Private Peer-to-peer similarity computation in personalized collaborative platforms

Mohammad Alaggan

ANNÉE 2013

UNIVERSITÉ DE RENNES 1

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale Matisse

présentée par

Mohammad ALAGGAN

préparée à l'unité de recherche IRISA – UMR6074
Institut de Recherche en Informatique et Système Aléatoires
ISTIC

Private peer-to-peer similarity computation in personalized collaborative platforms

Thèse soutenue à Rennes le 16 Décembre 2013
devant le jury composé de :

Ludovic MÉ
Prof. Supélec / *Président*

Daniel LE MÉTAYER
DR INRIA / *Rapporteur*

Marc-Olivier KILLIJIAN
CR CNRS / *Rapporteur*

Luís RODRIGUES
Prof. Univ. de Lisboa / *Examinateur*

Anne-Marie KERMARREC
DR INRIA / *Directrice de thèse*

Sébastien GAMBS
MdC Univ. Rennes 1/INRIA / *Co-directeur de thèse*

> For if we are observed in all matters, we are constantly under threat of correction, judgment, criticism, even plagiarism of our own uniqueness. We become children, fettered under watchful eyes, constantly fearful that — either now or in the uncertain future — patterns we leave behind will be brought back to implicate us, by whatever authority has now become focused upon our once-private and innocent acts. We lose our individuality, because everything we do is observable and recordable.
>
> — Bruce Schneier, *The Eternal Value of Privacy*

Acknowledgment

This research is supported by the Gossple ERC Starting Grant number 204742.

Contents

1 Introduction

How people interact with the Internet has changed fundamentally during the past decade as the web is no longer a read-only service. It turned instead into a collaborative platform, with which the users can interact and modify by expressing their preferences and circulating news.

Moreover, social networks, like Facebook and Twitter, were designed to provide a user-centric social experience on the web. Users are the main content producers and their social acquaintances are the content consumers. The contents they produce, beside their personal information (date of birth, job, hobbies, alma mater, and so on), are typically of personal nature: their pictures, other pictures they liked, pieces of news they were interested in, and so on. Acquaintances and friends are usually chosen explicitly by the users, and a friendship relation can be symmetric (like in Facebook) or asymmetric (like in Twitter).

Many other websites do not depend on an explicit social network, but still allow users to express their interests. Such as Delicious, a collaborative bookmarking platform, and Digg, a collaborative news aggregator. The information provided by the users could be leveraged to extract trends and hot topics and then personalized services could be provided to the users according to their preferences [1, 2, 3, 4, 5]. Moreover, similarities between the interests of users could be identified and used to recommend items using collaborative filtering. To leverage this similarity computation is at the heart of any system of personalization.

Privacy. Even though social networking has the potential to enhance the users' experience, they are also surrounded with privacy concerns. Users share their personal information and interests, some of which they might want no one to know. For instance, a Digg user might express her opinion to cancer-related news items in order to get more of these items in her personalized news feed, but he might not want someone else to know he is interested in cancer-related news because he considers her health information to be sensitive. The website hosting the user's private data is expected to implement proper access-control mechanisms, for instance by allowing the user to control who can access her data. Nonetheless, failure of such access-control mechanisms is not uncommon, including security breaches. Moreover, the recent National Security Agency (NSA) PRISM program [6] gives NSA access to users' private information in several websites, including Facebook and Google.

In addition, since friends influence each others, users' private data can be harnessed for advertising by displaying them to the user's friends [7, 8]. For example, the website could advertise a specific movie by showing some user that his friends, Alice and Bob, liked it, which is may be considered a privacy violation for Alice and Bob. Even a stray internet user, given public and seemingly-harmless item recommendations and some auxiliary information, can still mitigate the access-control mechanism and infer the private information of users. For instance, Calandrino, Kilzer, Narayanan, Felten and Shmatikov [9] showed that just by observing the public recommendations of items on Amazon, they could infer private purchases of some users. In general, personalization often implies some form of privacy breach as the user's private data has to be used in the process [10].

Furthermore, even if the user did not associate her true identity with her profile on a particular website, it is still possible, given the private information he submitted

to the website, to link her user account to another website to which he has provided her true identity [11], thus successfully inferring her true identity.

Peer-to-peer. From that we can see that delegating one's private data to third parties for storing or processing is not always safe as third parties cannot always be trusted. Instead, it is preferable to deploy the social networks and applications over Peer-to-Peer (P2P) overlay networks. Unlike classical centralized systems that are composed of a central server processing clients' requests in which clients communicate solely with the server and are oblivious to the existence of other clients in the system, P2P systems are distributed and the service is provided via the explicit and proactive collaboration among clients. Each client is generally a machine in the system that shares part of the load of providing the service. The P2P overlay network is the graph overlaid on the physical network topology, and whose nodes represent clients and edges represent a direct collaboration between two clients (typically means that they can communicate directly). Clients in P2P systems do not usually have a global knowledge of the network and are only aware of a small portion of the total number of clients, hence P2P systems are notorious for their scalability and robustness properties [12].

Since the applications we mentioned are user-centric, P2P system are a natural candidate for implementing these application. In particular, each user alongside with his personal information amounts to one node in the P2P network. This way, the user will store his data on his own machine, having full control over it, and does not need to trust anyone other than herself (given, of course, that his software and hardware setup is trusted). Since the number of users of social networks is quickly growing, the need to scale accordingly also gives a strong incentive for deploying them as decentralized and P2P applications [13, 14], however, at the cost of increased system complexity and synchronization overhead.

Similarity. Within this distributed framework, efficiently computing recommendations or personalized search results while keeping network traffic reasonable is of paramount importance. Many existing P2P systems achieve this by linking a user only to other users with whom he shares common interests [1, 15, 16, 17], (*i.e.* with which his similarity is relatively higher than other users.) Moreover, similarity between users can be cast as a form of a distance measure, which is a fundamental primitive in many data mining applications. Distances can be used to perform the tasks for clustering, nearest neighbors, or of direct interest to our context, collaborative filtering, which allows providing personalized recommendations.

Therefore, having an primitive that provides user-to-user similarity is key to provide social personalization in P2P collaborative platforms. This thesis is devoted to constructing such a primitive in a way that provides privacy for users, while still being efficient in terms of communication and computation costs. In the following section we further detail the privacy aspects.

1.1 Privacy

In order to perform collaborative filtering, information involving multiple users' data need to be combined. In our case, this information is the user-to-user similarity. The two users involved in computing their pair-wise similarity will be engaged in a distributed protocol in order to carry out that task. However there are two complimentary aspects when privately carrying out such a protocol. Consider first the context in which the privacy is defined. In particular, there is 1) a function to compute, 2) the inputs of the function, whose privacy has to be protected, and 3) the resulting value of the function (output). If there was only one party holding the input and performing the computation, there would be no privacy issue in general. Problem arises when there are several parties holding different parts of the input, in which the part held by a given party is considered private to him.

A seminal example of Yao [18] was about two millionaires, each of which holding his own amount of wealth as private input and they want to compute the function "greater-than" to know which one of them is richer. Ultimately, the two parties in this example have to be engaged in a protocol in order to combine their inputs and compute an output, which is then revealed to both parties. During this process, a particular party observes 1) the protocol transcript and 2) the output of the function. The traditional definition of secure multiparty computation [19] is only concerned about guaranteeing that the protocol transcript does not leak information about the input. That is, to an adversary observing the protocol transcript, no information about the private input of the honest users should be leaked. In that definition, there is no guarantee about what the output value (as opposed to the protocol transcript alone) may reveal about the input values.

In contract to that definition, privacy-preserving data mining [20] (such as [21]), would be concerned only about what the output value may reveal about the input, and not about how the output itself is computed. For example, the data may be entrusted to a central curator who performs the required computation on them before releasing the output publicly. In such setting, instead of guaranteeing the exact answers as output, a slight deviation from the true answer is allowed in order to gain in terms of privacy.

It is thus natural to aspire to immunize *both* the protocol transcript and the output value from leaking to get the best of both worlds. General constructions for secure multiparty protocols for implementing any functionality exist [22, 23, 24] so it would be straight-forward to implement one of the many differentially-private mechanisms using such a general construction. However, such general constructions are very expensive and impractical. Specialized protocols may be constructed for specific functions to achieve the task efficiently [25, 26]. This thesis extends this line of work to the world of distributed systems, which raises a unique set of challenges, such as dynamic large networks, offline peers, continuous computation (detailed later as the privacy budget issue), and heterogeneity of privacy expectations across users.

1.2 Differential privacy

An ideal privacy preserving function is one whose output does not leak any information about its input. In particular, what could be learned after releasing the function's output could also trivially be learned without observing the function's output [27]. However, as long as the output provides information about the input (*i.e.*, utility as measured by min-entropy [28]), it has been long suspected [29], demonstrated [30], and finally proven in [27] that this ideal is impossible to achieve. In particular, Dwork and Naor [27] proved that if the output of the privacy preserving function has n bits of min-entropy, then an n bit privacy breach can occur, given that the adversary has some auxiliary information. In the same paper Dwork and Naor suggest, as a solution, to move from the previous *absolute* concept of privacy to a *relative* one, which they coined as *differential* privacy. Moreover, the new notion does not depend on any assumptions regarding the auxiliary knowledge of the adversary. The property of being differentially private is a property of the function alone and not of its input, nor the adversary, which resulted in a strong and theoretically appealing framework that was widely adopted afterwards in the privacy community.

The spirit of differential privacy is that a small change in the input should induce no more than a small change also on the distribution of the output. The exact definition of differential privacy depends on how the notion of "small" is defined. In the traditional differential privacy literature, the function to compute takes vectors as input; either a vector of zeros and ones, or a vector of records of individuals' data. In this context, a small change in the input amounts to a change in at most one position in the vector. For example, for vectors of zeros and ones, a small change is represented by a modification of Hamming distance by 1. This notion has been generalized to arbitrary metrics in [31].

Differential privacy started in the context of statistical disclosure control [27]. In this setting, the function's inputs is a vector of records of individuals' data, and the small change in input is the presence or absence of the record of a single individual. Hence, a user contributing his data, for instance, is assured that the output of the function represents global trends among all users and only very little amount of information specific to him. Thus, any resulting privacy breach (if any) has minimal impact to his own, personal, privacy. In addition, even the mere fact of that he contributed to the data at all (called *membership privacy*), is protected as well. Moreover, this guarantee still holds even if the adversary knows all the other users who contributed their data but her.

The benefits to user's privacy in the statistical database model mentioned in the previous paragraph is clear. However, it requires some clarification to show the same in case of *micro-data* in which the input represents the data of a *single* individual, which is the situation assumed throughout this thesis. In his setting, a small change relative to which differential privacy is defined is the change in the presence or absence of one item in the individual's profile. Differential privacy will guarantee that any information that could be deduced from the output did not depend on a particular item. Therefore, the user has no reason to worry about whether to add one particular item to his profile or not. However, global trends about the user's profile, such as the broad categories of his interests may be revealed.

1.3 Contributions of the thesis

The goal of this thesis is to address the problem of privately computing similarities between users in P2P collaborative platforms. Our contributions, after the background in Chapter 2, are described in details.

Our first contribution in Section 3.2.2 is addressing the privacy budget issue (*cf.* Section 2.3.1) which sets an upper bound on the number of similarity computations a peer is allowed to engage in if he wants to achieve differential privacy. We introduced of the concept of bidirectional anonymous channel, which provides anonymity for the sender and for the recipient from each others, at the same time. Two protocols for implementing such a channel are described, one for passive adversaries and another one for active adversaries.

A two-party cryptographic protocol for deciding whether the similarity of two peers is above a given threshold is described in Section 3.4. The protocol utilizes several cryptographic primitives and sub-protocols in order to preserve the secrecy of the inputs, which correspond to the private profiles of the two peers involved, and releases nothing more than one bit of information.

We then describe a two-party protocol for securely computing similarity, with differential privacy guarantees, in Section 3.5. Two protocols for two-party distributed noise generation are presented. The impact of differential privacy to the threshold similarity protocol, as measured by false positives and false negatives is theoretically analyzed in Section 3.6.

Next, in Chapter 4 we describe a variant of differential privacy that can provide diverse privacy guarantees for different peers with different privacy expectations. Moreover, a single user can choose different privacy levels for different items in his private profile. A mechanism satisfying this new definition is constructed and evaluated.

Afterwards in Chapter 5, we introduce a non-interactive differentially private mechanism for estimating similarity between users. This mechanism avoids the need for the bidirectional anonymous channel while being more efficient in terms of computational cost at the same time.

Moreover, in Chapter 5 we also provide a way to choose a value for the privacy parameter ε in terms of utility and privacy. Utility is measured through a theoretical bound on the deviation of the similarity value while privacy is analyzed through two attacks. The value of ε should then be chosen such as it provides acceptable utility while at the same time evades the two attacks.

Finally, in Chapter 6 we try to investigate how differential privacy behaves with respect to an adversary with different kind of, but more realistic, side knowledge than is typically assumed for differential privacy. We design two attacks and evaluate their success.

2
Background

Abstract

In this chapter we describe the background and preliminaries for the rest of the thesis. We provide the system model then give a detailed description of the GOSSPLE peer-to-peer semantic clustering protocol, which we use for evaluating our techniques. Then we present differential privacy and the privacy budget issue. Finally, we also describe the datasets used for evaluation throughout the chapters.

2.1 System model

Through out this thesis we consider a decentralized collaborative platform composed of users, each user is a physical node in the network. A user has a profile which expresses his interests in particular items by assigning a binary value to each item. If the user likes an item, it is assigned the value 1 while if the user did not like the item, or has never seen the item before, it is assigned the value 0. A user's profile is the collection of these assignments into a binary vector of n bits, where n is the total number of items in the system. For example, if the system is Delicious (a collaborative platform described later in Section 2.4), then there are n URLs. If $n = 3$, then a user profile may look something like $(1, 0, 1)$ if he likes the first and third items.

2.2 Context of the thesis: Decentralized collaborative filtering

In this section we provide an overview of the system in which our work takes place.

2.2.1 Overview of the system

The system on which our work is based is the GOSSPLE system [1]. The goal of GOSSPLE is to cluster peers according to their interests. Such that neighbors are likely to have shared interests. In his PhD thesis, Afshin Moin described techniques for implementing collaborative filtering on top of GOSSPLE, given that peers sharing similar interests are connected to each other [32]. We describe an overview of how GOSSPLE clusters peers according to their interests in the following.

GOSSPLE depends on two sub-protocols, the first is a random peer sampling protocol [12, 33] that supplies each peer with a locally uniform random stream of other peers. The other one is a semantic clustering protocol that consumes the stream of random peers to perform clustering. Each of these sub-protocols maintain a set of neighbors to the peer, which we call a view. The first protocol maintains the *random view* ρ while the second maintains the *clustering view* c. The clustering view of a peer is supposed to contain a list of peers similar to him; the list being of constant length ℓ (usually $\ell = 10$). The overall protocol operates in rounds as described in the next paragraph, and is an instance of what is known as *iterative gossip-based protocols* [34].

In the following, let $c_i^{(r)}$ be the clustering view of peer i at round r, and $\rho_i^{(r)}$ its random view at the same round. We describe Algorithm 1 of GOSSPLE [1]. At round r, every peer i selects the oldest peer j in his clustering view $c_i^{(r)}$, or from $\rho_i^{(r)}$ if $c_i^{(r)} = \varnothing$. Then let $c_i' = c_i^{(r)} \cup c_j^{(r)} \cup \rho_i^{(r)}$. GOSSPLE uses a sophisticated group similarity measure, which takes into account the combined utility of the entire view. The sophisticated measure they use is computationally inefficient and needs special heuristics to efficiently compute it. Not only the heuristic they provide very difficult to compute in a secure multi-party protocol, it also requires the interaction of several peers and therefore poses a serious challenge to the anonymity requirements needed to solve the privacy budget issue (*cf.* Section 2.3.1). Therefore, we use a pairwise similarity metric instead and design the protocol such that the most similar peers are kept in the clustering view. In particular, we set $c_i^{(r+1)} \subset c_i'$ such that $\left|c_i^{(r+1)}\right| = \min(\ell, |c_i'|)$ and if $a \in c_i^{(r+1)}$, then $\mathsf{sim}(a,i) \geqslant \mathsf{sim}(b,i)$ for all $b \in c_i' \setminus c_i^{(r+1)}$, for a specified similarity measure sim. This thesis is concerned about how to privately compute such similarity measure, that is, while protecting the privacy of the peers' profiles.

At the end of the clustering phase, *i.e.* when the views converge after a sufficient number of rounds (around 20 rounds), each peer will have chosen the ℓ most similar peers to him. Denote the clustering view of peer i at this stage $c_i^\star$. In case the similarity computation were not perturbed in any way, we call this view the *perfect view*.

Similarity. Cosine similarity [35] is used by GOSSPLE to assess the similarity between two binary vectors:

$$\mathsf{cos\_sim}(\boldsymbol{d}, \boldsymbol{d}') = (\boldsymbol{d} \cdot \boldsymbol{d}')/\|\boldsymbol{d}\|\|\boldsymbol{d}'\| \ . \tag{2.1}$$

The value of the cosine similarity ranges from 0 to 1, in which 0 means nothing in common and 1 means identical binary vectors. The cosine similarity can be expressed in terms of inner products by noticing that $\|\boldsymbol{d}\|^2 = \boldsymbol{d} \cdot \boldsymbol{d}' + \boldsymbol{d} \cdot (1 - \boldsymbol{d}')$ and $\|\boldsymbol{d}'\|^2 = \boldsymbol{d} \cdot \boldsymbol{d}' + (1 - \boldsymbol{d}) \cdot \boldsymbol{d}'$, for binary vectors $\boldsymbol{d}, \boldsymbol{d}'$. Although the similarity metric used throughout this thesis, our protocol will work with any other binary metric as well. In particular, since any binary similarity measure between two binary vectors can ultimately be computed based on inner products of the original vectors or their negation [36], we focus on computing the inner product function instead and derive cosine similarity from it. The inner product of two binary vectors $\boldsymbol{d}, \boldsymbol{d}' \in \{0,1\}^n$ is $\sum_{i=1}^{n} \boldsymbol{d}_i \times \boldsymbol{d}'_i$.

The focus of this thesis is mostly on how to compute similarity while preserving privacy. For instance, in the *baseline* protocol just described, the similarity is always computed based on the original unperturbed profiles. In Chapter 3 and 4, perturbed similarity is computed in an interactive cryptographic protocol, while in Chapter 5 it is computed in a non-interactive manner. Moreover, in the interactive protocols, if the perturbed similarity is below a specific threshold it will not be released (*cf.* Section 3.4); because peers with low similarity will not need be in each other clustering views and therefore there will be no need to use the similarity value to keep

the most similar peers. In the following section we describe how our experiments are designed to evaluate the utility of this system.

2.2.2 Conducting experiments

The experiment is conducted by simulating GOSSPLE. Each node i randomly partitions its profile p_i into two disjoint subsets, the *training* subset t_i and the *search* (or *evaluation*) subset s_i, such that $t_i \cup s_i = p_i$ and $t_i \cap s_i = \varnothing$ and $|s_i|/|t_i| \approx 0.9$ (the value 0.9 has been chosen to match [1]). The experiment is split into two steps: the clustering (*i.e.*, training) phase and the evaluation phase. The training phase executes GOSSPLE as described in the previous section but were the system is run on the training profiles t instead of their actual profiles p. After a fixed number of cycles the evaluation phase begins. Two metrics are going to be used to evaluate utility. The first one is the ratio, for a peer i, between the cumulative similarity of the clustering view $\sum_{j \in c_i^r} \mathsf{sim}(i,j)$ to the cumulative similarity of the perfect view $\sum_{j \in c_i^\star} \mathsf{sim}(i,j)$, where r is the cycle at which the evaluation phase started. The other one is described in the next paragraph.

The clustering view provides utility if it is able to predict the items in the search subset. This ability is useful for searching and collaborative filtering. Measuring the ability to predict items is commonly evaluated using both the *recall* and *precision*. The recall is the ratio of the training items successfully predicted, while precision is the ratio of the correctly predicted items to all predicted items (*i.e.*, including those which were incorrectly predicted). A prediction algorithm that predicts all possible items ($\{1, \ldots, n\}$), regardless of its input, will give a recall of 1 but will have a very low precision. However, because the clustering view contains only a constant number of neighbors $\ell = |c_i^\star|$, predicting all possible items is not possible (assuming that for all j, $|t_j| = o(n)$, then $|\bigcup_{j \in c_i^\star} t_j| = o(n)$ too). Hence, the recall alone will be sufficient for the evaluation [1, Section 3.1].The profiles are split to training and search sets such that elements in the search set are guaranteed to exist in the training set of at least one other peer. That is, for all i, $x \in s_i$ implies that $x \in t_j$ for some $j \neq i$. More formally, the recall r_i for the peer i is:

$$r_i = \frac{|s_i \cap (\cup_{j \in c_i^\star} t_j)|}{|s_i|} \quad . \tag{2.2}$$

The experiment then will report the average recall over all peers $r = \mathbb{E}_i[r_i]$.

2.3 Differential privacy

Differential privacy has been introduced by Dwork in 2006 [37] as an alternative definition to privacy in response to an impossibility result presented in the same paper asserting the impossibility of absolute privacy as long as there is a non-negligible utility. It was meant to describe privacy as a property of the release mechanism instead of the data or the adversary's *a priori* or *a posteriori* knowledge of the data. To achieve this objective, differential privacy was designed so as to render the release mechanism insensitive to changes in a single field (a single bit in case of binary

vectors). In particular, if a single bit changes in the input profile, the distribution of the mechanism's outputs should not differ much. Thus, the release mechanism should reflect only global features of the whole profile rather than of individual bits (in the original paper a profile was a database and an individual bit was a row in this database corresponding to real person). First, we need to define the notion of adjacent (or neighboring) profiles.

Definition 2.1 (Neighboring profile [38])**.** Two profiles $\boldsymbol{d}, \boldsymbol{d}^{(i)} \in \mathbb{R}^n$ are said to be *neighbors* if there exists an item $i \in \{1, \ldots, n\}$ such that $\boldsymbol{d}_k = \boldsymbol{d}_k^{(i)}$ for all items $k \neq i$. This neighborhood relation is denoted by $\boldsymbol{d} \sim \boldsymbol{d}^{(i)}$.

An equivalent definition states that $\boldsymbol{d}$ and $\boldsymbol{d}^{(i)}$ are neighbors if they are identical except for the i-th coordinate. For instance, the profiles $(0, 1, 2)$ and $(0, 2, 2)$ are neighbored while the profiles $(0, 1, 2)$ and $(0, 2, 3)$ are not. The definition applies to binary vectors as well. In case the coordinate of the item on which the two profiles differ does not matter it may not be mentioned, and we will denote the relation as $\boldsymbol{d} \sim \boldsymbol{d}'$ instead. Differential privacy can be defined formally in the following manner.

