projected but in clear". This is not a security risk, and indexers cannot be legally prosecuted for failure to calculate the original document profile. First, indexers in cluster B will not search actively for the shared key of an external cluster A because they cannot identify it. Even if an attacker informs the indexer about the identity of the cluster A and distributes the shared projection matrix M_A, the projection matrices still hold the undecomposability threshold referred to in [18]. Hence, there are not enough components to learn even a single category of the original profile. Second, merchants are hidden in a social network that uses anonymous routing. Thus, indexers are not able to identify the source of a document description and cannot use any source-analysis mechanism to learn information about the document profiles. Third, since indexers do not distribute documents but only index URLs that are protected with the shared key K_A, they cannot access the actual document. We conclude that indexers: (i) can deny that they are able to access the document they index, (ii) cannot undo the projection and calculate the original profile with enough

precision and (iii) cannot identify the merchant that inserted the profile. This way, the system provides the indexer plausible deniability.

Since a malicious user in cluster A is able to make a good guess about the kind of profiles of the other users in the cluster and the document profiles they share, the attacker is able to inform indexers of the document about the profiles that they store. For this attack to be successful, the malicious user must be able to identify the indexer of a specific document profile. We will examine this attack later.

Cloud provider plausible deniability. The cloud provider only observes communication from customers to document indexers. Messages during queries use homomorphic encryption and thus cannot be accessed. The profiles that insertion messages include are projected, and therefore enjoy the same protection as individual indexers. As observers of the communications and thanks to the PBR system, they cannot identify the specific indexer that provided an answer. Finally, they could access the virtual machines in the cloud, including memory and data. Even if instances save their databases in clear, they only manage projected profiles. This way, the cloud provider enjoys the same protection as individual databases.

Oblivious routing. Nodes in the middle of the path between a user a 2 A and indexers in B route messages between them. They are inside the anonymous cloud, so they are able to learn as much information about the source of a message as the anonymous cloud allows. We propose a system where document profiles are published in a projected social space but in clear. As a consequence, the intermediate nodes are similar to indexers in terms of security analysis, and therefore the same considerations can be applied to them. However, since profiles will be projected to a dimension $m < \frac{n-1}{2}$, according to [18], the intermediate nodes cannot calculate any specific component of the profile. During the recommendation phase, messages exchanged by indexers and users are encrypted using a key only known by the customer a, and thus intermediate nodes cannot gain any knowledge about messages. We can conclude that intermediate nodes are as protected as indexers during the publication of the profiles, and that they cannot learn anything about queries during the recommendation phase.

Indexer anonymity. The multi PBR scheme introduced in Section 4.3.2 prevents users from learning the identity of the specific indexer that answered a query. In fact, not even the indexers taking part in the recommendation tree know whether or not they answered the query. Only the user issuing the query is able to distinguish empty items from real information, and in any case the local permutation of the vector of answers in every hop of the tree prevents gaining any knowledge about the indexer's identity. As explored in the next section, by means of limiting the size of the answer that an indexer returns, the system provides indexer anonymity.

Some participants in the indexers' network are able to identify other indexers contributing to a reply vector. Actually, the reader would notice that the very indexer that contributes to the reply vector is not aware whether it is contributing with useless data, and other indexers are not aware whether or not a specific indexer is contributing with useless data.

In the event that a specific indexer is absolutely isolated from the network and forced to link only to attackers, then the indexer could be identified as the only contributor to some offense that he is not aware. Even in this rare case, the reader will notice that (i) the indexer is not aware of what offense he is exactly collaborating to; (ii) the indexer will only provide a URL, and not the real document; (iii) this kind of attacker is very close to a powerful, global attacker, and we do not considered these kind of attackers in our paper; and (iv) the attackers will have to prepare so much their forged attack that the event of being accused back on ''forcing the commission of an offense'' seems really possible.

The first three requirements are met using the building blocks described in Section 3. Indexer anonymity is the only requirement that does not depend on external mechanisms but depends directly on the structure of the system that we are proposing. Specifically, the property depends on the number of contributions of each indexer participating in the tree depicted in Fig. 5, as discussed in Section 4.3.

In the next section, we analyze in depth the indexer anonymity property and the management of the indexer tree to maximize this property.

5.1. A metric for indexer anonymity

In this section, we analyze the indexer anonymity property that the system shows. We also provide a way to calculate the maximum number of items each indexer must return to provide plausible deniability to the indexers.