Definition 2.2 (ε-differential privacy [37])**.** A randomized function $\mathcal{M} : \mathbb{R}^n \to \mathcal{R}$, for some set $\mathcal{R}$, is said to be ε-differentially private, if for all neighboring profiles $\boldsymbol{d} \sim \boldsymbol{d}' \in \mathbb{R}^n$ and all $\mathcal{S} \subset \text{Range}(\mathcal{M})$:

$$\Pr[\mathcal{M}(\boldsymbol{d}) \in \mathcal{S}] \leqslant \exp(\varepsilon) \Pr[\mathcal{M}(\boldsymbol{d}') \in \mathcal{S}] \ ,$$

in which the probability is taken over the randomness of $\mathcal{M}$, and exp refers to the exponential function $x \mapsto \sum_{n=0}^{\infty} x^n/n!$.

The ε parameter represents the privacy level provided, but it is still an open research question to really understand its semantics [39, 40]. In general however, the lower the value of ε is, the better the privacy guarantees are.

Differential privacy aims at reducing the impact that any single coordinate of the profile can have on the output of a function. The maximal magnitude of such impact is captured by the notion of *global sensitivity*.

Definition 2.3 (Global sensitivity [38])**.** The global sensitivity $S(f)$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the maximum absolute difference obtained on the output of f over all neighboring profiles:

$$S(f) = \max_{\boldsymbol{d} \sim \boldsymbol{d}' \in \mathbb{R}^n} |f(\boldsymbol{d}) - f(\boldsymbol{d}')| \ . \tag{2.3}$$

The maximal impact that a particular coordinate in the input vector can induce is measured by the *modular sensitivity* instead.

Definition 2.4 (Modular sensitivity [41])**.** The modular sensitivity $S_i(f)$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the maximum absolute difference obtained on the output of f over all profiles that are neighboring on the item i:

$$S_i(f) = \max_{\boldsymbol{d} \sim \boldsymbol{d}^{(i)} \in \mathbb{R}^n} |f(\boldsymbol{d}) - f(\boldsymbol{d}^{(i)})| \ . \tag{2.4}$$

Modular sensitivity was implicitly introduced in [41, Lemma 1]. In a nutshell, the modular sensitivity of the item i reflects the maximum difference that i, in particular, can cause to the value of the function f while keeping the values of all other items fixed. Notice that $S(f) = \max_i S_i(f)$.

Dwork, McSherry, Nissim, and Smith proposed a technique called the *Laplacian mechanism* [38] that achieves ε-differential privacy by adding noise, proportional to the global sensitivity of a function, to its output. The noise is distributed according to the Laplace distribution with PDF $x \propto \exp(-\varepsilon|x|/S(f))$.

Theorem 2.1 (Laplacian mechanism [38])**.** *For a function $f : \mathbb{R}^n \to \mathbb{R}$, a randomized mechanism $\mathcal{M}$ achieves ε-differential privacy if it releases on input $\boldsymbol{d}$*

$$\mathcal{M}(\boldsymbol{d}) = f(\boldsymbol{d}) + \mathsf{Lap}\left(0, S\left(f\right)/\varepsilon\right) \quad , \tag{2.5}$$

in which $S\left(f\right)$ is the global sensitivity of the function f and $\mathsf{Lap}\left(0, \cdot\right)$ a randomly generated noise according to the Laplacian distribution with zero mean and variance $2S\left(f\right)^2/\varepsilon^2$.

2.3.1 Privacy budget

Consider a randomized, ε-differentially private mechanism $\mathcal{M}(\boldsymbol{d})$ for $\boldsymbol{d} \in \mathbb{R}^n$. We can refer to the random coins used by $\mathcal{M}$ as r, then we can instead refer to the deterministic mechanism $\mathcal{M}(\boldsymbol{d}; r)$. (Technically, $\mathcal{M}$ is a probabilistic function, and r should be an incompressible binary string of length $p(n)$ bits, in which p is a function bounding from above the number of steps $\mathcal{M}$ takes on inputs of length n.) The following lemma is central to differential privacy.

Lemma 2.1 (Composition and post-processing [42, 43])**.** *If $r_1, \ldots, r_k$ are k independent random strings and $a \in \{0,1\}^n$ then the function*

$$\mathcal{M}^{(k)} : \boldsymbol{d} \mapsto g(\mathcal{M}(\boldsymbol{d}; r_1), \ldots, \mathcal{M}(\boldsymbol{d}; r_k)) \quad , \tag{2.6}$$

*for any randomized function g that does not depend on $\boldsymbol{d}$, is $(k\varepsilon)$-differentially private (*i.e.*, less private than $\mathcal{M}$ if $k > 1$).*

Here, k represents the number of times the mechanism $\mathcal{M}$ is computed on the same input $\boldsymbol{d}$. In our case, this is the number of times the user computes his similarity with another user. If $k\varepsilon$ is not bounded, complete disclosure occurs. Therefore, either k has to be bounded, setting an upper bound on the number of users he is able to compute similarity with, or ε has to be set negligibly small, practically destroying the utility of the resulting similarity. This is known as the privacy budget issue, in which each computation of the mechanism on the same input "spends" privacy from a fixed total privacy budget. The fixed privacy budget in this case should be finite, in particular $k\varepsilon$ is the budget that should not be exceeded. A third solution is proposed in Chapter 3 to get around this problem, through the notion of bidirectional anonymous channel which guarantees that different computations cannot be linked together. Another solution which uses non-interactiveness, and thus the function is only needed to be computed once and for all, is used in Chapter 5 as

	# of users	# of items	average profile size	sparsity
Delicious	500	51453	135	0.26%
Digg	500	1237	317	25%
Survey	120	196	68	33%

Table 2.1 – Datasets characteristics

well. In particular non-interactive differential privacy [44]. Two non-interactive differentially private mechanisms has been specifically designed for estimating pair-wise similarity and distance. The first is the BLIP mechanism [40] and the second is the Johnson-Lindenstrauss mechanism [45]. The former is based on randomized-response and is designed to estimate inner product. The latter is based on sketching (using dimensionality-reduction) and is designed to estimate Euclidean distance. However Euclidean distance can be converted to inner product[1], thus it can also implement any binary similarity metric such as cosine similarity.

2.4 Datasets

We use three datasets to evaluate our approaches. They come respectively from Delicious, Digg, and a survey conducted within our lab. About 120 users participated in the survey and submitted their feedback (in forms of like/dislike) on approximately 200 news. Therefore, in the survey dataset a user's profile consists of the news he has liked, while for the Digg dataset a profile consists of the items that a user has forwarded to others users. Finally, in the Delicious dataset, the profile of the user consists of the items he has tagged. The characteristics of these datasets are briefly summarized in the Table 2.1. The sparsity is number of likes in the dataset divided by the number of users times the number of items. A low value for the sparsity means that on average, a user has expressed his preferences towards a small fraction of the items. More importantly, the sparsity indicates how many items does any two users drawn at random share, and a lower value implies that it is harder for a user to find good neighbors who share many items with him. Moreover, datasets with low sparsity are likely to produce low recall.

- *Delicious dataset.* Delicious (`delicious.com`) is a collaborative platform for keeping bookmarks in which users can tag the URLs of websites they liked. The Delicious dataset consists in the profiles of 504 users, a profile being a set of URLs that the user has tagged. The total number of URLs in the collective set of users' profiles is over 50000 URLs. In such a setting, the problem of similarity computation arises naturally, when providing personalized services such as the recommendation of URLs drawn from the ones tagged in Delicious. For the sake of simplicity, in the experiments conducted, each URL was assigned a unique identifier in the range of $\{1, \ldots, 50000\}$, in order to handle identifiers as integers instead of URL strings. The average size of a profile is 135 URLs.

[1] Via the identity $2\langle x, y\rangle = \|x\|_2^2 + \|y\|_2^2 - \|x - y\|_2^2$.

- *Digg dataset.* The dataset consists of 531 users of Digg (`digg.com`), a social news website. The profile of these users is composed of the news that they have shared over a period of 3 weeks in 2010. All the users considered have shared more than 7 items per week and the dataset contains 1237 items, each of which has been shared by at least 10 users. The average size of a profile is 317 items.

- *Survey dataset.* Around 196 randomly chosen news on various topics have been shown to 113 colleagues and relatives, who have then submitted their opinion in terms of like/dislike for each news. The average size of the profile is 68 (each user have responded to each and everyone of the 196 pieces of news; 68 represents the number of those pieces of news which was *liked*.)

3

The Interactive Case

Abstract

In this chapter we address the challenges of computing a differentially-private function in a distributed setting, which include the privacy budget issue, distributed noise generation, and carrying out similarity computation without having to trust a third party.

This chapter has been published as M. Alaggan, S. Gambs, and A.M. Kermarrec, "Private similarity computation in distributed systems: from cryptography to differential privacy," in *Proceedings of the 15th International Conference On Principles Of Distributed Systems (OPODIS'11)*, ser. Lecture Notes in Computer Science, A. Fernández Anta and G. Lipari and M. Roy, Ed., vol. 7109. Toulouse, France: Springer, 13–16 December, 2011, pp. 357–377.

3.1 Introduction

In this chapter we describe a two-party cryptographic protocol allowing two peers to compute the differentially-private similarity between their profiles without any of them observing the profile of the other and without relying on a trusted third party. Moreover, the protocol will abort if the differentially-private similarity between the two peers is lower than a predetermined threshold. While we focus on cosine similarity, our method is generic enough to be applied to other binary similarity metrics as long as they depend on inner product and simple arithmetic [36].

We study the impact of the differential privacy (which requires the addition of random noise) on the utility of the similarity measure, both through a theoretical analysis of the false negatives and experimental validation of the recall in Gossple.

The work presented in this chapter combines existing and well known cryptographic techniques and differential privacy mechanisms in the context of distributed similarity computation. Our main contribution besides the threshold similarity, is addressing the unique challenges of implementing differential privacy in a large and dynamic P2P system.

For instance, we acknowledge the difficulty of handling the privacy budget issue (*cf.* Section 2.3.1) in such large scale network and introduce the bidirectional anonymous channel primitive. We also provide several methods for distributed noise generation.

This chapter is organized as follows. First, Section 3.2 provides the required background and some preliminaries. Then, Section 3.3 reviews related work before introducing in Section 3.4 the threshold similarity protocol and prove its security with respect to a passive adversary. Afterwards in Section 3.5, we describe differentially-private protocols for the exact and threshold similarity. We analyze the utility in Section 3.6 and validate with experiments in Section 3.7. Finally, we conclude in Section 3.8.

3.2 Background

In this section we review the relevant cryptographic notions and describe the bidirectional anonymous channel.

3.2.1 Cryptographic background

Adversary model. In this chapter, we want privacy against a computationally-bounded *passive adversary* [19] (also sometimes called *semi-honest* or *honest-but-curious*) that can control a fraction of the peers. In this model (contrary to the active one), peers follow the protocol and do not deviate from its specifications. However, the adversary may try to infer as much information as possible from the inputs of peers it controls and from the protocol transcript (the set of messages exchanged during the protocol's execution). Furthermore, we assume the communication channels between peers are private. The number of peers controlled by the adversary is only relevant while analyzing the anonymity guarantee provided by gossip-on-behalf (explained later) as described in [1].

Definition 3.1 (Privacy with respect to a passive adversary [46])**.** A multi-party protocol is said to be *private with respect to a computationally-bounded passive adversary* controlling a peer (or a collusion of peers), if the adversary cannot learn, except with negligible probability, more information from observing the protocol's transcript than it could learn from its own input, the inputs of the peers it controls, and the output of the protocol.

Secure multi-party computation. General feasibility results showing the possibility of constructing a secure multi-party protocol for any functionality exist [18, 22, 23]. These feasibility results could be applied blindly to get a multi-party protocol for any differentially private mechanism such as the Laplacian mechanism [38]. However, such general results are extremely inefficient for practical use. For example, the garbled circuit technique [18] works by constructing an encrypted binary circuit of the desired functionality. Each binary gate in this circuit has two inputs and four possible outputs. The outputs are given in encrypted form and the gate's inputs are the keys to decrypt the correct output, which is in turn the key to next gate and so forth until the output gate. Instead, there are other techniques which are more suitable for practice, albeit they need to be tailored for specific functionalities. Such techniques are divided into two broad categories: *homomorphic encryption* and *secret-sharing* schemes [47]. Secret sharing is more efficient than homomorphic encryption and does not need cryptographic assumptions, providing privacy even from computationally-unbounded adversaries. However, it requires more than two parties so it is not suitable for our purposes, so we employ homomorphic encryption instead.

Homomorphic encryption. A homomorphic encryption is an encryption that allows computations to be performed on the encrypted text, without decrypting it. For example, an additively-homomorphic encryption has a function that takes as

input two encrypted numbers and their public key, and outputs another encrypted number, which if decrypted will give the sum of the two original ones. It may additionally allow multiplication with unencrypted numbers, an operation called scalaring.

Definition 3.2 (Homomorphic cryptosystem)**.** Consider a public-key (asymmetric) cryptosystem where

1. $\mathsf{Enc}_{pk}(a)$ denotes the encryption of the message a under the public key pk, and
2. $\mathsf{Dec}_{sk}(a) = a$ is the decryption of this message with the secret key sk[1].

This cryptosystem is said to *additively* homomorphic if there is an efficient operation $\oplus$ on two encrypted messages such that $\mathsf{Dec}(\mathsf{Enc}(a) \oplus \mathsf{Enc}(b)) = a + b$. Moreover, such an encryption scheme is called *affine* if there is also an efficient scalaring operation $\odot$ taking as input a cipher-text and a plain-text such that $\mathsf{Dec}(\mathsf{Enc}(c) \odot a) = c \times a$.

In addition to the previous elementary operations which can be carried out locally by one peer, there are more sophisticated techniques for which multi-party protocols exist, but require the active cooperation of more than one peer. For example, the operation of determining which of two encrypted numbers is greater than the other, without revealing any other information about the numbers [48, 49, 50]. Other examples include the multiplication of two encrypted numbers [51], or extracting the least significant bit [49], still in an encrypted form, out of an encrypted integer.

Paillier's cryptosystem [52] is an instance of a homomorphic encryption scheme that is both additive and affine. Moreover, Paillier's cryptosystem is also *semantically secure* (*cf.* Definition 3.3) which means that a computationally-bounded adversary cannot derive non-trivial information about a plain-text a given its encryption $\mathsf{Enc}(a)$ and the public key pk. For instance, a computationally-bounded adversary who is given two different cipher texts encrypted with the same key of a semantic cryptosystem, cannot even decide with non-negligible probability if the two cipher texts correspond to the encryption of the same plain text or not. In particular, semantically secure cryptosystem is by essence *probabilistic*, meaning that even if the same message is encrypted twice, the two resulting ciphertexts will be different except with negligible probability. In this chapter, we will also use a *threshold version* of Paillier's cryptosystem [53] (*cf.* Definition 3.4), which requires the active cooperation of the peers to decrypt.

Definition 3.3 (Semantic security [54])**.** An encryption scheme is said to be semantically secure if a computationally-bounded adversary cannot derive non-trivial information about the plain text from the cipher text and the public key only.

Definition 3.4 (Threshold cryptosystem)**.** A (t, n) threshold cryptosystem is a public key cryptosystem in which at least $t > 1$ peers out of n need to actively cooperate in order to decrypt an encrypted message. In particular, no collusion of $t - 1$ or

[1] In order to simplify the notion, we will drop the indices and write $\mathsf{Enc}(a)$ instead of $\mathsf{Enc}_{pk}(a)$ and $\mathsf{Dec}(a)$ instead of $\mathsf{Dec}_{sk}(a)$ for the rest of this chapter.

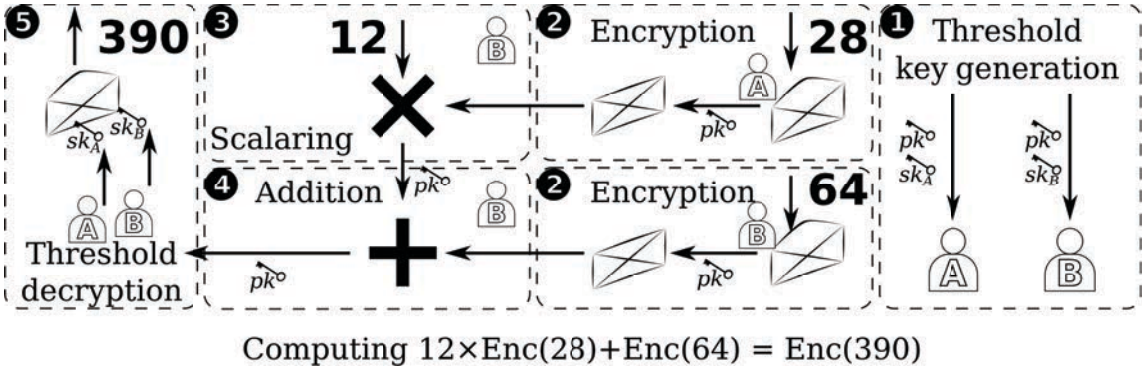


Figure 3.1 – Threshold homomorphic cryptosystem: Both peers generate the keys (step 1), encrypt their input values (step 2), and perform calculations on the encrypted values (step 3 and 4). To decrypt, in step 5 they need to use both their secret keys and engage in a decryption protocol.

less peers can decrypt a cipher text. However, any peer may encrypt a value on its own using the public key pk. After the threshold cryptosystem has been set up, each peer i gets his own secret key sk_i (for $i \in \{1, \ldots, n\}$), which is useless by itself but must be used as input to a threshold-decryption protocol by t peers or more to successfully decrypt a cipher-text.

In the previous definition, when we say that the peers cooperate to decrypt it means that they engage in an interactive protocol: the *decryption* protocol, which is part of the definition of the threshold cryptosystem. In such a protocol each peer's input is a *part* of the secret key, along with the cipher-text. After the peers exchange certain messages according to the protocol, the output is the plain text corresponding to the cipher text (*i.e.* effectively decrypting it).

Figure 3.1 gives an example of a threshold homomorphic cryptosystem in which peers perform computations on the cipher-text. To decrypt the result both peers must cooperate using their secret keys.

3.2.2 The bidirectional anonymous channel

The privacy budget issue (*cf.* Section 2.3.1) occurs when a peer needs to engage in more similarity computations than his privacy budget can afford. Let $\boldsymbol{d}$ be the peer's profile and $\mathcal{M}$ be a differentially private probabilistic function on $\boldsymbol{d}$ whose running time is bounded by some polynomial $g(n)$, in which n is the length of $\boldsymbol{d}$. Let $r_1 \neq r_2$ be two random binary strings of length $g(n)$ representing the random coins used by $\mathcal{M}$. Then $f_1(\boldsymbol{d}) = \mathcal{M}(\boldsymbol{d}, r_1)$ and $f_2(\boldsymbol{d}) = \mathcal{M}(\boldsymbol{d}, r_2)$ are deterministic functions. Suppose, for the sake of the argument, that if the adversary received either $f_1(\boldsymbol{d})$ or $f_2(\boldsymbol{d})$, but not both, then it would not be able to breach the peer's privacy, while if it held both f_1 and f_2 it will be able to breach his privacy. The peer could protect his privacy from being breached by either computing $\mathcal{M}$ only once in his lifetime, or alternatively, by ensuring that the adversary cannot receive both values. Ensuring that the adversary cannot get both values is very difficult

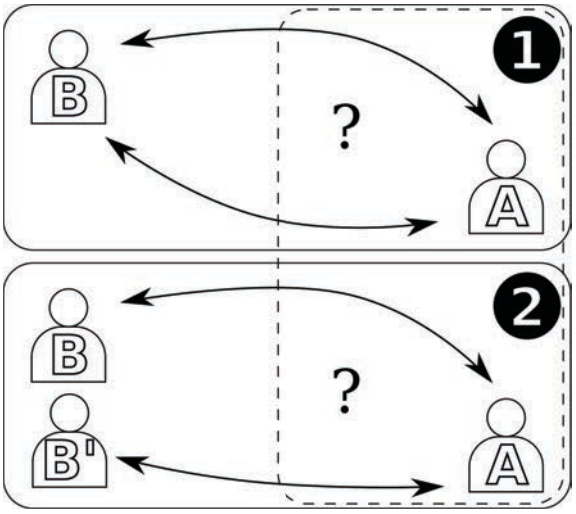


Figure 3.2 – Bidirectional anonymous channel: The peer A cannot distinguish scenario 1 from scenario 2. In particular, peer A is not able to tell whether peer B and peer B' are the same individual or two different ones.

in a P2P system due to vulnerabilities such as Sybil[2] attacks [55]. Instead, our objective is to prevent the adversary from linking both values to each others. In particular, even if the adversary was able to obtain both values, it would not be able to tell if they belong to the same peer, as illustrated in Figure 3.2. Requiring sender anonymity [56] alone is not sufficient because of the symmetric nature of P2P networks. Both peers carrying out a similarity computation need to have the same guarantee. Therefore, we describe the notion of *bidirectional anonymous channel* to fulfill this purpose.

The definition of *unlinkability* in [57] states that two items are unlinkable if the probability that they are related stays the same before (a priori knowledge) and after an execution of the system (a posteriori knowledge). In our definition, the two items that we require to be unlinkable are two similarity values computed over two different channels. An execution of the system, relative to which we measure the a priori and a posteriori knowledge, refers to the establishment of the channel, but does not take into account what is exchanged over that channel after it is established [57]. For instance, if a peer sends his identity over the channel, this would break the unlinkability. We consider that the attacker is one of the two correspondents of the channel and that it always has access to what is transmitted over it.

Definition 3.5 (Bidirectional anonymous channel)**.** Let A, B, and C be three different peers. Before establishing communication channels to B and C, A has a priori knowledge that peers B and C might be the same peer with probability p. The channels are called *bidirectional anonymous* if and only if (for all $A, B \neq A, C \neq A$) after A establishes those channels and before any information is exchanged, the a posteriori knowledge of A that B and C might be the same peer is still exactly

[2] A Sybil attack is an attack in which the adversary creates more than one identity to overcome identity-based restrictions.

p. Moreover, if $B = C$, then the a posteriori knowledge of B that he has two simultaneous channels to A is the same as his a priori knowledge. If $B \neq C$ and B was colluding with C, then their a posteriori knowledge that both their channels transmit to the same peer is the same as their a priori knowledge.

Gossip on behalf. In this section we describe two methods for implementing a bidirectional anonymous channel. The first method is more efficient but is suitable only against a passive adversary while the second method is more expensive but is more secure against a malicious adversary. We stress that we still do not consider a malicious adversary in this thesis but we present the second variant to show the feasibility of constructing a bidirectional anonymous channel.

A random peer sampling technique that is resilient to adversarial manipulation is used to make it less likely that the random choice of peers is biased by the adversary [33]. In particular, when we say in the following that a peer is chosen randomly, this means that the probability that the chosen peer belongs to the adversary is equal to the ratio of adversary-controlled peers to the total number of peers in the system.

The first variant, shown in Figure 3.3, a peer A starts by choosing at random another peer P in his neighborhood. Then, P selects one of his own acquaintances, denoted B, as the second peer with which A computes similarity. Peer P does not disclose the identity of B to A or vice-versa, effectively acting as an anonymizer. Then A and B initiate a key exchange protocol [58] with each other to make their communications private and unreadable by P. For example, A generates a pair of public key/secret keys for this channel and lets P transmit the public key to B. Afterwards, B uses the public key to encrypt a symmetric cipher's (such as AES) private key and transmits it back to A via P. A can decrypt the message and obtain the AES private key. A and B can now exchange encrypted messages through P, so P learns nothing about the information exchanged during the life time of this channel (besides the number of messages and their size). Peers A and B now have a private bidirectional anonymous channel, given that P does not collude with either one of them.

The second, more secure variant, is shown in Figure 3.4. For this variant, we need every peer in the network to have a public key of his own, that is spread with his address by the random sampling protocol. A peer A chooses a random peer A' at random. Next, A initiates a traditional anonymous channel (that provides anonymity only for A but not for A' [3]), such as [60], to A'. Using the public key of A', A performs key exchange with A' to make their channel private. Note that by using the public key of A', A validates that there is no man-in-the-middle between him and A'. Notice that the path between A and A' may be re-initiated using a different anonymous channel from time to time, as traditional anonymous channels require to avoid traffic analysis attacks. Now, A' is the proxy of A, valid only for one bidirectional anonymous channel. On the other hand, another peer B does the same steps and denotes B' as his proxy. When A' meets B' in his random view, A' sends A the public key of B', and B' sends B the public key of A'. Using this information, A and B engage in a key-exchange protocol with each other in the following way. A

[3]The hidden-services (randezvous points) of Tor [59] provides anonymity to one end and pseudonymity to the other end, but not anonymity to both ends.

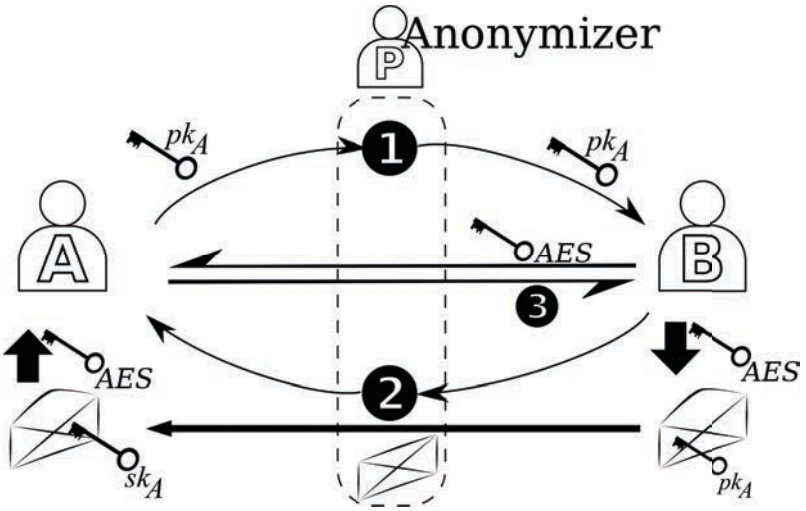


Figure 3.3 – Gossip-on-behalf: Peer P makes sure peer A and B does not know each others' identities. Step 1 and 2 is for securing the channel via exchanging an AES symmetric encryption key. Step 3, a bidirectional anonymous secure channel is established via peer P who cannot read the encrypted contents of the channel.

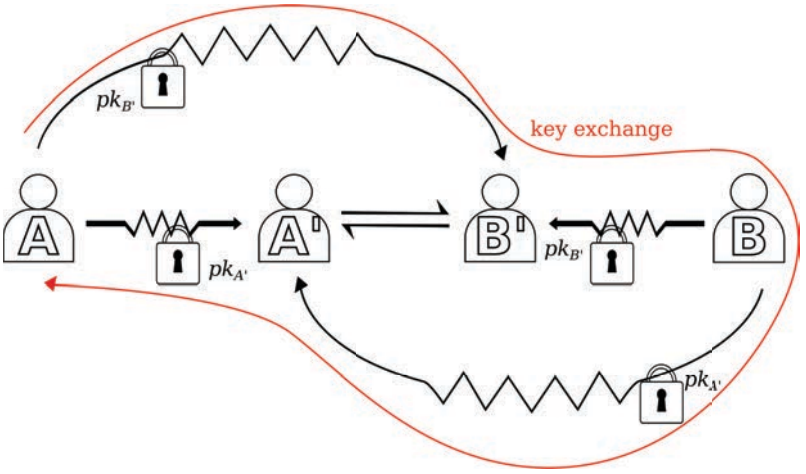


Figure 3.4 – Secure gossip-on-behalf. Peer A has an anonymous channel to A', and similarly for peer B. When A' and B' encounter each others A and B initiate a key-exchange protocol such that A' and B' cannot mount a man-in-the-middle attack unless they collude together.

initiates a one-way[4] anonymous channel to B' using the public key of B', and B does the same to A', then the key exchange protocol is performed by A sending a suitable message to B', which forwards it to B, which performs the appropriate computation and sends the result back to A via A'. This is done so that no man-in-the-middle attack may be mounted except if A' and B' collude together if A and B are honest. A and B now can communicate through A' and B' with bidirectional anonymity.

3.3 Related work

In this section, we describe work related to this chapter and how it compare to our contributions. In particular, we discuss anonymous channels and distributed noise generation. For the general system of providing both peer clustering according to interests in a P2P system, a protocol for nearest neighbor search in distributed settings has been proposed in [61] but it was designed only within the cryptography framework and not the differential privacy context. Differentially private protocols for collaborative platforms have also been considered in centralized systems such as [62] for analyzing the recommender system of Netflix, but the data are still entrusted to a central entity.

3.3.1 Anonymity

In the context of a communication channel, anonymity can be one of three types [56]: anonymity of the recipient, anonymity of the sender, and unlinkability of sender and receiver. Note that the latter (which is called *relationship anonymity*) is weaker than the first two [57] and can be achieved via a MIX-net [63], assuming that an adversary does not have full control on all the MIXes. In this scenario, even if an adversary wiretapped the similarity value, he cannot use this information directly unless he knows the sender and the receiver. However, the adversary may learn for a certain receiver its similarity distribution with respect to the network. This is not an issue in our protocol because an adversary who is not controlling either one of the two peers cannot eavesdrop because the channel between two peers is a private channel (implemented via cryptographic means).

The closest to bidirectional anonymous channels is Tor's hidden services [59], in which a client communicates with a server. The client is anonymous with respect to the web server. However, the web server is only pseudonymous with respect to the client. That is, two different clients know for sure that they are communication with the same server. This is not what a bidirectional anonymous channel provides. However, bidirectional anonymous channels solve a different use case than Tor's hidden services. For instance, a client using a bidirectional anonymous channel cannot choose which server he connects to. The AP3 system [64] provides decentralized anonymous communication for P2P systems. The provide secure pseudonyms, which does not serve our purpose of unlinkable channels. They also provide anonymous

[4] A one-way anonymous channel is a channel where messages pass only in one direction. One-way anonymous channels may be cheaper than two-way anonymous channels because they do not need to maintain state enabling the recipient to reply.

channels, providing sender anonymity only. The publish-subscrib framework they describe suffers from the same drawback as Tor's hidden services. Finally, *D*-Crowds [60] extends Crowds [65], which is also for P2P systems, but only provides sender anonymity.

The bidirectional anonymous channel provides both sender anonymity and recipient anonymity. Therefore, it may not be useful for client-server architectures because there is no way to initiate a connection to the desired server. However, it is a perfect match for P2P gossiping protocols in which there is no special role for any particular peer.

3.3.2 Distributed noise generation

Differential privacy requires randomness to operate, and this randomness must remain secret to the party receiving the differentially-private output. Otherwise, the noise could removed and the differential privacy guarantee would be violated. Therefore, since the two peers involved in the two-party computation are also the ones receiving its output, neither one of them should know the randomness used by the mechanism. More specifically, the task of generating the noise cannot be delegated to one of them; they have to engage in an interactive protocol to generate noise. Furthermore, the noise they generate collaboratively should be generated in an encrypted form.

The protocols used to generate random coins are known as coin-tossing protocols. It is known that in the two-party case, a malicious adversary can bias the coin with additive $\Theta(1/r)$, where r is the number of rounds of the coin-tossing protocol [66]. Since we assume a passive adversary in this thesis, we are able to generate unbiased coins, otherwise we could use the optimal two-party coin-tossing protocol from [66] to generate biased coins and then use the differential privacy mechanism from [67] which can provide differential privacy even from unbiased coins.

There are several coin-tossing protocols for the multi-party case (honest majority) [51, 25, 50, 68], which do not necessarily apply for the two-party case of this thesis. Thus we only mention the multi-party protocol of [25] (*ODO* framework) which is designed for generating noise specifically for differential privacy. ODO it is still not necessarily applicable to the two-party setting of this thesis. ODO framework can generate private Binomial and Poisson random variables, approximating Gaussian and Laplacian noise, respectively. However, their generated noise results in a relaxed variant of ε-differential privacy called (ε, δ)-differential privacy which is a strictly weaker definition [69]. To provide ε-differential privacy instead, query answers have to be bounded. In this chapter we describe a semi-honest two-party noise generation protocol that satisfies ε-differential privacy without bounding the query answers. Moreover, instead of generating noise jointly like [25], in our protocol, each party will generate noise locally and incorporate it with the other party's noise by addition, making the noise generation step more efficient, but is only secure against a passive adversary.

3.4 Threshold similarity

In this section, we introduce the concept of *threshold similarity*. The key idea of this concept is to avoid revealing the similarity value if it is below a given publicly known threshold. The motivation is two-fold. First, releasing only one bit of information (whether the similarity is below a threshold or not) is definitely better in terms of information lekage than releasing the value of the similarity itself. Second, it makes it more difficult for an adversary to guess the profile of a particular peer, for instance the situation in which the adversary keeps guessing items progressively to increase the similarity with a particular peer until he fully reconstructs his profile. In this case, threshold similarity makes it more difficult for the adversary because he gets no clues and thus its initial estimate of the profile must be considerably good.

In the rest of this section we discuss how to provide a secure implementation for threshold similarity computation that provides privacy in the secure multi-party computation model. The protocol is composed of several steps, including computing the inner product, squaring it, and then finally comparing whether the cosine similarity is greater than a predetermined threshold. In the following section we start by describing the first step: computing the inner product.

3.4.1 Computation of the inner product

There are several types of protocols that which can be used to compute the inner product between two binary vectors. The first type of protocols is the protocols directly computing the inner product while the second type is the *cardinality set intersection* protocols. A cardinality set intersection protocol takes two sets of integers as inputs and outputs the number of integers they have in common. This protocol can also be used to compute the inner product of two binary vectors because each binary vector $v \in \{0,1\}^n$ could be represented as a set of integers $\{i \mid v_i = 1, i \in \{1,\ldots,n\}\}$. The cardinality set intersection protocols could provide a performance advantage when n is big and the binary vectors are sparse (*i.e.* the number of bits set to 1 is small compared to the number of bits set to zero).

There are advantage and drawbacks for both set intersection based and inner product based approaches. To our knowledge, all existing secure set intersection protocols [70, 71, 72, 73] include steps proportional to the number of ones in the binary vectors, hence revealing this information. On the other hand, inner product protocols [74, 75, 76, 77] inherently require communication complexity proportional to n, which is independent of the number of ones in the binary vectors, although at the cost of an increased communication. On the computational size, the inner product approach has a linear computational complexity while the set intersection approach has a quadratic complexity. Since our primary interest is privacy, we selected the inner product approach since it hides the number of ones. First, we will give an overview of different inner product protocols in the literature in the following section as well as set intersection protocols.

Inner product approach. The protocols for inner product can be divided into two main branches, those based on secret-sharing [78, Section 4.3], and those based

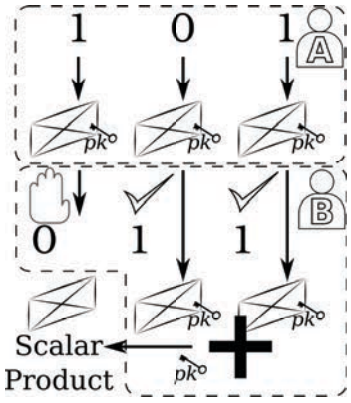


Figure 3.5 – inner product protocol: Peer B adds the encrypted bits of the peer A, which correspond to the ones of peer B to get the encrypted inner product.

on homomorphic encryption [74, 75, 76]. The one based on secret-sharing, although providing no inherent advantage on the message communication complexity, provides substantial advantage in the bit communication complexity because secret-sharing scheme provides perfect secrecy and thus does not required large cipher-texts.

Unfortunately, secure multiplication in secret-sharing needs at least 3 peers, thus requiring a semi-trusted[5] third party [78]. It would be possible, if we are using gossip-on-behalf for anonymization, to use the anonymizer as this semi-trusted third party.

The protocol in [76] reveals a permuted sum of the two vectors. This protocol is only secure if the other vector is uniformally distributed over the integers, which is definitely not secure for binary vectors as a peer could easily learn the number of zeros and number of ones in our case.

Both [74], [75] proposed the same protocol. We use this, to which we will refer as "InnerProduct", to implement our inner product step in Algorithm 1. In a preprocessing step to this protocol, the two peers engage in the setup phase of a key generation protocol for an threshold affine homomorphic cryptosystem [53]. At the end of this key generation phase, both peers have received the same public key pk that can be used to homomorphically encrypt a value and each one of the two peers has, as private input, a secret key, respectively sk_a for the first peer and sk_b for the second peer. The threshold cryptosystem[6] is such that any peer can encrypt a value using the public key pk but that the decryption of a homomorphically encrypted value require the active cooperation of the two peers. Figure 3.5 illustrates this protocol. For illustration purpose, we call the first peer Alice and the second peer Bob. This protocol is also illustrated in Algorithm 1.

Cardinality set intersection approach. We do not use cardinality set intersection in our work but we mention it for the sake of completeness.

Similarly to the inner product protocols, the protocols for computing the size of the set intersection are also divided into those who rely on secret-sharing schemes [70] and those based on homomorphic encryption [72, 73, 79]. The protocol in [70]

[5]Semi-trusted here refers to a passive adversary.

[6]The threshold cryptosystem should not be confused with the threshold similarity.

Algorithm 1 $\mathsf{InnerProduct}(\boldsymbol{d} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n), \boldsymbol{d}' = (\boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_n))$

for $i = 1$ to n **do**
  Alice computes $\mathsf{Enc}(\boldsymbol{d}_i)$
end for
Alice sends $\mathsf{Enc}(\boldsymbol{d}_1), \ldots, \mathsf{Enc}(\boldsymbol{d}_n)$ to Bob
Bob sets $s = \mathsf{Enc}(\boldsymbol{d}_1)$ if $\boldsymbol{d}'_1$ equals 1, and sets $s = \mathsf{Enc}(0)$ otherwise.
for $i = 2$ to n **do**
  if $\boldsymbol{d}'_i = 1$ **then**
    Bob sets $s = s \oplus \mathsf{Enc}(\boldsymbol{d}_i)$
  end if
end for
Bob sends s to Alice

is based secret-sharing and requires the help of a third party in the two-party case, therefore we focus instead on [72, 73, 79].

The protocols proposed in [73, Section 4.4] and [72, Section 5.2] rely on the same idea. More precisely, they both represent sets as polynomials, in which the roots of the polynomial are the set items represented as integer. The first protocol is a two-party one while the second protocol is a multi-party one. In both protocols, all items must be compared to each others, which leads to complexity that is quadratic. Inan, Kantarcioglu, Ghinita, and Bertino [79] relaxes this requirement by providing a differentially private protocol to *block* some unneeded comparisons. In the example they presented, they could save up to 17% of the comparisons needed compared to [73] and [72].

3.4.2 Threshold cosine similarity

We are interested in a form of similarity that outputs no more than one bit of information stating whether (or not) the similarity between two profiles is above some predetermined threshold τ.

Definition 3.6 (Threshold similarity)**.** Two peers are said to be τ-similar if the output of applying some similarity measure sim on their respective profiles $\boldsymbol{d}, \boldsymbol{d}'$ is above a chosen threshold $0 \leqslant \tau \leqslant 1$ (*i.e.* $\mathsf{sim}(\boldsymbol{d}, \boldsymbol{d}') > \tau$).

In practice, the value of the threshold τ depends on the application and is set empirically so as to be significantly above the average similarity in the population. Nonetheless, in Section 3.6 we provide heuristic for selecting the threshold as a function of the desired acceptance ratio.

The threshold similarity protocol takes as input two profiles $\boldsymbol{d}$ and $\boldsymbol{d}'$ (one profile per peer) represented as binary vectors and output one bit of information that is 1 if $\boldsymbol{d}$ and $\boldsymbol{d}'$ are τ-similar (*i.e.* $\mathsf{sim}(\boldsymbol{d}, \boldsymbol{d}') > \tau$ for sim a predefined similarity measure and τ the value of the threshold), and 0 otherwise. The threshold similarity is very appealing with respect to privacy as it guarantees that the output of the similarity computation will reveal no more than one bit of information, which is potentially much less than disclosing the exact value of the similarity measure. In order to

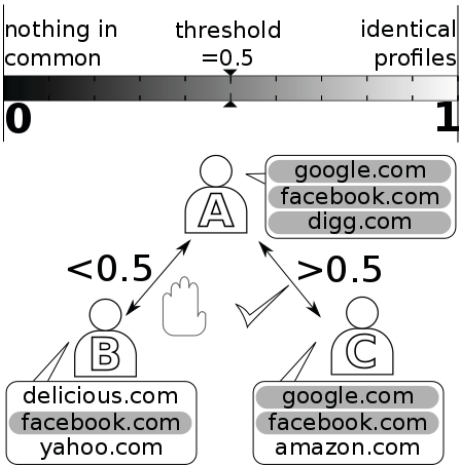


Figure 3.6 – Threshold similarity: The peer accepts only the peers having a similarity with him above a certain threshold. The similarity is based on the common items in their profiles, shaded in gray.

implement the threshold step, we employ a secure integer comparison protocol. In the next section, we review the relevant secure integer comparison literature.

3.4.3 Integer comparison step

The integer comparison problem is known as the socialist millionaires' problem in which two parties have their own private inputs and want to compare them without revealing anything about their inputs. A variety of solutions to this problem have been developed since Yao [80] proposed it.

Nonetheless, all these protocols can be gathered as two different families, both of which depend on bit-wise operations. One family of approaches is based on encrypted truth tables, while the other exploits homomorphic operations on encrypted bits.

The encrypted truth table technique requires the input to be known (*i.e.* in plaintext) to its owner as opposed to being passed as it is without requiring decryption after being evaluated from a previous circuit. The homomorphic operations on encrypted bits takes advantage of the homomorphic properties of a cryptosystem to achieve the same with less communication overhead than encrypted truth tables but at the expense of using asymmetric encryption.

Some homomorphic methods require both parties to know the value of their inputs to be able to correctly setup the protocol [81], while others can work directly on the bit-wise encryption of these inputs [48, 49]. In the situation in which the value as a whole is encrypted but the bit-wise encryption is not available, bit decomposition protocols are used.

Bit decomposition. Since all these methods depend on bit-wise operations, an important ingredient of these protocols is a bit decomposition scheme. A bit decomposition scheme takes a homomorphically encrypted integer and outputs the homomorphically encrypted values of each bit of this integer [82, 50]. Nishide and

Ohta [49] provide better solution for integer-comparison protocol by requiring only the least significant bit to be computed instead of the entire bit decomposition. This method worked originally for secret-sharing schemes but can be adapted to homomorphic encryption as well. Lin's method [81] requires at least one peer to know his number in plain-text while Garay's method [48] is less efficient than Lin's as it requires both homomorphic addition and multiplication to operate on the encrypted bits. We reject the former as both peers will not have their input in plain-text, and we reject the latter because Nishide's protocol (presented in the following paragraph) is more efficient than both Lin's protocol and Garay's protocol it as it uses only the least significant bit instead of all the bits.

To the best of our knowledge, Nishide's protocol [49] is potentially the most efficient protocol in terms of computation and communication. It operates only on the least significant bit as detailed. This method was originally presented in the context of secret-sharing, but we translate it to the setting of homomorphic encryption in Algorithm 2. In this algorithm, we use the notation $\bar{x}$ to denote the negation of a bit x (*i.e.* $\bar{x} = 1 - x$). We also use the notation $x \vee y$ to denote the OR gate, such that $x \vee y$ is 1 if and only if at least one of x or y is 1. The sub-protocol LeastSignificantBit is defined in [49] as well.

Algorithm 2 CompareIntegers($\mathsf{Enc}(a), \mathsf{Enc}(b)$)

a, b integers, k is the cryptosystem modulus [49]

1: $w \leftarrow \mathsf{LeastSignificantBit}(2\mathsf{Enc}(a) \mod k)$
2: $x \leftarrow \mathsf{LeastSignificantBit}(2\mathsf{Enc}(b) \mod k)$
3: $y \leftarrow \mathsf{LeastSignificantBit}(2(\mathsf{Enc}(a) - \mathsf{Enc}(b)) \mod k)$
4: **if** $\mathsf{Dec}(w\bar{x} \vee \bar{w}\bar{x}\bar{y} \vee wx\bar{y}) = 1$ **then**
5: Output "$a < b$"
6: **else**
7: Output "$a \geqslant b$"
8: **end if**

Checking the least significant bit of $2a \bmod k$, is equivalent to checking whether $a < k/2$ or not, in which k is the RSA modulus of the homomorphic encryption (or the secret-sharing scheme field cardinality).

Nishide and Ohta have showed using a truth-table that using w, x, y it is possible to uniquely determine each corresponding output for $a < b$. This method is efficient because it only uses affine multiplication, and a very shallow binary circuit with only 3 least significant bit protocol invocations (which is the just first round of any bit decomposition protocol).

Unfortunately, since affine homomorphic cryptosystems cannot provide multiplication of encrypted values, they cannot be used to compute the last binary circuit (the "if statement"). To handle this step, a method such as the conditional gate from [83] has to be used[7].

[7]The conditional gate provides multiplication for binary values, and it should not be confused with the multiplication gate used in the following section to multiply integers.