Indexers are arranged in a tree as Fig. 5 shows. In Section 4.3.1, we saw that items inside the answer vector must be randomly shuffled to prevent the attackers from being able to identify indexers by inspecting the positions of the results. Indeed, indexers that are deeper in the tree structure send their profiles to their root, which performs a Bloom filter to avoid repetition of profiles in the answer. With this filter, it is more likely that the information of a profile in the answer array comes from the leaves than from the root. In an extreme case, the root of the indexer's tree does not contribute at all to the answer. The attacker cannot identify the indexer that holds a profile, but can assign a different probability to each indexer in the tree. In this sense, the anonymity set is biased towards the inner leaves of the tree and is smaller than expected.

The trivial solution to avoid different contributions is to simply not implement Bloom filters and let every node contribute to the answer with its entire database. However, we describe an epidemic algorithm to spread document profiles among the indexers of B. Hence, the same document profile is stored by many indexers of B. Not removing duplicates in the answers implies that indexers must recalculate the PBR functions of large vectors, which is a slow process, and the number of exchanged bits grows unnecessarily. We will show that we can still get balanced contributions without having to send the whole database during each query.

In this section, we address the scenario in which an attacker learns the list of document profiles returned by a given query. This may be the case of the sender of a query. We establish that the attacker wants to identify the indexer storing a specific document profile. To do this, they issue a query that points exactly to the targeted document profile. We assume the attacker knows the identity of the indexers that participated in the indexer tree. This is the case, for example, of an attacker acting as a cloud provider, or a malicious indexer inside cluster B.

The main idea of this section involves calculating the average size of the vector $a = |E|$ that the tree of recommenders outputs. Hence, we can force the scenario where if the indexer tree has k nodes, then each indexer contributes to E with a/k URLs. In this sense, from the customer's perspective, any document profile could uniformly come from any document indexer in the tree.

5.1.1. Uniform assumption

Indexers of a cluster B store a set $D = \{d_1, d_2, \ldots, d_N\}$ of different document profiles. Each indexer stores $n < N$ of them. When a query arrives, the recommender system will randomly pick a subset of indexers $S \in B$ with cardinality k that help create the answer to the query, as explained in Section 4.3.2.

As an initial approach, we assume that document profiles are uniformly spread in B. That is to say, given a document d_i, whether or not an indexer stores d_i is independent of the indexer. For any indexer $b_u \in B$, we define an event $\overline{X}_{i,u}$ as "the document d_i is not in b_u". As we assume uniform distribution of documents, the probability distribution function (pdf) of X can be modeled as a hypergeometric distribution: given a set of N different documents, $x = 1$ are of interest to us. That is, d_i. Then, we pick without replacement n documents and calculate the chances that $l = 0$ of these are d_i.

$$pdf(\overline{X}_{i,u}) = hypergeom(x = 1; n = n; N = N; l = 0) \quad (11)$$

$$= \frac{\binom{1}{0}\binom{N-1}{n}}{\binom{N}{n}} = \frac{N-n}{N} \quad (12)$$

Given a set S of k indexers, each one storing n different document profiles, we define the event $\overline{Y}_i$ as "the document d_i is not in any of the indexers in S". The complement of this event, Y_i, means that the document d_i is in at least one indexer in S. Since we assume a uniform distribution of documents, the pdf of Y_i is constant for any document and indexer, and from here onward we will drop the subscript. Hence, the pdf of $\overline{Y}$ is:

$$pdf(\overline{Y}) = pdf(\overline{X})^k \quad (13)$$

$$pdf(Y) = 1 - pdf(\overline{Y}) = 1 - pdf(\overline{X})^k \quad (14)$$

Finally, we define an event Z_j as "the subset S has j different documents". Since each indexer stores n different documents, the minimum value of Z_j is n, that is to say, the k indexers are the same. Meanwhile, the maximum value of Z_j is nk, and this is the case where the k indexers store completely different documents. Hence, the universe of Z_j is $[n, kn]$. This event is equivalent to "the subset S contains at least one instance of j documents and no instance of $N - j$".

Now, we can calculate the pdf of Z_j as follows.

$$pdf(Z_j) = \begin{cases} \binom{N}{j} pdf(Y)^j [1 - pdf(\overline{Y})]^{N-j} & \text{if } n \le j \le nk \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

The pdf of Eq. (15) is a binomial distribution that has been shifted by kn, and therefore its average is:

$$E[Z_j] = (k-1)n + pdf(Y)N = (k-1)n + \frac{(k-1)n(N-n)^k}{N^k} \quad (16)$$

Eq. (16) captures the expected number of different items in the answered vector. The results of Eq. (16) can be used to improve the anonymity set of the indexers. Indeed, if each of the k indexers contributes with $n = E[Z_j]/k$ items, then the contribution of each indexer to the answer array is likely equal. In order to achieve this, we must encourage the scenario where $E[Z_j] = nk$. We call this n the optimal contribution coefficient for the indexers, n_{opt}, since it allows uniform contributions for the indexers and maximizes the anonymity of the set. We can calculate the optimal contribution of each indexer n_{opt} as the n that matches the following condition.

$$E[Z_j] = n_{opt}k = \frac{(k-1)n_{opt}(N-n_{opt})^k}{N^k} + n_{opt} \quad (17)$$

$$k = \frac{(k-1)(N-n_{opt})^k}{N^k} + 1 \quad (18)$$

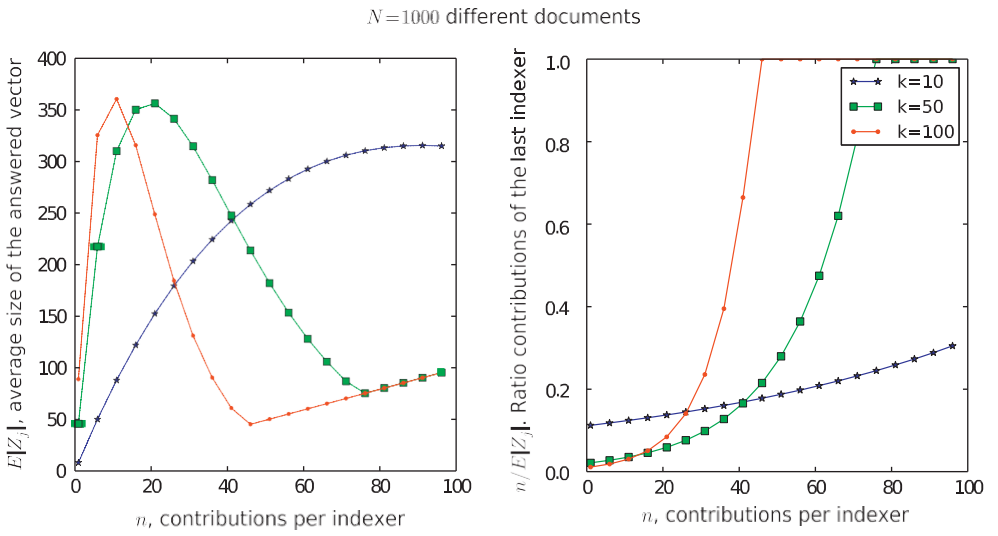

Fig. 6. Analysis of contributions for $N = 1000$ documents.

$$n_{opt} \frac{1}{4} N \quad 1 - \sqrt[k]{\frac{i}{k-1}} \qquad \text{ð19Þ}$$

Eq. (19) shows the optimal contribution of each indexer to achieve maximum anonymity. Alternatively, Eq. (17) shows the optimum number of indexers that must be contacted, for a fixed number of contributions from each indexer.

In the extreme case of n, $k \ll N$, $E[Z_j] \approx kn$, in order to achieve uniform contributions, each indexer should contribute $k \approx n$ items, nearly every item in the database. Since there are many more documents in the system than the capacity of an indexer, if the subset of indexers is small (k small), the chances of collision are small and indexers can collaborate with every item.

Even if this could simplify the system design, it is not desirable in terms of efficiency. Users of the system will want to calculate the affinity to as many profiles as possible, to be able to locate the more interesting documents. In this sense, the system will be designed for $kn \approx N$.

There is an additional consideration for this analysis. We started this analysis by selecting a random subset of indexers $S \in B$. Actually, the selection of a subset of indexers may affect negatively the anonymity of the indexers in set S. By means of the intersection attacks described in [4], a long term analysis of the answers may identify that some documents are only included when an indexer is inside S, and then the indexer is identified. In any case, this effect can only decrease the upper limit of the optimal number of contributions from each indexer and hence the results of Eq. (17) are still valid as an upper limit. Readers should take into account that the real limit is below this number. In any case, we do not consider long term attacks as a normal behavior and then, they are not the kind of attackers analyzed in this work.