The conditional gate is an interactive protocol to multiply two bits $x, y \in \{-1, 1\}$[8], which involves each peer in turn multiplying the same random number $s_i \in \{-1, 1\}$ ($i = 1$ for the first peer and 2 for the second peer) of his choice to both bits, ending up with $\mathsf{Enc}(s_1 s_2 x)$ and $\mathsf{Enc}(s_1 s_2 y)$. Finally they perform a threshold decryption of the former before scalarizing it with the other to obtain $s_1 s_2 x \odot \mathsf{Enc}(s_1 s_2 y)$ which yields $\mathsf{Enc}\left(xy \cancel{(s_1 s_2)^{2-1}}\right) = \mathsf{Enc}(xy)$ as desired. The conditional gate along with the negation gate[9] compose a NAND gate, which is universal for binary circuits, and is used to evaluate the Boolean logic expression in the protocol.

3.4.4 Private similarity computation

The cosine similarity as denoted in (2.1) requires the extraction of square root, which is a non-trivial operation to implement as it requires costly homomorphic encryption operations. Instead, we compute the squared cosine similarity. To do this, the numerator, namely the inner product, needs to be squared and this is achieved using the multiplication gate from [51] to multiply it by itself. The denominator can be easily computed. The first peer sends a homomorphic encryption of the number of ones in his binary vector to the second peer (*i.e.* $\mathsf{Enc}(\|\boldsymbol{d}\|_1)$), who will scalarize it by his own set cardinality by doing $\|\boldsymbol{d}'\|_1 \odot \mathsf{Enc}(\|\boldsymbol{d}\|_1)$ to obtain $\mathsf{Enc}(\|\boldsymbol{d}\|_1 \times \|\boldsymbol{d}'\|_1)$.

Recall, that the objective of the protocol is only to learn if the similarity between $\boldsymbol{d}$ and $\boldsymbol{d}'$ is above a certain (publicly known) threshold τ. We assume that the threshold is in $\mathbb{Q}$ and therefore can be represented as a fraction $\tau = \tau_1/\tau_2$ (for τ_1 and τ_2 positive integers such that $\tau_1 \leqslant \tau_2$). Our objective is to verify whether or not the following condition holds

$$\frac{(\boldsymbol{d} \cdot \boldsymbol{d}')^2}{\|\boldsymbol{d}\|_1 \times \|\boldsymbol{d}'\|_1} > \frac{\tau_1}{\tau_2} \iff \tau_2 (\boldsymbol{d} \cdot \boldsymbol{d}')^2 > \tau_1 \|\boldsymbol{d}\|_1 \times \|\boldsymbol{d}'\|_1 \tag{3.1}$$

The left side and right side of the inequality can be compared by secure protocols for integer comparison (*cf.* Section 3.4.3). We choose to apply specifically the comparison technique from [49] as it neither require knowledge of the input[10], nor a full bit decomposition of the input as other protocols. Although this protocol was developed initially for secret-sharing, it can be implemented with homomorphic encryption as well. The output of this comparison step is one bit stating whether or not the squared cosine similarity is above the threshold τ. This protocol is called ThresholdCosine and is detailed in Algorithm 3. For illustration purpose, we call the first peer Alice and the second peer Bob.

Theorem 3.1 (Protocol for threshold cosine similarity)**.** *The protocol ThresholdCosine is private with respect to a passive adversary (in the secure multi-party computation sense) and returns* 1 *if two peers are τ-similar and* 0 *otherwise. The protocol has a communication complexity of $O(n)$ bits and a computational cost of $O(n)$.*

[8] In case the encrypted bits was from the domain $\{0, 1\}$, it is straight-forward to convert them to $\{-1, 1\}$ using the affine homomorphic encryption properties.

[9] The negation gate of an encrypted bit $\mathsf{Enc}(x)$ is simply $1 \oplus (-1 \odot \mathsf{Enc}(x)) = \mathsf{Enc}(1 - x)$.

[10] Therefore, the input can be the encrypted output of a preliminary cryptographic protocol.

Algorithm 3 $\mathsf{ThresholdCosine}(\boldsymbol{d},\boldsymbol{d}',\tau = \tau_1/\tau_2)$

1: Alice and Bob generates the keys of the threshold homomorphic encryption
2: Alice receives sk_a, Bob receives sk_b and they both get the public key pk
3: Alice and Bob compute $\mathsf{Enc}\left(\boldsymbol{d} \cdot \boldsymbol{d}'\right)$ using the protocol $\mathsf{InnerProduct}$ (Algorithm 1)
4: Alice applies the multiplication gate from [51] to obtain $\mathsf{Enc}\left(\left(\boldsymbol{d} \cdot \boldsymbol{d}'\right)^2\right)$
5: Alice computes $\mathsf{Enc}\left(\left(\boldsymbol{d} \cdot \boldsymbol{d}'\right)^2\right) \odot \tau_2 = \mathsf{Enc}\left(\tau_2 \left(\boldsymbol{d} \cdot \boldsymbol{d}'\right)^2\right)$
6: Alice computes $\mathsf{Enc}\left(\|\boldsymbol{d}\|_1\right)$ and sends it to Bob
7: Bob computes $\mathsf{Enc}\left(\|\boldsymbol{d}\|_1\right) \odot \left(\tau_1 \|\boldsymbol{d}'\|_1\right) = \mathsf{Enc}\left(\tau_1 \|\boldsymbol{d}\|_1 \times \|\boldsymbol{d}'\|_1\right)$
8: Alice and Bob use the integer comparison protocol $\mathsf{CompareIntegers}$ (Algorithm 2) of [49] on $\mathsf{Enc}\left(\tau_2 \left(\boldsymbol{d} \cdot \boldsymbol{d}'\right)^2\right)$ and $\mathsf{Enc}\left(\tau_1 \|\boldsymbol{d}\|_1 \times \|\boldsymbol{d}'\|_1\right)$
9: **if** $\mathsf{Enc}\left(\tau_2 \left(\boldsymbol{d} \cdot \boldsymbol{d}'\right)^2\right) > \mathsf{Enc}\left(\tau_1 \|\boldsymbol{d}\|_1 \times \|\boldsymbol{d}'\|_1\right)$ **then**
10: output 1 to state that Alice and Bob are τ-similar
11: **else**
12: output 0
13: **end if**

Proof. All the communication exchanged between Alice and Bob is done using a homomorphic encryption scheme with semantic security (*cf.* Definition 3.3), therefore the encrypted messages exchanged do not leak any information about their content. Moreover as the encryption scheme is a threshold version, it means that neither Alice nor Bob alone can decrypt the messages and learn their content. The multiplication gate [51] as well as the integer comparison protocol [49] are also semantically secure, which therefore guarantees that the protocol is secure against a passive adversary.

Regarding the correctness, it can be seen from the execution of the protocol that if Alice and Bob are τ-similar then this will result in $\tau_2 \left(\boldsymbol{d} \cdot \boldsymbol{d}'\right)^2 > \tau_1 \|\boldsymbol{d}\|_1 \times \|\boldsymbol{d}'\|_1$ when the integer comparison protocol is executed (and therefore an output of 1) and in 0 otherwise. The multiplication gate and the integer comparison protocols are independent of n and can be considered as having constant complexity (both in terms of communication and computation) for the purpose of analysis. On the other hand, the protocol $\mathsf{InnerProduct}$ requires the exchange of $O(n)$ bits between Alice and Bob as well as $O(n)$ computations which result in a similar complexity for the global protocol $\mathsf{ThresholdCosine}$. □

3.5 Differentially private similarity computation

Cryptography gives us the tools to compute a distributed function in such a way that the computations themselves reveal nothing that cannot be learned directly from the output of the function. This is a strong privacy guarantee but at the same time it does not preclude the possibility that the output itself might leak information about the private data of individuals. Differential privacy provides guarantees on what the output itself may reveal. In order to obtain the best of both worlds (secure multi-party computation and differential privacy), the main idea is to use cryptographic techniques to securely compute a differentially private mechanism. In this section,

we give efficient and secure algorithms for computing a differentially private version of threshold similarity presented in the previous section.

For this chapter we use the notion of *bounded* differential privacy [84], in which two sets are considered to be neighbors if one of them could be obtained from the other by *replacing* one item, leaving the set cardinality unmodified (as opposed to the *unbounded* differential privacy in which one of the two sets could be obtained from the other by *adding or removing* one item). The adoption of this notion simplifies the computation of the global sensitivity of the cosine similarity. In our context, in which we treat binary vectors as a representation of sets, bounded differential privacy amounts to declaring two binary vectors to be neighbors if their Hamming distance is exactly 0 or 2 (both vectors have the same number of ones).

Global sensitivity of cosine similarity. The following lemmas state the sensitivity of inner product (equivalent to the sensitivity of cardinality set intersection) and the squared cosine similarity.

Lemma 3.1 (Sensitivity of the inner product)**.** *The global sensitivity of the function inner product function $a, b \mapsto \sum_i a_i b_i$ for binary vectors is 1.*

Proof. Let a' be a neighbor to a. Let $j \neq k$ be the only two positions at which a' differ from a. This implies that $a_i \neq a_j$ and $a'_i \neq a'_j$. Without loss of generality let $a_i = 0, a_j = 1$ and $a'_i = 1, a'_j = 0$. Then, for some binary vector b we have

$$|(a' \cdot b) - (a \cdot b)| = |(a' - a) \cdot b| = |(a'_i - a_i)b_i + (a'_j - a_j)b_j| = |b_i - b_j| \leqslant 1 \ . \quad (3.2)$$

□

It follows from the previous lemma that the sensitivity of cosine similarity, for bounded differential privacy, is $1/\left\|\boldsymbol{d}\right\|_1 \left\|\boldsymbol{d'}\right\|_1$. However for the squared cosine similarity, the situation is described by the following lemma.

Lemma 3.2 (Sensitivity of the squared inner product)**.** *The global sensitivity of the squared inner product function $a, b \mapsto \left(\sum_i a_i b_i\right)^2$ for binary vectors assuming a and b are not the vector of all zeroes, is*

$$2\min(\|a\|, \|b\|) - 1 \ . \quad (3.3)$$

Proof. Let the vector a' be a neighbor of the vector a, which means that, they differ on exactly two positions. Then for any binary vector b we have

$$\left|(a \cdot b)^2 - (a' \cdot b)^2\right| = \left|(a \cdot b)^2 - (a \cdot b + z)^2\right| \ , \quad (3.4)$$

for some $z \in \{-1, 0, 1\}$ by Lemma 3.1, then

$$\left|(a \cdot b)^2 - (a \cdot b + z)^2\right| = \left|(a \cdot b)^2 - (a \cdot b)^2 - 2z(a \cdot b) - z^2\right| = \left|-2z(a \cdot b) - z^2\right| \ . \quad (3.5)$$

Choosing $z = -1$ maximizes the expression $|(a \cdot b)^2 - (a' \cdot b)^2|$:

$$\left|(a \cdot b)^2 - (a' \cdot b)^2\right| \leqslant |2(a \cdot b) - 1| \quad . \tag{3.6}$$

In case $a \cdot b \geqslant 1$, it follows that

$$|2(a \cdot b) - 1| = 2(a \cdot b) - 1 \leqslant 2\min(\|a\|, \|b\|) - 1 \quad , \tag{3.7}$$

and for $a \cdot b = 0$, we use the assumption that both a and b have at least one non-zero entry to show that

$$|2(a \cdot b) - 1| = |-1| = 1 = 2\min(1, 1) - 1 \leqslant 2\min(\|a\|, \|b\|) - 1 \quad . \tag{3.8}$$

□

Corollary 3.1. *The sensitivity of the squared cosine similarity between two binary vectors a, b for bounded differential privacy is*

$$\frac{2\min(\|a\|_1, \|b\|_1) - 1}{\|a\|_1 \|b\|_1} \quad . \tag{3.9}$$

Proof. Note that in the bounded differential privacy model we treat $\|a\|$ and $\|b\|$ as constants, hence $|\mathsf{cos\_sim}(a, b)^2 - \mathsf{cos\_sim}(a', b)^2|$ for a' a vector that is a neighbor of a (therefore $\|a'\| = \|a\|$) is simply

$$\left|\frac{(a \cdot b)^2}{\|a\|\|b\|} - \frac{(a' \cdot b)^2}{\|a'\|\|b\|}\right| = |(a \cdot b)^2 - (a' \cdot b)^2| / \|a\|\|b\| \quad , \tag{3.10}$$

and the rest follows from Lemma 3.2. □

According to Theorem 2.1, to provide differential privacy it is sufficient to add Laplacian noise proportional to the sensitivity of the function to the true answer before releasing it.

This can be done interactively in a centralized environment in which a curator is holding the data and replying to queries. For the distributed setting, we discuss three possible alternatives for replacing the curator in the following sections. The first one depends on the availability of a semi-trusted third party while the last two are fully distributed.

We denote the noise that is to be added as the random variable

$$N = \mathsf{Lap}\left(0, \frac{2\min(\|\boldsymbol{d}\|_1, \|\boldsymbol{d}'\|_1) - 1}{\varepsilon \|\boldsymbol{d}\|_1 \|\boldsymbol{d}'\|_1}\right) \quad . \tag{3.11}$$

Specific values drawn from this distributed will be denoted by a lower case n_i, for some $i \in \mathbb{N}$. It shall not be confused with the total number of items in the system, n, because it will be subscripted with some index.

3.5.1 Using a semi-trusted third party

In the context of gossip-on-behalf, the peer that acts as an anonymizer in the bidirectional anonymous channel could also generate the required random noise. Note that while the anonymizer is assumed not to break anonymity, it should not observe the true value for the similarity. For instance, the anonymizer can add the random noise using the homomorphic property of the cryptosystem (as it may know the public key) to the similarity value that has been computed. Afterwards, the two peers that have been involved in the similarity computation can recover the result using threshold decryption. Algorithm 4 describes this procedure.

The noise has ultimately a finite representation on a digital computer and hence can be represented as a rational number. Rationals can be encoded using the standard Paillier cryptosystem [85] (while preserving the homomorphic addition and scalaring operations) given that the numerator is in $[-R, R]$, for some R, and the denominator is in $(0, S]$ for some S, such that $2RS < k$, in which k is the RSA modulus used by the Paillier cryptosystem. However, having finite precision (being rational) and a bounded range (the bound on R and S) may affect the privacy guarantees unless certain precautions are taken [86, 87]. These precautions are related to the noise generation, which may be performed locally by the anonymizer in plain-text, and to rounding the result after adding the noise, which could be using a constant number of rounds of a bit decomposition protocol [82, 50] by computing the low-order bits and subtracting them [86].

Algorithm 4 $\mathsf{DifferentialSquaredCosine}(\boldsymbol{d}, \boldsymbol{d}', \varepsilon)$

1: Alice and Bob generates the keys of the threshold homomorphic encryption
2: Alice receives sk_a, Bob receives sk_b and they both get the public key pk
3: Alice and Bob compute $\mathsf{Enc}\left((\boldsymbol{d} \cdot \boldsymbol{d}')^2\right) = \mathsf{InnerProduct}(\boldsymbol{d}, \boldsymbol{d}')^2$
4: Alice and Bob compute $\mathsf{Enc}\left(\|\boldsymbol{d}\|_1 \|\boldsymbol{d}'\|_1\right)$
5: Alice and Bob compute $\mathsf{Enc}\left(\left(\|\boldsymbol{d}\|_1 \|\boldsymbol{d}'\|_1\right)^{-1}\right)$ using the inversion gate from [88]
6: Alice, Bob and the anonymizer compute $\mathsf{Enc}\left((\boldsymbol{d} \cdot \boldsymbol{d}')^2 \left(\|\boldsymbol{d}\|_1 \|\boldsymbol{d}'\|_1\right)^{-1}\right)$ using the multiplication gate from [51]
7: The anonymizer adds Laplacian noise n_1 where $n_1 \sim N$ to obtain $\mathsf{Enc}\left((\boldsymbol{d} \cdot \boldsymbol{d}')^2 \left(\|\boldsymbol{d}\|_1 \|\boldsymbol{d}'\|_1\right)^{-1} + n_1\right)$
8: The anonymizer sends the perturbed squared cosine similarity (which is homomorphically encrypted) to Alice and Bob
9: Alice and Bob cooperate to decrypt the homomorphically encrypted value and use [85] to decode the value as a rational number

Theorem 3.2 (Protocol for differential squared cosine)**.** *Algorithm 4 is ε-differentially private and is secure with respect to a passive adversary. The protocol has communication cost of $O(n)$ bits and a computational complexity of $O(n)$.*

Proof. It follows from Theorem 3.1 that the first part of the protocol before the inversion step, is secure with respect to a passive adversary. The inversion gate needs the parties to jointly generate a random integer, which can be done efficiently with semantic security [51], so is the multiplication gate, both in constant number of rounds

and with communication complexity $O(n)$. The anonymizer has only the knowledge of the public key and thus cannot decrypt the messages he sees. Moreover, the messages are semantically secure and therefore leak no information to the anonymizer. At the end of the protocol, assuming that the anonymizer does not collude with Alice or Bob, Alice and Bob only get to learn $\mathsf{cos\_sim}(\boldsymbol{d}, \boldsymbol{d}')^2 + \mathsf{Lap}\,(0, S(\mathsf{cos\_sim}^2)/\varepsilon)$, which ensures the ε-differential property of the protocol according to Theorem 2.1. In terms of complexity, due to the use of the protocol InnerProduct as a subroutine, the protocol DifferentialSquaredCosine has a communication cost of $O(n)$ bits as well as a computational cost of $O(n)$, for n the number of items in the system (we consider here that the threshold decryption has constant complexity and is negligible with respect to the cost of the inner product). □

3.5.2 Distributed noise generation

Instead of depending on a third party to generate the noise, this section address two possibilities of distributed noise generation.

Difference of two exponentials. This method used the observation that the difference of two exponential random variable is a Laplacian random variable[11]. Alice and Bob generate (each one on his own) two exponential random variables n_1 and n_2 with parameter $\lambda = \varepsilon/S(\mathsf{cos\_sim}^2)$. Afterwards, during the protocol, Alice adds n_1 to the encrypted squared cosine similarity while Bob subtracts n_2.

If the similarity value is revealed, an honest-but-curious peer may remove his exponential noise from the revealed value. For example, if the output value was $x = s+n_1-n_2$, where s is the true similarity, the peer who generated n_2 could remove it from x once he received it. The value he ends up with is $x + n_2 = s + n_1$ satisfies a weaker variant of differential privacy called (ε, δ)-differential privacy [25], as a consequence this method is only recommended for threshold similarity computation in which the perturbed similarity value is not revealed.

Two Laplacians. This method is based on a secure protocol for computing the inner product. The sensitivity of the inner product function is 1 hence only $\mathsf{Lap}\,(0, 1/\varepsilon)$ noise is needed. Suppose that Alice and Bob want to release the result of the inner product between their two profiles. At the end of the protocol Alice and Bob could both simply add independently generated random noise with distribution $\mathsf{Lap}\left(0, \frac{1}{\varepsilon}\right)$ using the homomorphic property of the encryption scheme. Afterwards, they could cooperate to perform the threshold decryption and they would both get to learn the perturbed inner product. Then, Alice can subtract her own noise from the released output to recover a version of the inner product which have been perturbed only with Bob's noise (which she cannot remove). We provide in Algorithm 5 a protocol for inner product that satisfies ε-differentially privacy using the two Laplacians technique.

Theorem 3.3 (Protocol for differential inner product)**.** *The protocol DifferentialInnerProduct (Algorithm 5) is secure with respect to a passive adversary and is*

[11] Follows by multiplying the characteristic function $f(t)$ of the exponential distribution by $f(-t)$.

Algorithm 5 DifferentialInnerProduct($\boldsymbol{d}$,$\boldsymbol{d}'$,ε)

1: Alice and Bob generates the keys of the threshold homomorphic encryption
2: Alice receives sk_a, Bob receives sk_b and they both get the public key pk
3: Alice and Bob compute $\mathsf{Enc}\left((\boldsymbol{d}\cdot\boldsymbol{d}')^2\right) = \mathsf{InnerProduct}(\boldsymbol{d},\boldsymbol{d}')$
4: Alice generates Laplacian noise n_A parametrized by $\frac{1}{\varepsilon}$ and computes $\mathsf{Enc}\left((\boldsymbol{d}\cdot\boldsymbol{d}')\right)\oplus n_A = \mathsf{Enc}\left((\boldsymbol{d}\cdot\boldsymbol{d}')+n_A\right)$ and sends the result to Bob
5: Bob generates Laplacian noise n_B parametrized by $\frac{1}{\varepsilon}$ and computes $\mathsf{Enc}\left((\boldsymbol{d}\cdot\boldsymbol{d}')+n_A\right)\oplus n_B = \mathsf{Enc}\left((\boldsymbol{d}\cdot\boldsymbol{d}')+n_A+n_B\right)$
6: Alice and Bob cooperate to decrypt the homomorphically encrypted value and get as output $((\boldsymbol{d}\cdot\boldsymbol{d}')+n_A+n_B)$

ε-differentially private. The protocol has a communication cost of $O(n)$ bits and a computational complexity of $O(n)$, in which n is the total number of items in the system.

Proof. From an argument similar to one used in Theorem 3.2, the proof follows for the security against a passive adversary, the ε-differential privacy, and the complexity. Security against a passive adversary holds because all the messages exchanges during the protocol are semantically secure. Moreover, the output is ε-differentially private because it instantiates the Laplacian mechanism. Because of the use of the protocol InnerProduct as a subroutine, the protocol DifferentialInnerProduct has a communication cost of $O(n)$ bits as well as a computational cost of $O(n)$ (we consider here that the threshold decryption has constant complexity and is dominated with respect to the cost of the inner product). □

3.6 Utility analysis

When doing threshold similarity, adding noise will result in false positive and false negatives when it comes to deciding whether similarity is above a certain threshold. More precisely, a false negative arises when the protocol outputs that two peers are not τ-similar when in fact they are, and outputs that they are τ-similar when in fact they are not for false positive. False negatives may reduce utility while false positive may reduce privacy if it results in releasing the true similarity value afterwards. Nonetheless, as our experiments show in Figure 3.11, privacy is twice better than the baseline case where no threshold step takes place, even in presence of false positives.

We measure utility in terms of the false negative rate. In particular, the lower the false negative rate the better the utility. In the following section, we describe a theoretical model to estimate both the false negative rate and the false positive rate. For a fixed ε, when the threshold increases, both false positive rate and false negative rate decrease, which is better for *both* privacy and utility. For example, if $\tau = 1$, then all comparisons will be true negatives, and false negatives and false positives will be identically zero. Thus, the false negative rate cannot be the only utility measure, since in this case the false negative rate indicates optimum utility but the peers still fail to get clustered according to their interests. Thus, the recall measure

experimentally evaluated in Section 3.7 should also be considered in conjunction with the false negatives.