5.1.2. Epidemics assumption

In Section 4.1, we suppose that there is an epidemic routing algorithm in the indexer set. The effect of this algorithm on document profiles is that it is much more likely for neighbor indexers to share similar document profiles, and the likelihood of replicated data is higher if indexers are adjacent. Hence, in a real system the distribution of profiles is not uniform as we assumed in the last section, and the pdf shown in Eq. (12) will depend on the position of the indexers in the tree. Hence, $pdf(X_j)$ is not a simple hypergeometric distribution as calculated in the simplified scenario. On the contrary, $pdf(X_j)$ must be weighted with the position of the indexer.

As an initial approach to analyzing this problem, we are going to assume that indexers are ordered in a line. This is a simplified tree with no branches. As in the last section, the event $\overline{X}_{j;u}$ represents "the document r_j is not in d_u", but this time we define u as the position in line, from the root $u = 0$ to the branch $u = k$. Then, we describe a routing epidemic protocol in such a way that there are two real numbers ε and d such as:

$$P\delta\overline{X}_{j;u}\mid\overline{X}_{j;u-1}\flat \frac{1}{4} p \flat \varepsilon \qquad \text{ð20Þ}$$

$$P\delta\overline{X}_{j;u}\mid\overline{X}_{j;u-1}\flat \frac{1}{4} p - d \qquad \text{ð21Þ}$$

$$0 \leq \varepsilon \flat d \leq 1 \qquad \text{ð22Þ}$$

with p being the probability of the uniform assumption in Eq. (12). These equations may be interpreted as follows: the probability that a document is (is not) in an indexer is higher if it is (is not) in the preceding indexer. Furthermore, we analyze a routing protocol that makes the variation of likelihood of $\overline{X}_{j;u}$ negligible given $\overline{X}_{j;v}$.

Eq. (22) represents a Markov chain of probabilities. This was analyzed in [16]. To aid the reader, we will now present the solution for $P(X_{j,d})$ using our notation.

$$P(\overline{X_{j;u}}) = \frac{(p-d) - (E)(d)^u(E)^d - (1-p)^d}{1-E} \quad (23)$$

The last term of this equation attenuates with k, and therefore is a monodic decreasing function with a maximum of p for $k = 1$. In this new scenario, $P(\overline{X_{i;u}}) \leq p$. The values for E and d cannot be easily computed since they depend on the actual epidemics algorithm in use, but we can conclude that any epidemics algorithm we choose should use Eq. (19) as an upper limit for the contribution of each indexer.

A real scenario with several branches in the indexer's tree is even more complex. The probability $P(\overline{X_{j;u}})$ follows a Fisher's non-central hypergeometric distribution. In order to calculate the new pdfs or achieve conclusions similar to those of last section, we need to model the epidemics algorithm used by the indexer set. This paper does not assume a particular epidemic algorithm for the set of indexers, and the exact calculation of the k_{opt} that maximizes anonymity falls outside the scope of this paper.

5.1.3. Discussion

Even if we do not achieve a final result for the probability, we can extract some conclusions from the last scenario. Fisher's non-central hypergeometric distribution is always shifted toward the left and its average is less than that of the central hypergeometric distribution. Besides, the analysis of this section for a simplified tree showed that the Markov chain that epidemic algorithm creates always decreases $P(X_{j,u})$. In practice, this means we can use Eq. (19) as an upper limit for the amount of collaboration of the indexers of the system.

Fig. 6 clarifies the analysis of this section. We used a recommender network that indexes $N = 1000$ documents under the uniform assumption. The figure on the left represents the expected size of the answered vector for different sizes of the indexer tree. For a tree of $k = 10$ indexers, if each indexer contributes with $n = 80$ elements the answered vector has an expected length of 300 elements. The right side of the figure shows the anonymity loss of the first indexer inside the tree. In the last example ($k = 10, n = 80$), the indexer in the inner leaves of the tree was the source of an item 0.2 of the time. Since the anonymity set of the tree has a size of $k = 10$ elements, attackers learn that the inner indexers are the source of an item about twice as often as in the maximum anonymity scenario.

For $k = 50$, the scenario is similar: the maximum length of the answered vector of affine document profiles occurs when each node contributes with $n = 20$ items. In this case, the inner indexer is the source of an item with a probability of 0.05, when the maximum anonymity occurs at $1/k = 0.02$. In this case, the probability of an item to come from an indexer doubles the maximum anonymity scenario, and inner indexers in a tree of $k = 50$ indexers, when each contribute with $n = 20$ items, are as protected as if the tree has only $k^0 = 25$ indexers.

These figures show an interesting behavior. In a scenario with duplicated documents, increasing the number of contributions from each indexer has two effects (i) increases the probability that inner nodes are the source of a recommendation and (ii) decreases the number of items in the answer, since it makes more likely that documents are duplicated in the different indexers. Then, limiting the number of results that each indexer contributes to the answered vector enhances the anonymity of the indexers as well as increases the number of results that the customer gets. Unfortunately, maximizing one feature does not maximize the other, as the provided figures show. This interesting behavior should be a matter of future research.

These figures and the equations from this section can be used to determine the number of contributions from each indexer, the apparent size of the anonymity set and the expected size of the returned vector. Larger returned vectors enhance the efficiency of the system, since more affine documents are discovered during a query. But if the number of indexers in the indexer tree is not chosen accordingly, the anonymity loss of the indexers that first contribute to the answered vector may be unacceptably high.

6. Conclusions and future research

We introduced DocCloud, a recommender system that protects virtual machines and cloud providers from legal attacks. We established that *plausible deniability* was a necessary security service that a recommender system must provide, since it protects all participants of the system against legal attacks. We defined plausible deniability as the ability of a node to deny any knowledge of the document it is recommending to the system's users. We explored a probabilistic approach to provide plausible deniability.

We explored some mechanisms to protect the profiles that users store in indexers to ensure that these profiles have no information about the original profiles, but retain affinities between them so that queries can be answered correctly. We also defined a private block retrieval scheme that connects customers and recommenders. This scheme ensures that recommenders cannot identify the profile of the document they are providing to the user. This is not only a safeguard to protect the user's privacy; it also prevents recommenders from being prosecuted by aiding in the process of downloading a protected document and provides intermediate nodes the security service of oblivious routing.

Finally, we proposed the organization of databases in a tree-shaped structure to prevent identification of the source of the recommendation, and provided plausible deniability to databases. Furthermore, this tree structure lets the customer

download an item from the database without leaking the identity of the database that answered the query. Not even the database can decide whether or not it was answering a specific query. We explored two different assumptions for the distribution of document profiles inside the tree structure: a uniform distribution and a social distribution. The former is easier to analyze, but the latter is more similar to the organization of nodes in our recommender system. Finally, we provided an upper limit on the number of items that indexers must answer in order to provide optimal deniability inside the indexers tree.

There are some areas for improvement in this system. When a user gets the *URL* of an interesting document, they still has to contact another network to actually download the document. It is not clear whether or not the process of downloading can be separated from the process of selecting documents. For example, an attacker controls an indexer that forges special *URLs* in such a way that they are able to decide whether or not these are downloaded afterward. This way, they would be able to link a query to a user. This kind of attack was not addressed in this document.

A second open line of research involves the management of the social cloud. If cluster A is created only with users with similar profiles, then a ''representative'' profile may be calculated for the cluster, and it may be close enough to the individual descriptions of each user to unacceptably leak private information that can be used to learn the users' profiles. In this case, we propose that clusters should be created with users with several ''classes'' of profiles. At the same time, users may show different profiles according to their current interests, and join different clusters of the network at the same time. The impact of these ''multi-ethnic'' clusters on the efficiency of the system remains unclear. Additionally, although we were concerned about the protection of the user's privacy and introduced some mechanisms to provide this protection, we have not thoroughly analyzed the effect of these mechanisms on the efficiency of the recommendation process, and the amount of protection they provide.

Finally, in this paper we described a theoretical model to protect document indexers of a cloud system. We are testing the system and its proposed mechanisms using a real implementation to get an idea of the impact they have on the quality of the recommendations that users obtain from the system in large social networks. This approach is obviously limited to the quality of the assumptions that we force upon the user descriptions. An implementation on an actual social cloud will be the real test for this proposal. We are currently working on this implementation.

Acknowledgments

This work has been partially supported by the Spanish Government research Projects TAMESIS (TEC2011-22746) and ARES (CSD2007-00004) and by the Project granted with FEDER funds WoO (TSI-020400-2011-29).