3.6.1 Model

We describe here the statistical model that we use for our analysis. This model can also be used as for choosing the threshold τ as well as the privacy parameter ε (*cf.* Section 3.6.4).

Let $x = \|\boldsymbol{d}\|_1$ and $y = \|\boldsymbol{d}'\|_1$ be the number of items in the profiles of the two peers computing their threshold similarity. Denote N, the additive noise, as a Laplace random variable with mean 0 and the scale parameter used in Section 3.5. For a random variable X, we denote by $F_X(x) = \Pr[X < x]$ and $f_x(x) = \Pr[X = x]$. We model the inner product between $\boldsymbol{d}$ and $\boldsymbol{d}'$ with the random variable S. In Lemma 3.3, we prove that if $\boldsymbol{d}$ and $\boldsymbol{d}'$ follow the uniform distribution on $\{0,1\}^n$, then S follows a Hypergeometric distribution (*cf.* Definition 3.7). The similarly function is the squared cosine similarly and is equal to the random variable S^2/xy.

Definition 3.7 (Hypergeometric distribution [89])**.** A hypergeometric distribution $\mathsf{Hypergeometric}(n, m, N)$ is a model of a box that contains N balls, among which there are m white balls. Afterwards, n balls are drawn at random from that box without replacement. We count a success when a white ball is drawn. A Hypergeometric random variable corresponds to the numbers of successes in such a run. The maximum number of successes is m.

Lemma 3.3 (Inner product follows hypergeometric distribution)**.** *Assuming that* $\boldsymbol{d}$ *and* $\boldsymbol{d}'$ *are drawn from a uniform distribution over* $\{0,1\}^n$*, then the random variable* S *representing their inner product follows*

$$\mathsf{Hypergeometric}(\max(x,y), \min(x,y), n) \ ,$$

in which $x = \|\boldsymbol{d}\|_1$ *and* $y = \|\boldsymbol{d}'\|_1$*.*

Proof. In this proof, we treat $\boldsymbol{d}$ as the set $\{i \mid \boldsymbol{d}_i = 1\}$ and similarly for $\boldsymbol{d}'$. Without loss of generality, assume that $\boldsymbol{d}$ is the smaller set of the two. Assume that $\boldsymbol{d}'$ is a collection of y items picked randomly from $\{1, \ldots, n\}$ without replacement. A pick, or alternatively a member of $\boldsymbol{d}'$, is called successful if it belongs to $\boldsymbol{d}$. Therefore, the number of possible successes is at most $m = x$. A successful pick adds 1 to S, the inner product of $\boldsymbol{d}$ and $\boldsymbol{d}'$, because it belongs to both. Therefore, S is the number of successes and follows its distribution, which is hypergeometric. □

3.6.2 False negative rate

Theorem 3.4 (False negative rate)**.** *The false negative rate is:*

$$\mathcal{U}_-(x, y, n, \tau, \varepsilon) = \sum_{s=\lfloor\sqrt{xy\tau}\rfloor+1}^{\min(x,y)} \frac{f_S(s) F_N(\tau - \frac{s^2}{xy})}{1 - F_S\left(\lfloor\sqrt{xy\tau}\rfloor\right)} \ . \tag{3.12}$$

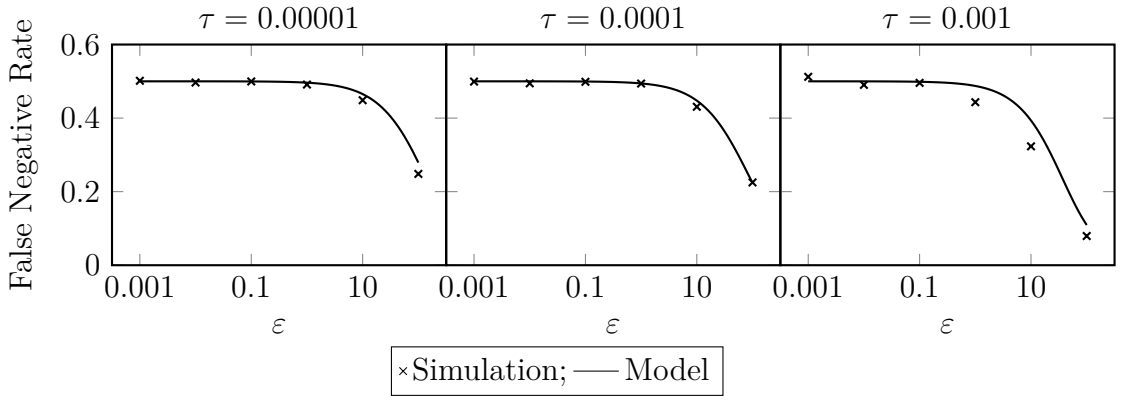


Figure 3.7 – The false negative probability model versus simulation results. Marks refer to the simulation results while the solid lines correspond to our model. Discrepancies may be due to the assumption that items are distributed uniformly. When the threshold increases, the false negative rate decreases (ε kept constant). The explanation being that when τ increases, the *true* positives decrease faster than the false negatives increase, and having the false negative rate = false negatives /(false negatives + true positives) explains the rest, because the numerator increases, while the denominator decreases.

Proof. The probability that a peer gets accepted is $\Pr[S^2/xy > \tau]$, in which τ is the public threshold value. The probability of being rejected after adding the Laplacian noise is $\Pr[S^2/xy + N \leqslant \tau] = \Pr[N \leqslant \tau - S^2/xy]$. Let $\theta = \sqrt{xy\tau}$ and $\gamma = \tau - \frac{s^2}{xy}$, then

$$\Pr\left[\text{rejected after adding the noise}|\text{accepted before adding the noise}\right] =$$

$$\begin{aligned}
\Pr\left[N \leqslant \gamma \middle| \frac{S^2}{xy} > \tau\right] &= \Pr\left[N \leqslant \gamma | S > \theta\right] \\
= \frac{\Pr\left[N \leqslant \gamma \wedge S > \theta\right]}{1 - F_s\left(\theta\right)} &= \sum_{s>\theta} \frac{\int_{-\infty}^{\gamma} f_{N,S}(n,s)\, dn}{1 - F_s\left(\theta\right)} \\
= \sum_{s>\theta} \frac{\int_{-\infty}^{\gamma} f_N(n) f_S(s)\, dn}{1 - F_S\left(\theta\right)} &= \sum_{s>\theta} \frac{f_S(s) \int_{-\infty}^{\gamma} f_N(n)\, dn}{1 - F_S\left(\theta\right)} \\
= \sum_{s>\theta} \frac{f_S(s) F_N(\gamma)}{1 - F_S\left(\theta\right)} &\ .
\end{aligned}$$

This summation runs from $\lfloor\theta\rfloor + 1$ to $\min(x, y)$. □

We validate the model with our experiments that are detailed in Section 3.7, whose results are shown in Figure 3.7.

3.6.3 False positive rate

In this section, we analyze the false positive rate as a measure of privacy. Indeed, a false positive may cause an exchange of the similarity even if the similarity is less than the threshold, which has a negative impact on privacy.

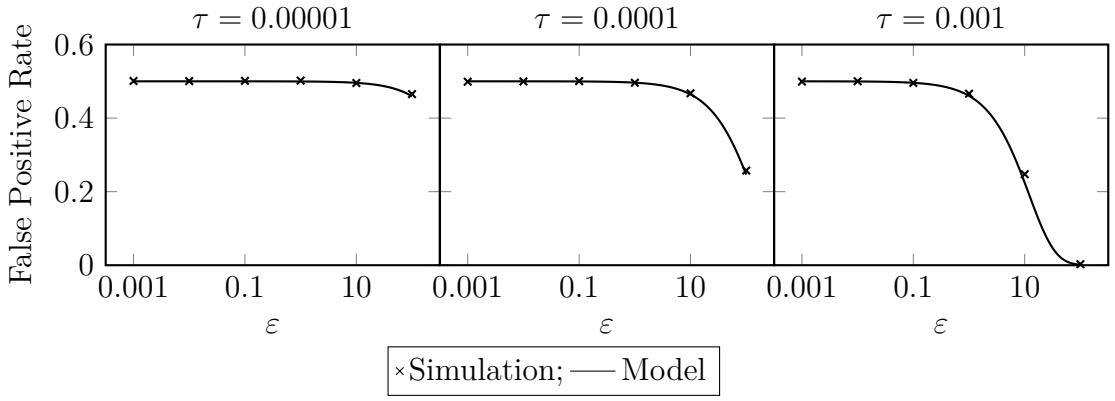


Figure 3.8 – The false positive probability model versus simulation results. Marks refer to the simulation results while the solid lines correspond to our model. The plot confirms the intuition that a higher threshold is better for privacy.

By a construction similar to that of Theorem 3.12, we find that the false positive rate is equal to

$$\begin{aligned}\mathcal{U}_+(x,y,n,\tau,\varepsilon) &= \Pr\left[N > \tau - \frac{S^2}{xy}\middle|\frac{S^2}{xy} \leqslant \tau\right] \\ &= \sum_{s=0}^{\lfloor\sqrt{xy\tau}\rfloor} \frac{f_S(s)(1-F_N(\tau-\frac{s^2}{xy}))}{F_S\left(\lfloor\sqrt{xy\tau}\rfloor\right)} \ ,\end{aligned}$$

in which x and y are the number of items in the two profiles computing similarity, N is the random variable representing the Laplacian noise added, τ is the threshold value, and S is the random variable representing the number of items in common between the two profiles.

Since N is a Laplacian random variable symmetric around 0, $1 - F_N(n) = F_N(-n)$, hence the false positive rate is equal to

$$\mathcal{U}_+(x,y,n,\tau,\varepsilon) = \sum_{s=0}^{\lfloor\sqrt{xy\tau}\rfloor} \frac{f_S(s)F_N(\frac{s^2}{xy}-\tau)}{F_S\left(\lfloor\sqrt{xy\tau}\rfloor\right)} \ . \tag{3.13}$$

This model was validated experimentally in Section 3.7, whose results are shown in Figure 3.8.

3.6.4 Selecting the threshold

Assume that a peer does not want to exchange his similarity with more than $r = 20\%$ of the peers he meets, which we call his *acceptance rate*. If the distribution of user-user similarities is known, then the threshold could be set to the r^{th} quantile. Otherwise, the peers could select the threshold in a preprocessing step using the inverse cumulative density function (inverse CDF, or F^{-1}) of the hypergeometric distribution in the following manner. The constraint can be expressed as

$$r = \Pr\left[\frac{S^2}{xy} > \tau\right] = 1 - F_S(\sqrt{xy\tau}) \ , \tag{3.14}$$

hence

$$F_S^{-1}(1-r) = \sqrt{xy\tau} \ , \tag{3.15}$$

and therefore

$$\tau = \left[F_S^{-1}(1-r)\right]^2 / xy \ . \tag{3.16}$$

A third solution in case the two peers wish to avoid the computational cost associated with securely evaluating the inverse CDF is for each peer to choose an initial value for the threshold at random then adjust it as he meets new peers, augmenting or decrementing in response to the observed acceptance ratio.

3.7 Experimental evaluation.

We have also studied experimentally the impact of the proposed mechanisms in the context of Gossple (*cf.* Section 2.2). In the baseline version of the clustering protocol (which we refer to simply as "baseline" thereafter), each peer samples the network and exchange his true profile with other peers, to compute pair-wise similarities. In our experiments, we compare three different privacy models against the baseline model.

Threshold In this model, a *threshold* similarity protocol is run, in which peers exchange their true similarity only if the threshold protocol (Algorithm 4), outputs 1. This protocol privately computes the similarity measure and outputs 1 if the similarity between the two peers exceeds the predetermined threshold τ. If a peer has in his view less than ℓ peers whose similarity is above τ, the rest of the view is chosen at random and the similarity are not transmitted.

Threshold & differentially private (TDP) This model is a variant of the threshold version in which we equip the cryptographic protocol with ε-differential privacy. In particular, two peers exchange their true similarity only if their perturbed true similarity (perturbed by adding Laplacian noise) is greater than the threshold. Computing similarity between two peers uses $O(n)$ bits, in which n is the number of items in the system.

Noisy Release In this model, the perturbed (noisy) similarity value is release directly without a threshold step. For this reason, this model is directly comparable to the non-interactive case in Chapter 5.

Evaluations are conducted through the simulation of a network of peers from the datasets described in Section 2.4. The threshold values have been selected as per Section 3.6.4. In particular, we choose the thresholds to be equal to the user-user similarity quantiles $\{50\%, 75\%, 85\%, 95\%\}$ for each dataset. We let the privacy parameter ε vary in the set $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$.

We evaluate the three models (threshold, TDP, and noisy release) according to the following metrics:

Utility The utility is measured in terms of the percentage of perfect view of the clustering view of a peer (Figure 3.10 and Figure 3.13), and the recall, defined in Section 2.2 (Figure 3.9 and Figure 3.12).

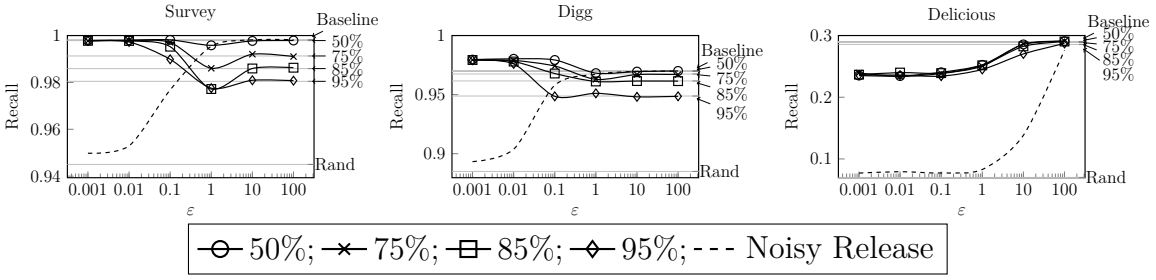


Figure 3.9 – Recall after convergence vs. several values of ε. The different lines denote different values for the threshold. The threshold "50%" is the 50% quantile of user-user similarity in this dataset and similarly for "75%" and so on. The grey lines denote the attained value of the recall when no noise have been added at all. The grey line denoted "Baseline" is the attained value of the recall when no noise have been added and no threshold have been applied. The grey line denoted "Rand" is the recall achieved by a totally random clustering of the peers.

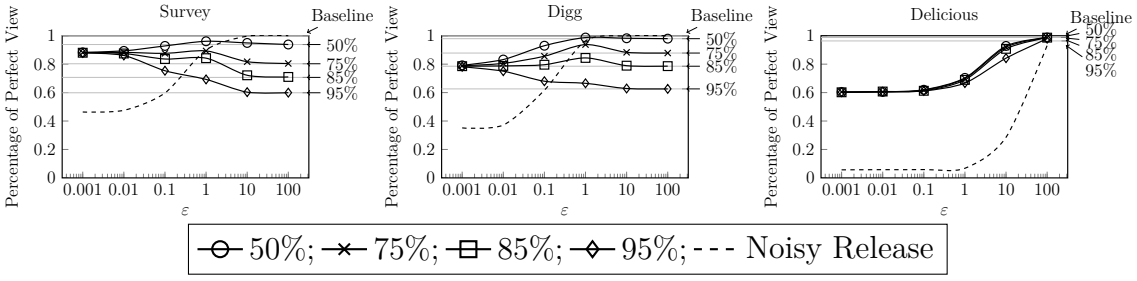


Figure 3.10 – Percentage of perfect view after convergence for vs. several values of ε.

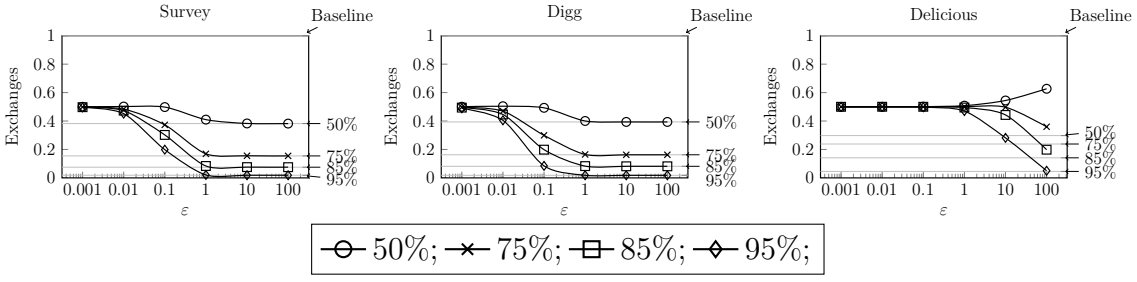


Figure 3.11 – Number of exchanges after convergence vs. several values of ε. The y-axis is the ratio between the number of exchanges to the total number of comparisons.

Privacy The privacy is measured as the number of encounters in which the true similarity was exchanged (Figure 3.11).

3.7.1 Results

We discuss the experimental results in this section.

Convergence The values for the recall and percentage of perfect view is plotted for every cycle of the protocol in Figure 3.12 and Figure 3.13. A first observation is that the different datasets take different number of cycles to converge. For the Survey dataset around 10 cycles are enough to converge for the percentage of perfect view, although the recall stabilizes on the 5^{th} cycle, indicating that an increase in the percentage of perfect view does not necessarily imply an equal increase in the value of the recall. For the Digg dataset the percentage of perfect view stabilizes around the 40^{th} cycle, however the recall demonstrates a more intriguing behavior. The baseline recall for Digg reaches a peak on the 6^{th} cycle then keeps decreasing until is levels off on the 50^{th} cycle. Taking into account that the corresponding percentage of perfect view increases while the recall decreases it appears that this behavior is due to overfitting. Therefore it becomes clear why the perturbed approach does not succumb to this overfitting phenomenon where its recall does not decrease. The Delicious dataset converges for the percentage of perfect view after 100 cycles while the recall levels off after 50 cycles. The number of peers of Digg and Delicious is roughly the same so it may not be the reason why Delicious takes double the number of cycles to converge. Instead, it might be related to the sparsity of the dataset (*cf.* Table 2.1).

After convergence In Figure 3.9, 3.10, and 3.11, the x-axis represents the privacy parameter ε. The larger ε is, the less privacy (*i.e.*, noise) is added.

One general observation that the higher the threshold value, the less the recall and percentage of perfect view are, giving less utility. However, the number of similarity exchanges also decreases, giving higher privacy. For lower values of ε the similarity exchanges approach 50% as expected, giving low privacy and high utility, regardless of the threshold. In particular, for Survey and Digg, for high and low values for ε, utility is high and privacy is low (for high ε exchanges are low but non-exchanges may reveal information). The right balance seems to be when the value of ε is in the middle ($\varepsilon = 1$), where privacy is high and utility is almost high. This explains the valley in the recall curves at $\varepsilon = 1$ (and corresponding peak in the percentage of perfect view curves).

Another observation is in Digg, in which we observe that the recall for low ε is higher than the baseline recall, while the corresponding percentage of perfect view is lower than the baseline percentage. This is explained by observing the evolution of the recall against cycles in Figure 3.12, in which we concluded in our analysis of the convergence that the baseline decreases due to overfitting.

For the privacy as measured by the number of exchanges of true similarity (when the perturbed similarity is higher than the threshold) in Figure 3.11, we observe that Delicious needs higher values of ε compared to the other datasets in order to activate

the thresholds. This is due to the sparsity of the dataset which makes the user-user similarity distribution much more concentrated around its mean.

To summarize, applying the threshold protocol impacts only slightly the recall, but reduces up to $50\% - 90\%$ the number of similarities exchanged, thus providing higher privacy. Hence, we conclude that it is possible to achieve reliable clustering and high recall when using a differentially private threshold mechanism before exchanging the true similarity between peers.

3.8 Conclusion

In this chapter, we have provided a secure multi-party protocol for the computation of pair-wise threshold similarity, in a way that addresses the unique set of challenges faces in a large scale dynamic P2P system. More precisely, we discussed how to handle the privacy budget issue through the bidirectional anonymous channel, then, we provided several methods for distributed noise generation, as well as a theoretical model for the false positives and false negatives resulting from the threshold similarity and validated it experimentally. Furthermore, we carried out simulations to validate that the utility (as measured by recall and the percentage of the perfect view), is still maintained at the same time while providing a high level of privacy.