References

[1] D. Achlioptas, Database-friendly random projections: Johnson–Lindenstrauss with binary coins, Journal of Computer and System Sciences (2003).
[2] A. Anglade, M. Tiemann, F. Vignoli, Complex-network theoretic clustering for identifying groups of similar listeners in p2p systems, in: RecSys '07: Proceedings of the 2007 ACM Conference on Recommender Systems, ACM, New York, NY, USA, 2007, pp. 41–48.
[3] S.K. Belle, M. Waldvogel, Consistent deniable lying: privacy in mobile social networks, in: Workshop on Security and Privacy Issues in Mobile Phone Use.
[4] O. Berthold, H. Langos, Dummy traffic against long term intersection attacks, in: Privacy Enhancing Technologies.
[5] J. Bian, Jigdfs: A secure distributed file system, in: IEEE Symposium on Computational Intelligence in Cyber Security.
[6] J. Canny, Collaborative filtering with privacy, in: Proceedings of the 2002 IEEE Symposium on Security and Privacy, 2002, pp. 45–57.
[7] K. Chard, S. Caton, O. Rana, K. Bubendorfer, Social cloud: cloud computing in social networks, in: Proceedings of the 3rd International Conferenceon Cloud Computing IEEE Cloud, 2010.
[8] C.W. Clifton, M. Murugesan, Providing Privacy Through Plausibly Deniable Search, 2009.
[9] enigmax, Cyberlocker Ecosystem Shocked as Big Players Take Drastic Action, 2012. <http://torrentfreak.com/cyberlocker-ecosystem-shocked-as-big-players-take-drastic-action-120123/>.
[10] P. Euster, R. Guerraoui, A.M. Kermarrec, L. Maussoulie, From epidemics to distributed computing, IEEE Computer 37 (2004) 60–67.
[11] C. Gentry, Z. Ramzan, Single-database private information retrieval with constant communication rate, in: Automata, Languages and Programming, vol. 3580/2005 of Lecture Notes in Computer Science, pp. 803–815.
[12] A. Glorioso, U. Pagallo, G. Ruffo, The social impact of p2p systems, in: X. Shen, H. Yu, J. Buford, M. Akon (Eds.), Handbook of Peer-to-Peer Networking, Springer, US, 2010, pp. 47–70.
[13] O. Hermoni, N. Gilboa, E. Felstaine, S. Shitrit, Deniability—an alibi for users in p2p networks, in: 3rd International Conference on Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008, pp. 310–317.
[14] J. Hernández-Serrano, J. Vera-del-Campo, J. Pegueroles, C. Gañán, Low-cost group rekeying for unattended wireless sensor networks, Wireless Networks (2012) 1–21.
[15] W. Hu, T. Yang, J.N. Matthews, The good, the bad and the ugly of costumer cloud storage, in: ACM SIGOPS Operating Systems Review, vol. 44, pp. 110–115.
[16] E.T. Jaynes, Probability Theory: The Logic of Science: Principles and Elementary Applications, Cambridge University Press, 2003.
[17] S. Lee, D. Lee, S. Lee, Personalized DTV program recommendation system under a cloud computing environment, IEEE Transactions on Consumer Electronics 56 (2010) 1034–1042.
[18] K. Liu, H. Kargupta, J. Ryan, Random projection-based multiplicative data perturbation for privacy preserving distributed data mining, IEEE Transactions on Knowledge and Data Engineering 18 (2006) 92–106. Senior Member-Kargupta,, Hillol.
[19] H. Ma, M.R. Lyu, I. King, Learning to recommend with trust and distrust relationships, in: Proceedings of the third ACM Conference on Recommender Systems, RecSys '09, ACM, New York, NY, USA, 2009, pp. 189–196.
[20] A. Machanavajjhala, A. Korolova, A.D. Sarma, On the (Im)possibility of Preserving Utility and Privacy in Personalized Social Recommendations, Technical Report arXiv:1004.5600, 2010.
[21] C.D. Manning, P. Raghadan, H. Schütze, An Introduction to Information Retrieval, Cambridge University Press, 2009.
[22] M. Mao, Ontology mapping: An information retrieval and interactive activation network based approach, in: The Semantic Web, vol. 4825 of Lecture Notes in Computer Science, pp. 931–935.

[23] B. Markines, C. Cattuto, F. Menczer, D. Benz, Andreas, Evaluating similarity measures for emergent semantics of social tagging, in: 18th International Conference on World Wide Web, pp. 641–650.
[24] D. Meyer, Acta rejected by Europe, leaving copyright treaty near dead 2012. <http://www.zdnet.com/acta-rejected-by-europe-leaving-copyright-treaty-near-dead-7000000255/>.
[25] R. Mortier, A. Madhavapeddy, T.W. Hong, D. Murray, M. Schwartzkopf, Using Dust Clouds to Enhance Anonymous Communication.
[26] R. Ostrovsky, I.W.E. Skeith, A survey of single-database private information retrieval: techniques and applications, in: Proceedings of the 10th International Conference on Practice and Theory in Public-Key Cryptography, PKC'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 393–411.
[27] P. Paillier, Public-key cryptosystems Based on Composite Degree residuosity classes, in: EUROCRYPT.
[28] S. Pearson, Y. Shen, M. Mowbray, A privacy manager for cloud computing, in: CloudComp 2009, Number 5931 in Lecture Notes in Computer Science, pp. 90–106.
[29] J. Pouwelse, J. Yang, M. Meulpolder, D. Epema, H. Sips, Buddycast: an operational peer-to-peer epidemoc protocol stack, in: Fourteenth Annual Conference of the Advanced School for Computing and Imaging.
[30] D. Rebollo, J. Forné, Optimized query forgery for private information retrieval, in: IEEE Trans. Inform. Theory, vol. 56. pp. 4631–4642.
[31] D. Rebollo-Monedero, J. Forné, E. Pallarès, J. Parra-Arnau, A modification of the Lloyd algorithm for k-anonymous quantization, Elsevier Information Science 222 (2013) 185–202.
[32] M. Reiter, A. Rubin, Crowds: anonymity for web transactions, in: ACM Transactions on Information and System Security.
[33] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, Grouplens: an open architecture for collaborative filtering of netnews, in: Conference on Computer supported Cooperative Work, ACM, New York, NY, USA, 1994, pp. 175–186.
[34] R. Schifanella, A. Panisson, C. Gena, G. Ruffo, Mobhinter: epidemic collaborative filtering and self-organization in mobile ad-hoc networks, in: ACM Conference on Recommender systems, ACM, New York, NY, USA, 2008, pp. 27–34.
[35] C. Sosa, B.C. Sutton, H.H. Huang, Picfs: the privacy-enhancing image-based collaborative file system.
[36] Stockholm District Court, Verdict b 13301-06, 17 April 2009, 2009.
[37] R. Thiagarajan, G. Manjunath, M. Stumptner, Finding experts by semantic matching of user profiles, in: 3rd Expert Finder Workshop on Personal Identification and Collaborations: Knowledge Mediation and Extraction.
[38] Trade European Commission, The anti-counterfeiting trade agreement (acta), 2010. <http://ec.europa.eu/trade/creating-opportunities/trade-topics/intellectual-property/anti-counterfeiting/>.
[39] D.T. Tran, S. Bloehdorn, P. Cimiano, P. Haase, Expressive resource descriptions for ontology-based information retrieval, in: Proceedings of the 1st International Conference on the Theory of Information Retrieval (ICTIR'07), 18–20th October 2007, Budapest, Hungary, pp. 55–68.
[40] US Department of Justice, Justice Department Charges Leaders of Megaupload with Widespread Online Copyright Infringement, 2012. http://www.fbi.gov/news/pressrel/press-releases/justice-department-charges-leaders-of-megaupload-with-widespread-online-copyright-infringement.
[41] USLegal, Plausible deniability law & legal definition, 2011. http://definitions.uslegal.com/p/plausable-deniability/.
[42] J. Vera-del-Campo, J. Hernández-Serrano, J. Pegueroles, Scfs: Design and Implementation of a Secure Distributed Filesystem, SECRYPT, 2008.
[43] J. Vera-del-Campo, J. Hernández-Serrano, J. Pegueroles, Profile-based searches on p2p social networks, in: The Ninth International Conferenceon Networks, ICN.
[44] J. Vera-del-Campo, J. Hernández-Serrano, J. Pegueroles, M. Soriano, Design of a p2p content recommendation system using affinity networks, Computer Communications 36 (2012) 90–104.
[45] J. Zhan, C. Hsieh, I. Wang, T. Hsu, C. Liau, D. Wang, Privacy-preserving collaborative recommender systems, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 40 (2010) 472–476.
[46] F. Zhang, J. Wang, J. Chao, Cross-system privacy preserving recommendation algorithm based on secure multi-party computation, Journal of Computational Information Systems 6 (2010) 3013–3021.