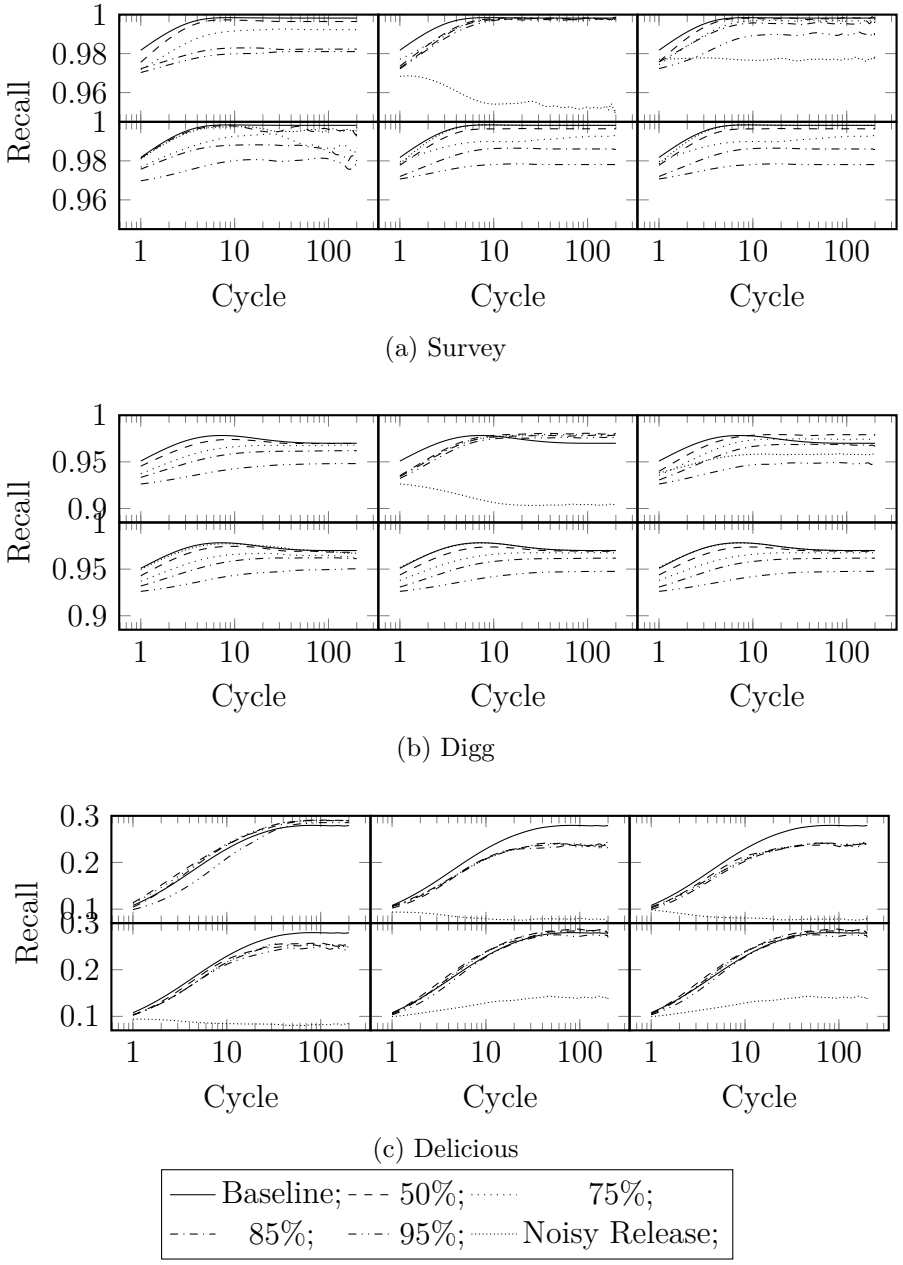
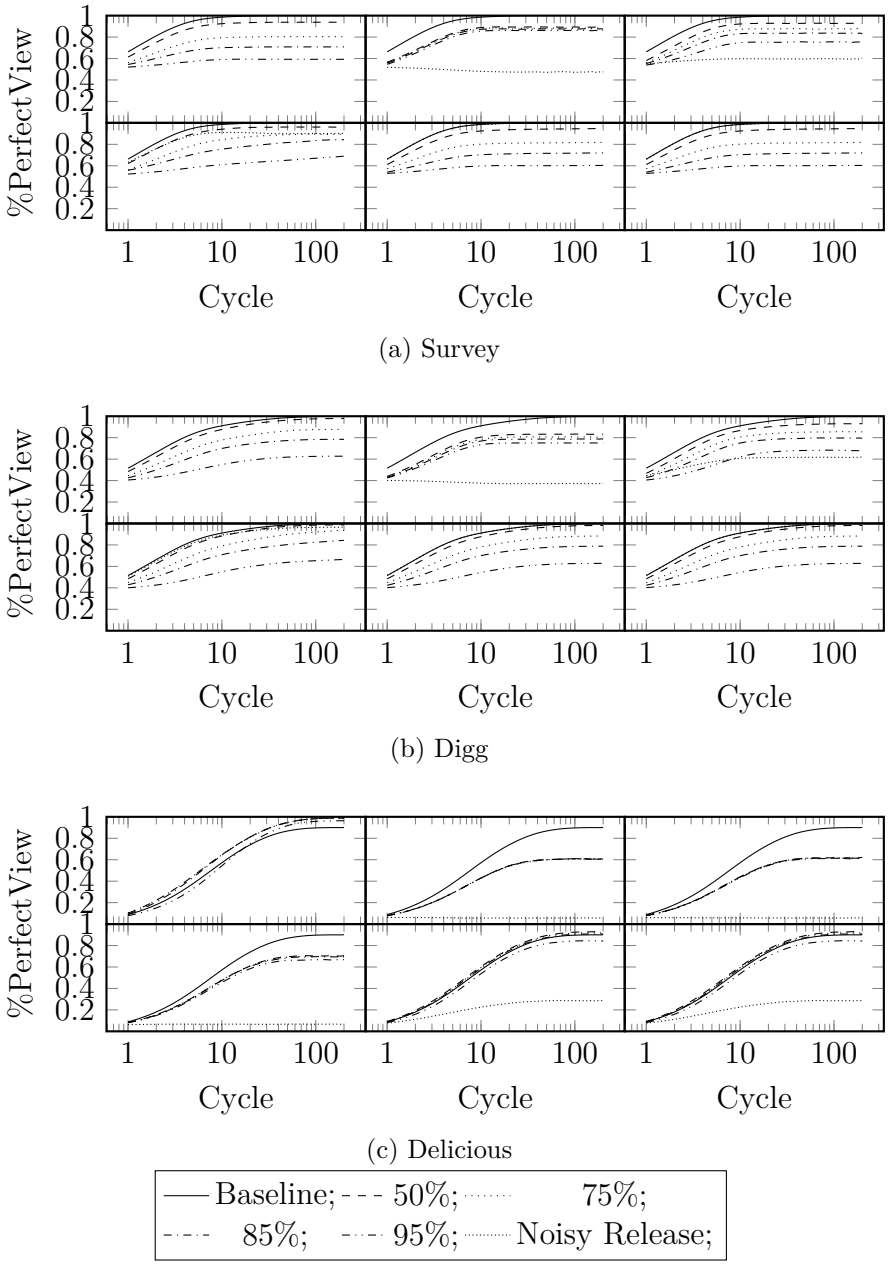


Figure 3.12 – The value of the recall vs. cycles. The first row gives the results for threshold-only protocol and for $\varepsilon = 0.01, 0.1$ (from left to right). The second row gives the results for $\varepsilon = 1, 10, 100$ from left to right.



(a) Survey

(b) Digg

(c) Delicious

Figure 3.13 – The value of the percentage of perfect view vs. cycles. The first row gives the results for threshold-only protocol and for $\varepsilon = 0.01, 0.1$ (from left to right). The second row gives the results for $\varepsilon = 1, 10, 100$ from left to right.

4

Heterogeneous Differential Privacy

Abstract

Most of the proposed approaches to preserve privacy in personalization systems usually address this issue uniformly across users, ignoring the fact that users have different privacy attitudes and expectations (even among their own personal data). In this chapter, we propose to account for this non-uniformity of privacy expectations by introducing the concept of heterogeneous differential privacy. This notion captures both the variation of privacy expectations among users as well as across different pieces of information related to the same user. We also describe an explicit mechanism achieving heterogeneous differential privacy, which is a modification of the Laplacian mechanism by Dwork, McSherry, Nissim, and Smith. The basic idea underlying the mechanism is manipulating the sensitivity of the function using a linear transformation of the input domain. Finally, we evaluate on real datasets the impact of the proposed mechanism with respect to Gossple. The results of our experiments demonstrate that heterogeneous differential privacy can account for different privacy attitudes while sustaining a good level of utility as measured by the recall.

4.1 Introduction

Most of the approaches to privacy implicitly assume homogeneity by considering that users have uniform privacy requirements [10, 90, 40, 21, 91, 92, 93, 94, 21, 27, 38, 26]. However, in an environment composed of a myriad of communities, such as the Internet, it is highly plausible that users have heterogeneous privacy attitudes and expectations. For instance, in a collaborative social platform like the ones we consider in this thesis, it is natural to expect that for a particular user, some items in his profile are considered more sensitive by him than others, thus calling for a system that can deal with different privacy requirements across items. Similarly, Alice might be more conservative about her privacy than Bob, requiring different privacy requirements across users.

This non-uniformity of privacy attitudes has been acknowledged by major social networking sites [95, 96]. For instance in Facebook, a user can now set individual privacy settings for each item in his profile. However in this particular example, privacy is mainly addressed by restricting, through an access-control mechanism, who is allowed to access and view a particular piece of information. Our approach can be considered to be orthogonal but complementary to access-control. More precisely, we consider a personalized service, such as recommendation algorithm, and we enforce the privacy requirements of the user on its output. Heterogeneous privacy requirements might also arise with respect to pictures, depending on the location in which the picture was taken or the persons appearing [96]. In the future, users are likely to expect item-grained privacy for other services[1].

[1]Note that systems supporting item-grained privacy can also provide user-grained privacy (*i.e.*, for instance by setting the privacy level of all items in some user's profile to the same value in the

Furthermore, as highlighted by Zwick and Dholakia in 1999 [97] and as evidenced by anthropological research, privacy attitudes are highly dependent on social and cultural norms. A similar point was raised in 2007 by Zhang and Zhao in a paper on privacy-preserving data mining [98] in which they mentioned that in practice it is unrealistic to assume homogeneous privacy requirements across a whole population. In particular, their thesis is that enforcing the same privacy level across all users and for all types of personal data could lead to an unnecessary degradation of the performance of such systems as measured in terms of accuracy. More specifically, enforcing the same privacy requirements upon all users (even those who do not require it) might degrade the performance in comparison to a system in which strict privacy requirements are only taken into account for those who ask for it. The same type of argument can also be made for different items of the same user. Hence, designing a system supporting heterogeneous privacy requirements could lead to a global improvement of the performance of this system as compared to a homogeneous version. Therefore, the main challenge is to be able to account for the variety of privacy requirements when leveraging personal data for recommendation and personalization.

In this chapter, we address this challenge through the introduction of the concept of *heterogeneous differential privacy*, which considers that the privacy requirements are not homogeneous across users and items from the same user (thus providing item-grained privacy). This notion can be seen as an extension of the concept of differential privacy [92] introduced originally by Dwork in the context of databases. We also describe an explicit mechanism achieving heterogeneous differential privacy, which we coin as the "stretching mechanism". This novel mechanism achieves heterogeneous differential privacy by modifying the sensitivity of the function to be released (and therefore the function itself) before applying the standard Laplacian mechanism. We derive a bound on the distortion introduced by our mechanism, which corresponds to a distance between the expected output of the mechanism and the original value of the function to be computed. Finally, we conduct an experimental evaluation of our mechanism on GOSSPLE (*cf.* Section 2.2) on three real datasets (*cf.* Section 2.4). The results obtained show that the proposed approach can still sustain a high utility level (as measured in terms of recall) while guaranteeing heterogeneous differential privacy.

The outline of the chapter is as follows. First, in Section 4.2, we describe the background of differential privacy as well as some preliminaries on matrices and sets necessary to understand our work. Then, in Section 4.3, we present the related work on heterogeneous privacy mechanisms. Afterwards in Section 4.4, we introduce the novel concept of heterogeneous differential privacy along with the description of an explicit mechanism achieving it. Then, we assess experimentally the impact of the proposed mechanism by evaluating it on GOSSPLE in Section 4.5. Lastly, we conclude with a discussion on the limitations of the approach as well as possible extensions in Section 4.6.

---

privacy setting of this user), and therefore the former can be considered as a generalization of the latter. However, this assumes that the privacy weights have a global meaning across the entire system, and are not defined only relative to a user.

4.2 Background

In this section, we briefly introduce the background on differential privacy as well as some notions on matrices and sets that are necessary to understand the concept of heterogeneous differential privacy

Before delving into the details of our approach, we need to briefly introduce some preliminary notions on matrices and sets such as the concept of *shrinkage matrix* [99]. A shrinkage matrix is a linear transformation that maps a vector to another vector with less magnitude, possibly distorting it by changing its direction.

Definition 4.1 (Shrinkage matrix)**.** A matrix A is called a shrinkage matrix if and only if $A = \text{diag}(\alpha_1, \ldots, \alpha_n)$ such that each diagonal coefficient is in the range $0 \leqslant \alpha_i \leqslant 1$.

For example, the matrix

$$\begin{pmatrix} 0.7 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

is a shrinkage matrix.

Definition 4.2 (Semi-balanced set)**.** A set $D \subseteq \mathbb{R}^n$ of column vectors is semi-balanced if and only if for all shrinkage matrices $A = \text{diag}(\alpha_1, \ldots, \alpha_n)$, and for all $\boldsymbol{x} \in D$, we have $A\boldsymbol{x} \in D$.

For instance, the set

$$\{\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2 \mid 0 < x_1, x_2 < 1\}$$

is a semi-balanced set (imagine a square from $(0, 0)$ to $(1, 1)$ in the Euclidean plane).

4.3 Related work

The majority of previous works on heterogeneous privacy has focused only on user-grained privacy [100, 101], in which each user may define his own privacy level (instead of having the same privacy guarantee for all users across the system). As opposed to item-grained privacy, which allows each item of an individual user to have a different privacy weight, user-grained privacy restricts all the items of the same user to the same privacy weight. For instance, Das, Bhaduri, and Kargupta [100] have proposed a secure protocol for aggregating sums in a P2P network. In this setting, each peer has an input vector, which could be for instance a profile. In a nutshell, in this protocol, each peer picks at random a few other peers of the system with whom it computes some local function[2] in a private manner (the local function begins with a sum as well). The more peers a specific peer chooses to participate to the computation, the higher the privacy will be obtained by this peer according to the considered definition of privacy. More precisely, in their setting,

[2]The function is local in the sense that it depends only on the inputs of the peer and the peers it has chosen.

privacy is mainly quantified by the probability of collusion of the peers chosen by a particular peer when the aggregation protocol is run. This probability can be made smaller by choosing a larger set of peers, the main intuition being that for a particular peer running the aggregation protocol with a larger group diminishes the probability that all these peers will collude against him. Thus, the best privacy guarantees could be obtained by running the protocol with the entire set of peers but this would be too costly in practice. The main objective is this protocol is to be adaptive by providing a trade-off between the privacy level chosen by a user and the resulting cost in terms of computation and communication. In particular, each user has the possibility to choose heterogeneously the peers with whom he wants to run the aggregation protocol by taking into account his own privacy preferences. However, this work does not seem to be easily extendable to integrate item-grained privacy.

Another work due to Kumar, Gopal, and Garfinkel [101] is a form of generalization of k-anonymity [102]. The standard definition of k-anonymity requires that in the sanitized database that is released, the profile of a particular individual should be indistinguishable from at least $k-1$ other individuals (thus here k can be considered as being the privacy parameter). The proposed generalization [101] essentially enables each user to require a different value for k for each attribute in his profile. For example, a user may require that his data should be included in the published database only if there are at least 4 other users sharing his ZIP code and at least 8 other users whose age difference with him is at most 3 years. The possibility of setting the range of a particular attribute could be regarded as item-grained heterogeneous privacy in the sense that an attribute whose privacy range is large is less likely to be useful for de-anonymizing the user than a less private attribute. To summarize, the main objective of this approach is to protect the privacy of a user by anonymizing it (*e.g.*, to prevent de-anonymization and linking attacks), while in our work the main objective is to prevent the possibility of inferring the presence or absence of a particular item in the profile.

A line of research on auctions *for* privacy has provided almost the same definition for the heterogeneous differential privacy as ours [103, 41]. The key difference from our contribution is that they do not provide a mechanism to realize heterogeneous difference privacy. Instead, they only use the definition to realize the privacy guarantees offered by the release. In their model, the participants are composed of a data analyst and a group of users. Each user has as input a private bit and the data analyst wants to estimate in a differentially-private manner a global function of the private bits of all users, such as the sum or the weighted sum. The data analyst is willing to pay each user for the loss of privacy he incurred by participating to this process. More precisely, each user i has a *privacy valuation* $v_i(\varepsilon_i) : \mathbb{R}_+ \to \mathbb{R}_+$ indicating the amount of his loss given the privacy guarantee he gets. The user has no control over ε_i (*i.e.*, the privacy guarantee he ends up with), which is decided solely by the auction mechanism. As such the valuation function v_i merely affects the payment of the user, as his payment is decided indirectly by the mechanism given the valuation function and is not decided directly by him. Therefore, our work is incomparable to theirs, because the privacy parameter ε_i acts mainly as an indication about the level of privacy reached, while in our setting the privacy parameter represents the

user's requirement about the privacy of a particular item of his profile. Moreover, in [103] it is stated that users finally end up having completely homogeneous privacy guarantees. More precisely, each user ends up having ε-differential privacy, with some ε being the same for all users. On the contrary in [41], users effectively have heterogeneous privacy guarantees. However, these guarantees are determined by the public weights of the auctioneer, which the auctioneer chooses so as to compute the weighted average of the users' inputs and independently of the privacy valuations of the users.

Finally, Nissim, Raskhodnikova and Smith [104] have investigated how the amount of noise necessary to achieve differential privacy can be tailored by taking into account to the particular inputs (*i.e.*, profiles) of participants, in addition to the sensitivity of the function considered. The main objective of this approach is to reduce the amount of noise that needs to be added to inputs that are not *locally sensitive* (*i.e.*, for which the output does not change much if only one item is changed). However, they also show that the amount of noise added may itself reveal information about the inputs. Hence, they defined a differentially private version formalizing the notion of local sensitivity called *smooth sensitivity*, guaranteeing that the amount of noise added is itself ε-differentially private. Similarly, we have ensured that for our notion of heterogeneous differential privacy, the amount of noise added is not impacted by the specific profile considered or by the privacy requirements formulated by a user. Rather, we have modified the function under consideration and its sensitivity, which also impacts the distortion induced of the output (*cf.*, Section 4.4.2). We have also proven that the privacy requirements of a user expressed in the form of *private weights* remain private are they are also covered by ε-differentially privacy guarantees. Thus, it is difficult for an adversary observing the output of an heterogeneous differentially private mechanism to guess the privacy weight that a user has put on a particular item of his profile.

4.4 Heterogeneous differential privacy

In this section, we introduce the novel concept of *heterogeneous differential privacy* (HDP). We start by giving the necessary definitions in Section 4.4.1, before describing in Section 4.4.2 how to construct the stretching mechanism, which ensures heterogeneous differential privacy. More precisely, we first detail how to construct the privacy-preserving estimator. Afterwards, we discuss why and how the privacy vector expressing the privacy expectations of a user should also be kept private. Finally, an upper bound on the distortion induced by the stretching mechanism is provided.

4.4.1 Definitions

We now define HDP-specific notions such as the concept of *privacy vector*, which is a key notion in HDP. This vector contains the privacy requirements of each coordinate (*i.e.*, item) in the input profile (*i.e.*, vector) of a user, and is defined as follows.

Definition 4.3 (Privacy vector)**.** Given a profile $\boldsymbol{d} \in D$ in which D is a semi-balanced set of column vectors composed of n coordinates, let $\boldsymbol{v} \in [0,1]^n$ be the privacy vector associated with the profile $\boldsymbol{d}$. The owner of item d_i is responsible for choosing the privacy weight v_i associated to this item (by default v_i is set to be 1 if it was not explicitly specified by the owner). A privacy weight v_i of zero corresponds to *absolute privacy* while a value of 1 refers to *standard privacy*, which in our setting directly correspond to the classical definition of ε-differential privacy.

The mere presence of the privacy vector introduces potential privacy breaches, thus this vector should also be protected. Therefore, we need to ensure that in addition to the profile, the privacy vector $\boldsymbol{v}$ also remains private, such that each entry v_i of this vector should only be known by its owner. Otherwise, the knowledge of a privacy weight of a particular item might leak information about the profile itself. For instance, learning that some items have a high privacy weight may reveal that the user has high privacy expectations for and is therefore interested in this specific type of data. We define *heterogeneous differential privacy* in the following manner.

Definition 4.4 ((Heterogeneous) $(\varepsilon, \boldsymbol{v})$-differential privacy)**.** A randomized function $\mathcal{M} : D \to \mathbb{R}$ is said to be $(\varepsilon, \boldsymbol{v})$-differential privacy if for all items i, for all neighboring profiles $\boldsymbol{d} \sim \boldsymbol{d}^{(i)}$, and for all possible outputs $t \in \mathbb{R}$ of this function, the following statement holds:

$$\Pr[\mathcal{M}(\boldsymbol{d}) = t] \leqslant \exp(\varepsilon v_i) \Pr[\mathcal{M}(\boldsymbol{d}^{(i)}) = t] \ , \tag{4.1}$$

in which exp refers to the exponential function.

Since a privacy weight $v_i \leqslant 1$, heterogeneous differential privacy implies the standard notion of ε-differential privacy as shown by the following remark.

Remark 4.1 (Equivalence of $(\varepsilon, \boldsymbol{v})$-DP and ε-DP.). Let $\overline{\varepsilon} = \varepsilon \overline{v}$ and $\underline{\varepsilon} = \varepsilon \underline{v}$, such that $\overline{v} = \max_i v_i$ (the maximum privacy weight) and $\underline{v} = \min_i v_i$ (the minimum privacy weight). Then, we have: $\underline{\varepsilon}$-DP $\implies$ $(\varepsilon, \boldsymbol{v})$-DP and $(\varepsilon, \boldsymbol{v})$-DP $\implies$ $\overline{\varepsilon}$-DP . As a consequence, $(\varepsilon, \mathbf{1})$-DP holds *if and only if* ε-DP also holds, in which $\mathbf{1} = (1, \cdots, 1)$.

4.4.2 The stretching mechanism

Thereafter, we describe a generic mechanism achieving heterogeneous differential privacy that we coin as the *Stretching Mechanism*. We assume that the privacy preferences for each item are captured through a privacy vector $\boldsymbol{v}$ (*cf.* Definition 4.3). Given an arbitrary *total* function $f : D \to \mathbb{R}$, in which D is a semi-balanced set of columns vectors of n coordinates, and whose global sensitivity $S(f)$ is finite, we construct a randomized function $\hat{f}(\boldsymbol{d}, \boldsymbol{v}, \varepsilon)$ *estimating* f while satisfying $(\varepsilon, \boldsymbol{v})$-differential privacy.

Before delving into the details of this method, we provide a little intuition on how and why it works. A lemma in [41, Lemma 1] asserts that the Laplacian mechanism $\mathcal{M}(\boldsymbol{d}) = f(\boldsymbol{d}) + \mathsf{Lap}(\sigma)$ with mean 0 and standard deviation σ provides

$$\Pr[\mathcal{M}(\boldsymbol{d}) = t] \leqslant \exp(\varepsilon_i) \Pr[\mathcal{M}(\boldsymbol{d}^{(i)}) = t] \ , \tag{4.2}$$

in which $\varepsilon_i = S_i(f)/\sigma$. Thus, the achieved level of differential privacy for a particular coordinate when the Laplacian mechanism is used, depends on the modular sensitivity. Therefore, a natural approach for enforcing heterogeneous differential privacy is to manipulate the modular sensitivity $S_i(f)$ by modifying the function f itself.

Constructing the estimator. Let $T : [0,1]^n \to \mathbb{R}^{n\times n}$ be a function taking as input a privacy vector $\boldsymbol{v}$ and returning as output a shrinkage matrix, with the property that $T(\mathbf{1}) = I$, such that I is the identity matrix and $\mathbf{1} = (1, \cdots, 1)$. Let also R be a mapping sending a function $f : D \to \mathbb{R}$ and a privacy vector $\boldsymbol{v} \in [0,1]^n$ to the function $R(f, \boldsymbol{v}) : D \to \mathbb{R}$ with

$$R(f, \boldsymbol{v})(\boldsymbol{d}) = f(T(\boldsymbol{v}) \cdot \boldsymbol{d}) \ . \tag{4.3}$$

Recall that the Laplace distribution centered at 0 with scale parameter σ has the following probability density function

$$h(x) = \exp(-|x|/\sigma)/2\sigma \ . \tag{4.4}$$

Finally, let N be a Laplacian random variable with parameter $\sigma = \sigma(f, \varepsilon) = S(f)/\varepsilon$, in which $S(f)$ refers to the global sensitivity of the function f and ε the privacy parameter. The following statement proves that this *Stretching Mechanism* R (4.3) satisfies heterogeneous differential privacy.

Theorem 4.1 (Achieving HDP via stretching mecanism)**.** *Given a privacy vector $\boldsymbol{v}$, if the function $T(\boldsymbol{v})$ satisfies $S_i(R(f, \boldsymbol{v})) \leqslant v_i S(f)$ then the randomized function $\hat{f}(\boldsymbol{d}, \boldsymbol{v}, \varepsilon) = R(f, \boldsymbol{v})(\boldsymbol{d}) + N$ satisfies $(\varepsilon, \boldsymbol{v})$-differential privacy.*

Proof. For all two neighboring profiles $\boldsymbol{d}, \boldsymbol{d}^{(i)}$, and for all outputs $t \in \mathbb{R}$ of the function f we have

$$\frac{\Pr[\hat{f}(\boldsymbol{d}, \boldsymbol{v}, \varepsilon) = t]}{\Pr[\hat{f}(\boldsymbol{d}^{(i)}, \boldsymbol{v}, \varepsilon) = t]} = \frac{h(t - R(f, \boldsymbol{v})(\boldsymbol{d}))}{h(t - R(f, \boldsymbol{v})(\boldsymbol{d}^{(i)}))} \tag{4.5}$$

$$\leqslant \exp(\frac{\varepsilon |R(f, \boldsymbol{v})(\boldsymbol{d}) - R(f, \boldsymbol{v})(\boldsymbol{d}^{(i)})|}{S(f)}) \tag{4.6}$$

$$\leqslant \exp(\frac{\varepsilon S_i(R(f, \boldsymbol{v}))}{S(f)}) \tag{4.7}$$

$$\leqslant \exp(\frac{\varepsilon v_i \cancel{S(f)}}{\cancel{S(f)}}) = \exp(\varepsilon v_i) \ , \tag{4.8}$$

in which $h(\cdot)$ is defined as (4.4), thus proving the result. □

In a nutshell, $T(\boldsymbol{v})$ is a shrinkage matrix, whose shrinking factor in each coordinate is computed independently of all other coordinates. More precisely, the shrinking factor for a particular item depends only on the privacy weight associated to this coordinate. The value used by the mechanism is the lowest amount of shrinkage (*i.e.*, distortion) still achieving the target modular sensitivity of that coordinate. In the following section we provide an explicit construction of $T(\boldsymbol{v})$ for which we prove (Lemma 4.1) that the condition of Theorem 4.1 is satisfied, and therefore that $\hat{f}$ achieves $(\varepsilon, \boldsymbol{v})$-differential privacy.

Computing the shrinkage matrix. The HDP mechanism $\hat{f}(\boldsymbol{d}, \boldsymbol{v}, \varepsilon)$ adds Laplacian noise to a modified function

$$R(f, \boldsymbol{v})(\boldsymbol{d}) = f(T(\boldsymbol{v}) \cdot \boldsymbol{d}) \ . \tag{4.9}$$

In this section, we specify how to construct $T(\boldsymbol{v})$ such that $\hat{f}$ satisfies HDP. Thereafter, we use R to denote $R(f, \boldsymbol{v})$ for the sake of simplicity. Let $T(\boldsymbol{v}) = \mathrm{diag}(\boldsymbol{w})$ for some $\boldsymbol{w} \in [0, 1]^n$ to be computed from the privacy vector $\boldsymbol{v}$ and $S(R, \boldsymbol{w})$ be the sensitivity of $R = f(T(\boldsymbol{v}) \cdot \boldsymbol{d}) = f(\mathrm{diag}(\boldsymbol{w}) \cdot \boldsymbol{d})$ given $\boldsymbol{w}$. Similarly, let $S_i(R, \boldsymbol{w})$ be the modular sensitivity of R given $\boldsymbol{w}$. We denote by $(\boldsymbol{w}_{-i}, w_i')$ the vector resulting from replacing the item w_i in $\boldsymbol{w}$ to w_i' (*e.g.*, $(\mathbf{1}_{-i}, w_i) = (1, \ldots, w_i, \ldots, 1)$). Each w_i can be computed from v_i by solving the following optimization problem:

$$\begin{array}{ll} \text{maximize} & w_i \ , \\ \text{subject to:} & S_i(R, (\mathbf{1}_{-i}, w_i)) \leqslant v_i S(f) \ . \end{array} \tag{4.10}$$

Note that a solution satisfying this constraint always exists and is reached by setting w_i to 0. The w_i's are never released after they have been computed locally by the rightful owner, and the modular sensitivity $S_i(R)$ is only used in the proof and is not revealed to the participants as well as the noise generated. The participants only have the knowledge of the global sensitivity $S(f)$. Therefore, the only way in which the profile $\boldsymbol{d}$ could leak is through its side effects to the output, which we prove to achieve ε-DP in Theorem 4.2.

Lemma 4.1 (Premise of Theorem 4.1)**.** *If $T(\boldsymbol{v}) = \mathrm{diag}(\boldsymbol{w})$ such that for all i:*

$$S_i(R, (\mathbf{1}_{-i}, w_i)) \leqslant v_i S(f) \tag{4.11}$$

(the constraint of (4.10)*), then R satisfies:*

$$S_i(R, \boldsymbol{w}) \leqslant v_i S(f) \tag{4.12}$$

for all i.

This lemma follow from the following series of facts.

Proposition 4.1 (Monotonicity of subdomain optimization)**.** *Let θ and θ' be the result of two maximization problems p_1 and p_2 of the function g in which the maximization is over domains J and J', respectively. Then, if $J \subseteq J'$, this implies that $\theta \leqslant \theta'$. The opposite statement also holds for minimization problems.*

Proof. Since θ' is the optimal result of p_2 over J', this means that by definition:

$$g(\theta') \geqslant g(j) \ , \tag{4.13}$$

for all j in J'.

Moreover, since any result θ for p_1 will always be in J, and therefore in J', then $g(\theta') \geqslant g(\theta)$ by (4.13). The proof that the opposite statement holds for minimization problems follows from the same arguments and therefore we choose to omit it. □

Lemma 4.2 (Shrinkage matrices composition)**.** *If A and B are two shrinkage matrices and D a semi-balanced set, then $ABD \subseteq BD \subseteq D$.*

Proof. By definition of semi-balanced set, we have $BD \subseteq D$. Then it remains to prove that $ABD \subseteq BD$ (or equivalently, that BD is a semi-balanced set). We observe that a vector $\boldsymbol{w}$ belongs to ABD if and only if $\boldsymbol{w} = AB\boldsymbol{b}$ for some $\boldsymbol{b} \in D$. Because shrinking matrices commute, $\boldsymbol{w} = BA\boldsymbol{b}$. Let $\boldsymbol{a} = A\boldsymbol{b}$, then by definition of semi-balanced set, $\boldsymbol{a} \in D$. Therefore, $\boldsymbol{w} = B\boldsymbol{a}$ for $\boldsymbol{a} \in D$, which means $\boldsymbol{w}$ belongs to BD by definition of BD. □

Lemma 4.3 (Monotonicity of the global sensitivity)**.** *If $\boldsymbol{w}' \leqslant \boldsymbol{w}$ then $S(R, \boldsymbol{w}') \leqslant S(R, \boldsymbol{w})$.*

Proof. Let $\boldsymbol{c}$ be such that

$$c_i = \begin{cases} w'_i/w_i & \text{if } w_i \neq 0 \\ 0 & \text{otherwise} \end{cases} , \tag{4.14}$$

and let $C = \operatorname{diag}(\boldsymbol{c})$ is a shrinkage matrix. We have that $\boldsymbol{w}' = C\boldsymbol{w}$. Let $T' = \operatorname{diag}(\boldsymbol{w}')$ and $T = \operatorname{diag}(\boldsymbol{w})$ be two other shrinkage matrices. In this case $T' = CT$. By Lemma 4.2 and since D is semi-balanced:

$$T'D = CTD \subseteq TD \subseteq D \ . \tag{4.15}$$

The result follows from Proposition 4.1 because $S(R, \boldsymbol{w})$ is over the domain TD while $S(R, \boldsymbol{w}')$ is a maximization problem over the domain $T'D \subseteq TD$. □

Corollary 4.1 (Monotonicity of the modular sensitivity)**.** *If $\boldsymbol{w}' \leqslant \boldsymbol{w}$ then $S_i(R, \boldsymbol{w}') \leqslant S_i(R, \boldsymbol{w})$ for all i.*

Proof. By Lemma 4.3, we have that $S(R, \boldsymbol{w}') \leqslant S(R, \boldsymbol{w})$. Let $i^* = \arg\max_i S_i(R, \boldsymbol{w})$ and therefore $S(R, \boldsymbol{w}) = S_{i^*}(R, \boldsymbol{w})$. In order to get a contradiction, we assume that $S_{i^*}(R, \boldsymbol{w}') > S_{i^*}(R, \boldsymbol{w})$, thus we have

$$S(R, \boldsymbol{w}') = \max_i S_i(R, \boldsymbol{w}') > S_{i^*}(R, \boldsymbol{w}) = S(R, \boldsymbol{w}) \ , \tag{4.16}$$

which is a contradiction. □

Now we can finally prove Lemma 4.1.

Proof of Lemma 4.1. Since $\boldsymbol{w} \leqslant (\mathbf{1}_{-i}, w_i)$ for all i, then:

$$S_i(R, \boldsymbol{w}) \leqslant S_i(R, (\mathbf{1}_{-i}, w_i)) \text{ for all } i, \tag{4.17}$$
$$\leqslant v_i S(f) \quad \text{for all } i, \tag{4.18}$$

in which the first inequality follows from Corollary 4.1 and the second inequality follows from the premise of the lemma, thus concluding the proof. □

Hiding the privacy vector. The privacy weights by themselves, if released publicly, could reveal private information [103, 41]. Therefore, their impact on the observable output of the mechanism should be studied. The following theorem states that when the profile $\boldsymbol{d}$ is fixed, the randomized function $\hat{f}$ satisfies ε-differential privacy over neighboring privacy vectors $\boldsymbol{v} \sim \boldsymbol{v}^{(i)}$. Thus, the privacy vector can also be considered to be hidden and protected by the guarantees of differential privacy.

Theorem 4.2 (Protecting the privacy vector with ε-DP)**.** *The randomized function $\hat{f}$ provides ε-differential privacy for each individual privacy weight of $\boldsymbol{v}$. This means that for all neighboring privacy vectors $\boldsymbol{v} \sim \boldsymbol{v}^{(i)}$, for all outputs $t \in \mathbb{R}$ and profiles $\boldsymbol{d}$, the following statement holds:*

$$\Pr[\hat{f}(\boldsymbol{d}, \boldsymbol{v}, \varepsilon) = t] \leqslant \exp(\varepsilon) \Pr[\hat{f}(\boldsymbol{d}, \boldsymbol{v}^{(i)}, \varepsilon) = t] \ .$$

We will need the following two propositions to prove this theorem.

Proposition 4.2 (Semi-balanced sets are closed under shrinkage.)**.** *If D is a semi-balanced set and A is a shrinkage matrix, then AD is also a semi-balanced set.*

Proof. Follows from the proof of Lemma 4.2. □

Proposition 4.3 ($T(\boldsymbol{v})$ and $T(\boldsymbol{v}^{(i)})$ are neighbors)**.** *$T(\boldsymbol{v}).\boldsymbol{d}$ and $T(\boldsymbol{v}^{(i)}) \cdot \boldsymbol{d}$ are neighbors on item i, since w_i is a function of v_i only.*

We can now prove Theorem 4.2.

Proof of Theorem 4.2. Let $\boldsymbol{d}_* = T(\boldsymbol{v}) \cdot \boldsymbol{d}$ and $\boldsymbol{d}_*^{(i)} = T(\boldsymbol{v}^{(i)}) \cdot \boldsymbol{d}$. By Proposition 4.3, $\boldsymbol{d}_*$ and $\boldsymbol{d}_*^{(i)}$ are neighboring profiles. Moreover due to Proposition 4.2, they still belong to D, thus $|f(\boldsymbol{d}_*) - f(\boldsymbol{d}_*^{(i)})| \leqslant S_i(f) \leqslant S(f)$. Therefore, we have:

$$\frac{\Pr[\hat{f}(\boldsymbol{d}, \boldsymbol{v}, \varepsilon) = t]}{\Pr[\hat{f}(\boldsymbol{d}, \boldsymbol{v}^{(i)}, \varepsilon) = t]} = \frac{h(t - R(f, \boldsymbol{v})(\boldsymbol{d}))}{h(t - R(f, \boldsymbol{v}^{(i)})(\boldsymbol{d}))} \tag{4.19}$$

$$\leqslant \exp\left(\frac{\varepsilon |R(f, \boldsymbol{v})(\boldsymbol{d}) - R(f, \boldsymbol{v}^{(i)})(\boldsymbol{d})|}{S(f)}\right) \tag{4.20}$$

$$= \exp\left(\frac{\varepsilon |f(\boldsymbol{d}_*) - f(\boldsymbol{d}_*^{(i)})|}{S(f)}\right) \tag{4.21}$$

$$\leqslant \exp\left(\frac{\varepsilon \cancel{S(f)}}{\cancel{S(f)}}\right) = \exp(\varepsilon) \ , \tag{4.22}$$

in which $h(\cdot)$ is defined in (4.4), thus proving the result. □

Estimating the distortion induced by HDP. Let f be a continuous and differentiable function on a semi-balanced set D, and let $(\boldsymbol{v}, \boldsymbol{d}) \in [0, 1]^n \times D$ be respectively, the privacy vector and the profile considered. The following theorem provides a bound on the distortion introduced on the output by modifying the global sensitivity of the function f as done by the HDP (*i.e.*, stretching) mechanism described previously.

Theorem 4.3 (Bound on the distortion induced by the stretching mechanism)**.** *Let $f : D \to \mathbb{R}$ be a function from a semi-balanced set D to the reals, and let $\boldsymbol{v} \in [0,1]^n$ be a privacy vector and $T : [0,1]^n \to \mathbb{R}^{n\times n}$ be a function taking a privacy vector to a shrinkage matrix. Finally, let R be a mapping sending a function f and a privacy vector $\boldsymbol{v}$ to the function $R(f,\boldsymbol{v}) : D \to \mathbb{R}$ such that $R(f,\boldsymbol{v})(\boldsymbol{d}) = f(T(\boldsymbol{v})\cdot\boldsymbol{d})$ for all vectors $\boldsymbol{d}$. The distortion (*i.e.*, distance) between f and $R(f,\boldsymbol{v})$ is bounded by:*

$$|f(\boldsymbol{d}) - R(f,\boldsymbol{v})(\boldsymbol{d})| \leqslant (1-\underline{w})\|\boldsymbol{d}\| \left(\max_{0\leqslant c\leqslant 1}\|\nabla f(B\cdot\boldsymbol{d})\|\right) \ , \tag{4.23}$$

where $B = cI + (1-c)T(\boldsymbol{v})$, $\underline{w} = \min_i w_i$ is the minimum of $\boldsymbol{w}$ (the diagonal of $T(\boldsymbol{v})$), and ∇f is the gradient of the function f.

Proof. Let $\boldsymbol{y} = \boldsymbol{d}$ and $\boldsymbol{x} = T(\boldsymbol{v})\cdot\boldsymbol{d}$, then by the mean value theorem [105, Theorem 14.4, p. 301], there exists a constant $0 \leqslant c \leqslant 1$ depending on $\boldsymbol{d}$ and $T(\boldsymbol{v})$ and f such that:

$$f(\boldsymbol{y}) - f(\boldsymbol{x}) = \nabla f((1-c)\boldsymbol{x} + c\boldsymbol{y})\cdot(\boldsymbol{y}-\boldsymbol{x}) \ , \tag{4.24}$$

in which $\cdot$ denotes the inner product. Therefore, by the Cauchy-Schwarz inequality, we have:

$$|f(\boldsymbol{y}) - f(\boldsymbol{x})| \leqslant \|\nabla f((1-c)\boldsymbol{x} + c\boldsymbol{y})\|\|\boldsymbol{y}-\boldsymbol{x}\| \ . \tag{4.25}$$

Finally, by the fact that

$$\begin{aligned}\|\boldsymbol{y}-\boldsymbol{x}\| &= \|\boldsymbol{d} - T(\boldsymbol{v})\cdot\boldsymbol{d}\| = \|(I - T(\boldsymbol{v}))\cdot\boldsymbol{d}\| && (4.26)\\ &= \sqrt{\sum_i ((1-w_i)d_i)^2} && (4.27)\\ &\leqslant (1-\min_i w_i)\sqrt{\sum_i d_i^2} = (1-\underline{w})\|\boldsymbol{d}\| \ , && (4.28)\end{aligned}$$

in which $T(\boldsymbol{v}) = \mathrm{diag}(\boldsymbol{w})$, and that

$$\begin{aligned}(1-c)\boldsymbol{x} + c\boldsymbol{y} &= (1-c)T(\boldsymbol{v})\cdot\boldsymbol{d} + c\boldsymbol{d} && (4.29)\\ &= (cI + (1-c)T(\boldsymbol{v}))\cdot\boldsymbol{d} \ , && (4.30)\end{aligned}$$

the theorem follows directly. □

Intuitively, if $\underline{w}$ is the minimum of $\boldsymbol{w}$ (the diagonal of $T(\boldsymbol{v})$), and $\boldsymbol{d}$ is the input profile, then the distortion introduced by *stretching* the function as measured in terms of absolute additive error is bounded by $1-\underline{w}$ times the norm of $\boldsymbol{d}$ multiplied by the norm of the gradient of the *semi-stretched* function at $\boldsymbol{d}$.

This bound is particularly useful in situations in which the norm of the gradient of the function f is bounded from above by a constant. However, even in the cases in which the norm of the gradient is not bounded by a constant, the bound can still be useful. For instance, in the case of the inner product function $f(\boldsymbol{d}) = \sum_{i=1}^{n/2} d_i d_{i+n/2}$, the gradient is $\|B\cdot\boldsymbol{d}\| \leqslant \|\boldsymbol{d}\|$ (since B is a shrinkage matrix). Hence, the bound on the distortion will be $(1-\underline{w})\|\boldsymbol{d}\|^2$, and since in case of the inner product function $\boldsymbol{w} = \boldsymbol{v}$, the distortion is $(1-\underline{v})\|\boldsymbol{d}\|^2 \leqslant (1-\underline{v})n^2$. This distortion corresponds to the bias of the estimator, which is distinct from the actual expected error due to the

randomization. From this, some constraints may be enforced on $\underline{v}$ to guarantee any desired upper bound on the distortion. For instance, if the distortion should be less than $\log(n)^d\sqrt{n}$ for some $d > 0$, then $\underline{v}$ should be greater than $1 - \log(n)^d/n^{3/2}$. One restriction on the application of this bound is that the function f to be protected should have a finite global sensitivity, and therefore the inner product function mentioned has to restrict its domain to be finite, thus preventing the distortion bound from being infinite.

4.5 HDP in practice

To assess the practicality of our approach, we have applied the HDP mechanism to GOSSPLE (Section 2.2). In this context, we are interested in providing heterogeneous differential privacy guarantees to peers. More precisely, we consider the scenario in which a particular peer can assign a privacy weight, between 0 and 1, to each item of his profile. The value 0 corresponds to the strongest privacy guarantee in the sense that the presence (or absence) of this item will not affect the outcome (the clustering) at all, while the value 1 is the *lowest* level of privacy possible in our framework. However, even this lowest level of privacy still provides the standard guarantees of ε-differential privacy. Thus, the privacy weights of a peer directly reflect his privacy attitudes with respect to particular items of his profile, and as a side effect determines the influence of this item in the clustering process. In particular, an item with a higher weight will contribute more to the clustering process, while a item with a lower weight will influence less the resulting clustering.

Let $\mathsf{SP}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y}$ refers to the inner product between the two profiles. The privacy vector $\boldsymbol{v}$ is composed of two parts, one for the profile $\boldsymbol{x}$ and the other for the profile $\boldsymbol{y}$: $(\boldsymbol{v}^{\boldsymbol{x}}, \boldsymbol{v}^{\boldsymbol{y}})$. Consider the matrix $T(\boldsymbol{v}) = \mathrm{diag}(\boldsymbol{v})$ and let $R(\mathsf{SP}, \boldsymbol{v}) = \mathsf{SP}(T(\boldsymbol{v}^{\boldsymbol{x}}) \cdot \boldsymbol{x}, T(\boldsymbol{v}^{\boldsymbol{y}}) \cdot \boldsymbol{y})$ be the Stretching Mechanism, in which T is the stretch specifier (which describes how to stretch every coordinate). This mechanism R satisfies the premise of Theorem 4.1 and therefore the choice of $T(\boldsymbol{v}) = \mathrm{diag}(\boldsymbol{v})$ also ensures HDP, as proven in the following lemma.

Lemma 4.4. *Consider a matrix $T(\boldsymbol{v}) = \mathrm{diag}(\boldsymbol{v})$ and a mechanism $R(\mathsf{SP}, \boldsymbol{v}) = \mathsf{SP}(T(\boldsymbol{v}^{\boldsymbol{x}}) \cdot \boldsymbol{x}, T(\boldsymbol{v}^{\boldsymbol{y}}) \cdot \boldsymbol{y})$, such that $\boldsymbol{x}$ and $\boldsymbol{y}$ correspond to profiles, $\boldsymbol{v}^{\boldsymbol{x}}$ and $\boldsymbol{v}^{\boldsymbol{y}}$ to their associated privacy vectors, and $\boldsymbol{v} = (\boldsymbol{v}^{\boldsymbol{x}}, \boldsymbol{v}^{\boldsymbol{y}})$. In this situation, the following statement is true for all i:*

$$S_i(R(\mathsf{SP}, \boldsymbol{v})) \leqslant v_i S(\mathsf{SP}) \ . \tag{4.31}$$

Proof. Each profile being represented as a binary vector, the global sensitivity of the inner product is one (*i.e.*, $S(\mathsf{SP}) = 1$). Thereafter, for the sake of simplicity, let R denotes $R(\mathsf{SP}, \boldsymbol{v})$. As $T(\boldsymbol{v})$ is a diagonal matrix, it is strictly identical to its transpose $T(\boldsymbol{v})^\top$. We can assume without loss of generality that $\boldsymbol{d}^{(i)} = (\boldsymbol{x}, \boldsymbol{y}^{(j)})$ for item $j = i - \dim(x)$, and therefore that:

$$S_i(R) = \max_{\boldsymbol{d} \sim \boldsymbol{d}^{(i)}} |T(\boldsymbol{v}^{\boldsymbol{x}})\boldsymbol{x} \cdot T(\boldsymbol{v}^{\boldsymbol{y}})\boldsymbol{y} - T(\boldsymbol{v}^{\boldsymbol{x}})\boldsymbol{x} \cdot T(\boldsymbol{v}^{\boldsymbol{y}})\boldsymbol{y}^{(j)}| \tag{4.32}$$

$$= \max_{\boldsymbol{d} \sim \boldsymbol{d}^{(i)}} |(\boldsymbol{x}^\top T(\boldsymbol{v}^{\boldsymbol{x}}) T(\boldsymbol{v}^{\boldsymbol{y}})) \cdot (\boldsymbol{y} - \boldsymbol{y}^{(j)})| \ . \tag{4.33}$$

However, the vector $\boldsymbol{y} - \boldsymbol{y}^{(j)}$ has all its coordinates set to 0 except for the j^{th} coordinate. Therefore, the maximum is reached when $y_j = 1$, $\boldsymbol{x} = \mathbf{1} = (1, \cdots, 1)$, and is such that:

$$\boldsymbol{v}_j^{\boldsymbol{x}} \boldsymbol{v}_j^{\boldsymbol{y}} \leqslant v_i = v_i \times 1 = v_i S(\mathsf{SP}) \ , \tag{4.34}$$

which concludes the proof. □

The previous lemma proves that the proposed modified version of inner product is differentially private (by Theorem 4.1), while the next lemma (which is a heterogeneous variant of the post-processing lemma, *cf.* Lemma 2.1) simply states that if we rely on this differentially private version of inner product to compute the cosine similarity (or any similar metric), the outcome of this computation will still be differentially private.

Lemma 4.5 (Effect of post-processing on HDP)**.** *If a randomized function $\hat{f}$ satisfies $(\varepsilon, \boldsymbol{v})$-differential privacy, then for any randomized function $g : \mathsf{Range}(\hat{f}) \to \mathbb{R}$ independent of the input to $\hat{f}$, the composed function $g \circ \hat{f}$ also satisfies $(\varepsilon, \boldsymbol{v})$-differential privacy. The randomness of the function g is assumed to be independent of the randomness of $\hat{f}$ in order for this property to hold.*

Proof. The theorem is equivalent to prove that for any two neighboring profiles $\boldsymbol{d} \sim \boldsymbol{d}^{(i)}$ the following holds:

$$\Pr[g \circ \hat{f}(\boldsymbol{d}) = t] \leqslant \exp(\varepsilon v_i) \Pr[g \circ \hat{f}(\boldsymbol{d}^{(i)}) = t] \ . \tag{4.35}$$

To prove this, consider any two neighboring profiles $\boldsymbol{d} \sim \boldsymbol{d}^{(i)}$:

$$\Pr[g \circ \hat{f}(\boldsymbol{d}) = t] = \int_{s \in \mathsf{Range}(\hat{f})} \Pr[\hat{f}(\boldsymbol{d}) = s] \cdot \Pr[g(s) = t] \tag{4.36}$$

$$\leqslant \int_{s \in \mathsf{Range}(\hat{f})} \exp(\varepsilon v_i) \Pr[\hat{f}(\boldsymbol{d}^{(i)}) = s] \cdot \Pr[g(s) = t] \tag{4.37}$$

$$= \exp(\varepsilon v_i) \Pr[g \circ \hat{f}(\boldsymbol{d}^{(i)}) = t] \ , \tag{4.38}$$

thus concluding the proof. □

4.5.1 Calibrating the threshold

Similar to Chapter 3.4, we have considered a threshold version of the similarity computation. The output of the computation of the *threshold similarity* consists of only one bit of information whose value is 1 if the similarity computed between two profiles is above a predefined threshold τ, and 0 otherwise. As the value of the similarity function will be modified by HDP, the threshold τ considered should be also updated to take into account this modification. Thereafter, we describe an heuristic procedure for this purpose.

Consider two different profiles $\boldsymbol{x}$ and $\boldsymbol{y}$, and fix the profile $\boldsymbol{x}$ and its privacy vector $\boldsymbol{v}^{\boldsymbol{x}}$. From the viewpoint of $\boldsymbol{x}$, we have that for any possible profile $\boldsymbol{y}$, if $\boldsymbol{v}^{\boldsymbol{x}} = \mathbf{1} = (1, \cdots, 1)$ for all items of the profile (the homogeneous case), then the

value of the heterogeneous inner product is at most $\|\boldsymbol{x}\|_1$. Otherwise, the value of the heterogeneous inner product is at most $\|\widehat{\boldsymbol{v}^{\boldsymbol{x}}}\|_1$, such that $\widehat{\boldsymbol{v}^{\boldsymbol{x}}}$ is defined similarly to $\boldsymbol{v}^{\boldsymbol{x}}$, with the exception that $\widehat{v_i^{\boldsymbol{x}}} = 0$ if the item $i \notin \boldsymbol{x}$. In this situation, it is natural for the peer whose profile is $\boldsymbol{x}$ to compensate for the difference induced by the modified inner product. For instance, if originally the threshold for the inner product was τ, then this peer should consider to use a different threshold τ', which is a scaled version of τ. We suggest to use the scale $\|\widehat{\boldsymbol{v}^{\boldsymbol{x}}}\|_1/\|\boldsymbol{x}\|_1$, which results in the relation $\tau'/\|\widehat{\boldsymbol{v}^{\boldsymbol{x}}}\|_1 = \tau/\|\boldsymbol{x}\|_1$. The new threshold can be computed locally by a peer but is also likely to be different for each peer, which results in an asymmetric computation of the heterogeneous cosine similarity. As a consequence, the cryptographic protocol comparing the output of the cosine similarity computation should have two outputs, one given to the first peer and the other being sent to the second peer. This is to be contrasted with the homogeneous case in which the threshold is supposed to be the same for each peer, thus resulting in the same output for both peers involved in the similarity computation.

4.5.2 Experimental evaluation

For the experiments, we assume that in real life, peers will assign different privacy weights to the items in their profiles. In order to simulate this, we generate privacy weights uniformly at random from a set of n equally-spaced values in a fixed range $[\underline{u}, \overline{u}]$. More formally, each item is associated with a privacy weight sampled uniformly at random from the set $\{\underline{u}, \overline{u} + \delta, \ldots, \overline{u} - \delta, \overline{u}\}$, $\delta = (\overline{u} - \underline{u})/(n-1)$, for $0 \leqslant \underline{u} < \overline{u} \leqslant 1$. For instance, if $\underline{u} = 0.5$, $\overline{u} = 1$ and $n = 3$, then the weights assigned to items will be uniformly chosen from the set $\{0.5, 0.75, 1\}$.

We simulate the GOSSPLE system described in Section 2.2 on the three datasets described in Section 2.4 namely Delicious, digg, and survey.

In Figure 4.1, we have plotted the three cases for which the interval $(\underline{u}, \overline{u})$ is set to be $(0,1), (0.5,1)$, and $(0.9,1)$. The x-axis represents $\underline{u}$, while the y-axis is the recall averaged over all slices (from $n = 1$ to $n = 10$) for the experiment in the range $[x, 1]$. Afterwards in Figure 4.2, we have fixed the range $\underline{u} \in \{0, 0.5, 0.9\}$ and $\overline{u} = 1$ and plot the average recall over all peers over all runs versus n, the number of slices (ranging from 1 to 10). In both figures, the error bars represent the variance.

From Figure 4.1 (Delicious), we can observe that there is not much difference in terms of utility between the situations in which $\underline{u} = 0.5$ and $\underline{u} = 0.9$, as both situations are close to the utility obtained with the baseline algorithm. Indeed, the largest difference is obtained when $\underline{u}$ is set to 0, in which case the utility gets closer to the utility obtained through a random clustering. Furthermore, Figure 4.2 (Delicious) demonstrates that varying the number of slices has almost no effect on the utility achieved by $\underline{u} \in \{0.5, 0.9\}$, but has a significant impact on the situation in which $\underline{u} = 0$, for which the utility decreases as the number of slices increases. One possible interpretation is that as the number of slices n increases, there are more and more items whose privacy weight differs from 1. In a nutshell, this suggests that items with high privacy weights (above 0.5) have an important impact on the utility. Combining this observation with the fact that, when $\underline{u} \geqslant 0.5$ the utility was not affected, shows that items with low privacy weights (less than 0.5) can harm the

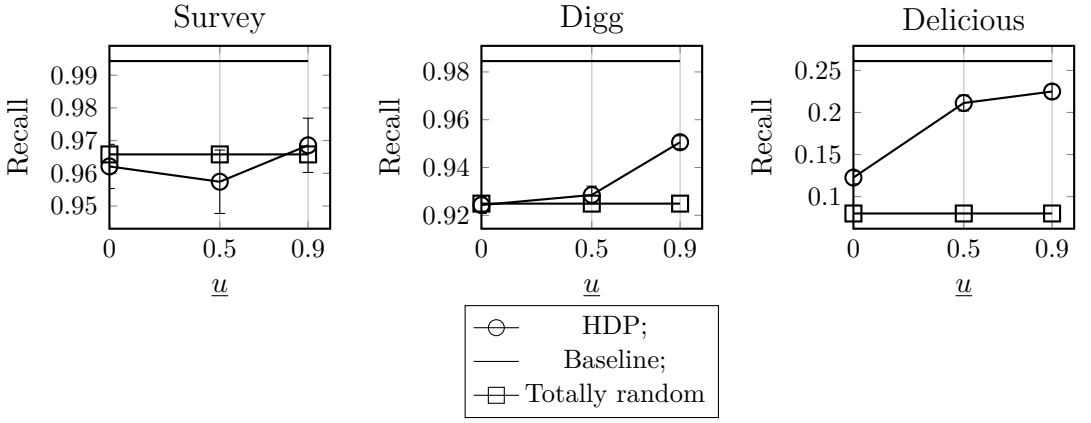


Figure 4.1 – The value reported is the average recall obtained when all peers have the same distribution over privacy weights for all items, averaged over the number of slices. *Baseline* refers to the recall obtained when the system run with no privacy guarantees using the plain version of the clustering algorithm, while *Random* refers to a random clustering process in which peers choose their neighbors totally at random.

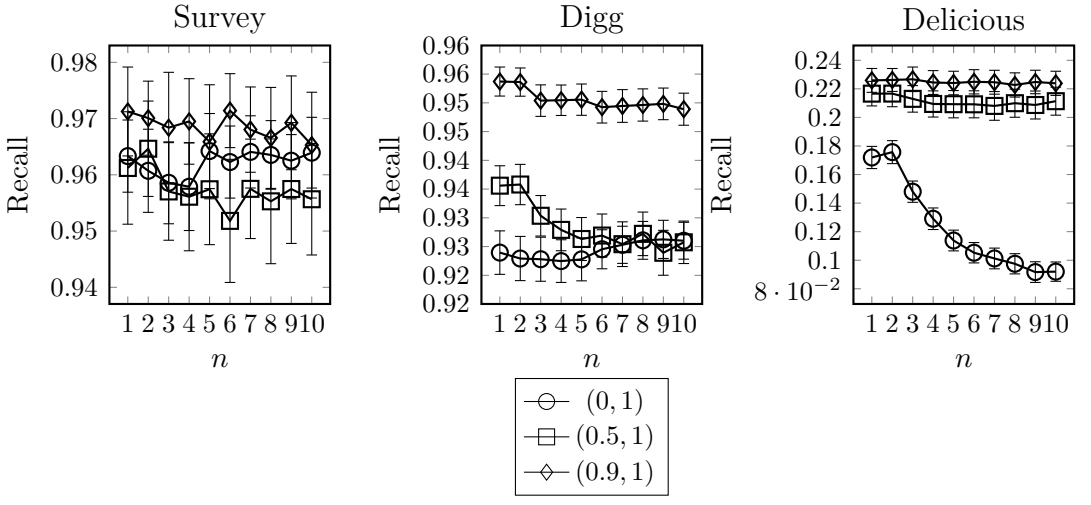


Figure 4.2 – The value reported is the average recall obtained when all peers have the same distribution over privacy weights for all items, plotted against the number of slices.

utility in a non-negligible manner. While this may seem strange at first glance, it actually solves an apparent contradiction when taken from a different point of view. Consider for instance that a privacy weight is changed by 0.1 (*e.g.*, from 0.9 to 0.8 or from 0.1 to 0.2). The impact on utility resulting from this change depends not only on the value of the difference, but also on the original value being changed. On one hand, the utility gain resulting from modifying the privacy weight from 0.1 to 0.2 is less than the utility loss if the privacy weight were modified from 0.9 to 0.8. On the other hand, changing $\underline{u}$ from 0.5 to 0 can cause a significant damage to utility because the average privacy weight drops from 0.75 to 0.5.

4.5.3 Varying privacy attitudes among users

The results of the previous section were obtained for the setting in which all peers draw their privacy weights from the same distribution (*i.e.*, all users have the same privacy attitude). However, according to a recent survey [106], users of information systems can be classified in at least three very different groups called the *Westin categories* [107]. These three groups are: PRIVACY FUNDAMENTALISTS, PRIVACY PRAGMATISTS and PRIVACY UNCONCERNED. The first group is composed of the users concerned about their privacy, while on the contrary the third group is composed of the ones that are the least concerned (according to a particular definition of concern detailed in the cited poll), while finally the second group is anything in between. For the following experiments, we have adopted the spirit of this classification and consider the three groups of users defined thereafter.

Each group is equipped with a different distribution from which they pick their privacy weights as follows.

1. The UNCONCERNED group corresponds to users that do not really care about their privacy and thus all their items have a privacy weight of 1.

2. The PRAGMATISTS group represent users that care a little bit about their privacy, such that all their items have a privacy weight chosen uniformly at random among $\{0.5, 0.75, 1\}$.

3. The FUNDAMENTALISTS group embodies users that really care a lot about their privacy and whose items have a privacy weight chosen uniformly at random among $\{0, 0.5, 1\}$.

The main issue we want to investigate is how the presence of a relatively conservative group (*i.e.*, having relatively high privacy attitudes) affect the utility of other groups. More specifically, we want to measure whether or not the presence of a group of peers with high privacy attitudes indirectly *punish* (*i.e.*, reduce the utility) of other more open groups.

During the experimentations, we have tried different proportions of these groups for a total number of users of 500. Each value plotted in Figure 4.3, has been averaged over 10 runs but the partition in groups is fixed for a given set of runs. All experiments are averaged on $\varepsilon \in \{0.1, 0.5, 1, 2, 3\}$. According to a 2004 poll [106], the percentage of each of the privacy groups FUNDAMENTALISTS, PRAGMATISTS and UNCONCERNED are respectively, 34%, 43% and 23%. Nonetheless, we also

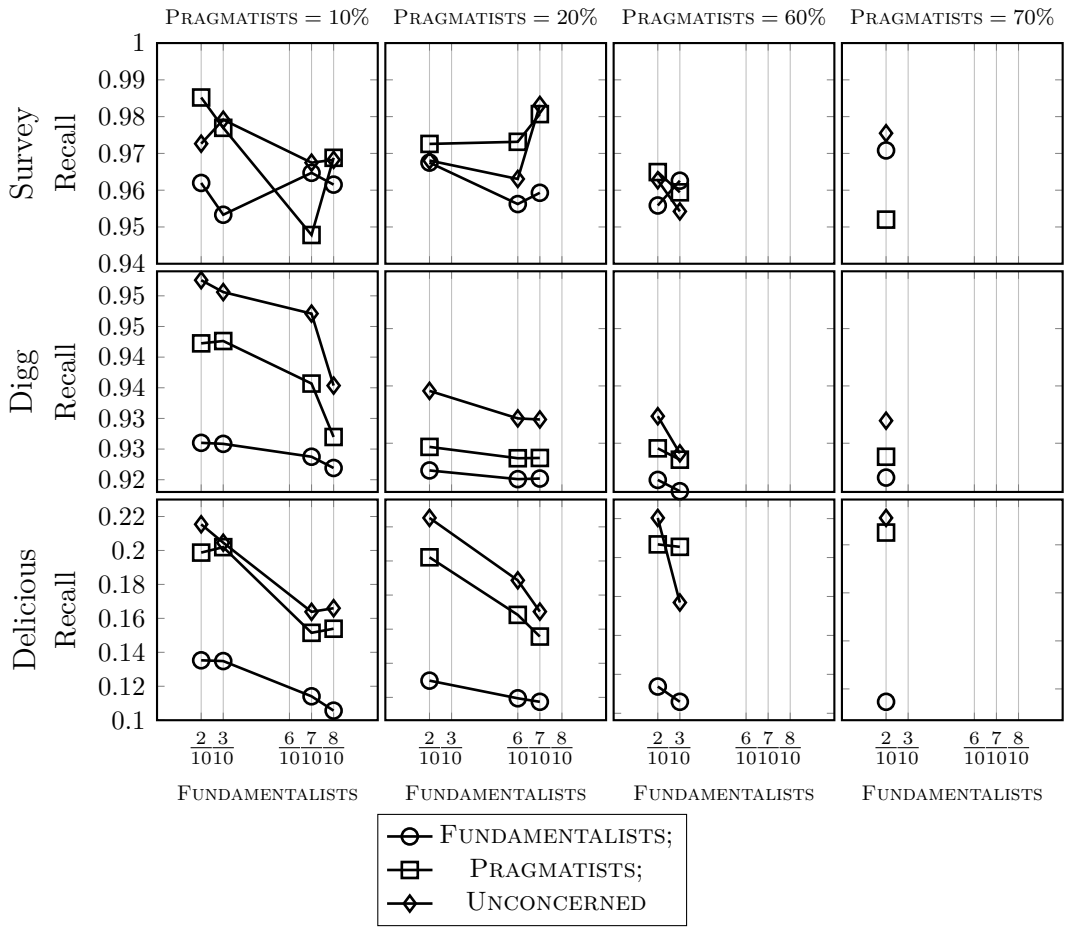


Figure 4.3 – Results obtained for the Delicious, Digg and survey datasets. The heterogeneous differential privacy has been computed for 3 groups with different privacy attitudes. For a particular figure and a particular x tick, the percentage of Unconcerned group is fully determined as (1− Pragmatists − x).

experiment a combination of several other distributions in order to investigate other possible settings. In particular, we have also tried the following percentages for each group: the proportion of the Unconcerned group and Pragmatists group vary in the following range $\{10\%, 20\%, 60\%, 70\%\}$, while the Fundamentalists group is assigned to the remaining percentage (*i.e.*, there is only two degrees of freedom). If Unconcerned group + Pragmatists group > 100%, then this combination is discarded. In Figure 4.3, the x-axis represents the percentage of the Fundamentalists group, while the y-axis corresponds to the recall. Each of the three lines correspond to the recall of one of the three groups (Fundamentalists, Pragmatists, and Unconcerned). For each of the four plots, the proportion of the Pragmatists group is denoted in the plot by the expression *Pragmatists = some value*. The proportion of the remaining group (Unconcerned) can be directly inferred by subtracting the proportions of the two other groups from 100%.

From the results obtained, we can conclude that (1) Pragmatists and Unconcerned always have better recall than Fundamentalists and (2) Unconcerned often have a better recall than Pragmatists, though not always. This seems to

indicate that the group caring more about privacy usually is punished more (*i.e.*, its utility is lower) than groups that are more liberal with respect to privacy expectations. This not really surprising as a low privacy weight will result in users from the FUNDAMENTALISTS group segregating themselves from other users in the clustering to the point that they will not necessarily have meaningful neighbors in their view. Finally, to the question whether (or not) more liberal groups will be punished by conservative groups, the answer seems to be negative. Indeed it can be seen from the results of the experiments, that conservative groups are punished more than liberal groups. For instance, the utility of liberal groups only decreases from 0.22 to 0.19 as the percentage of conservative groups increases from 20% to 80%.

4.6 Conclusion

In this chapter, we have introduced the novel concept of *heterogeneous differential privacy* that can accommodate for different privacy expectations not only per user but also per item as opposed to previous models that implicitly assume uniform privacy requirements. We have also described a generic mechanism achieving HDP called the *Stretching Mechanism*, which protects at the same time the items of the profile of user and the privacy vector representing his privacy expectations across items of the profile. We applied this mechanism for the computation of the cosine similarity and evaluate its impact on GOSSPLE by using the recall as a measure of utility. Moreover, we have conducted an experimental evaluation of the impact of having different groups of users with different privacy requirements.

5

The Non-interactive Case

Abstract

In this chapter, we describe a non-interactive protocol for private peer-to-peer similarity computation. The need for a non-interactive protocol arises from the privacy budget issue, which enforces an upper bound on the number times the similarity may be computed (*cf.* Section 2.3.1). Non-interactive protocols are not subject to this bound because they are computed only once. Moreover, a non-interactive protocol avoids the need to use cryptographic tools, allowing for more efficient execution and small communication cost; as its output may be cached. Another advantage of this non-interactive mechanism is that similarity computation may take place even when the user is offline, which is impossible to achieve with interactive mechanisms.

We introduce a novel privacy mechanism called BLIP (for BLoom-and-flIP) for this purpose. In brief, the profile of a user will be represented in a compact way, as a Bloom filter (*cf.* Section 5.2.1) that will be perturbed via the flipping of some bits. The main objective is to privately estimate the similarity between two users using this perturbed Bloom filter representation.

An analysis of the protection offered by BLIP is provided with the objective of deriving an upper and lower bound for the value of the differential privacy parameter ε, for which it is difficult to grasp an intuition for. More specifically, we define a probabilistic inference attack, called the "Profile Reconstruction Attack", that can be used to reconstruct the profile of an individual from his perturbed Bloom filter representation, along with the "Profile Distinguishing Game" which measures whether an adversary can distinguish a change in one item. An upper bound for ε is a value that makes one of these attacks succeed. The lower bound is both given by a theoretical bound on the deviation from the true answer, and empirically by finding the values of ε which provide a poor utility for a particular application.

This chapter is an expanded version of M. Alaggan, S. Gambs, and A.M. Kermarrec, "Blip: Non-Interactive Differentially-Private Similarity Computation on Bloom Filters," in *Proceedings of the 14th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'12)*, ser. Lecture Notes in Computer Science, A. W. Richa and C. Scheideler, Eds., vol. 7596. Toronto, Canada: Springer, October 1–4, 2012, pp. 202–216.

5.1 Introduction

One of the limits of interactive differential privacy is that each time a differentially private mechanism is computed, new information is released, thus incurring privacy costs. Therefore, if this computation takes place too many times, the adversary may be able to reconstruct almost all of the user's profile. For example, if the additive

noise for each computation is $o(n)$[1], n being the number of items in the system, then if an exponential number of computations takes place an adversary can reconstruct the profile [30, Theorem 2]. Moreover, the Laplacian mechanism [38] can only be computed a sublinear number of times in n [108]. Therefore, an upper bound on the number of computations taking place needs to be set. If the mechanism provides ε-differential privacy and the total privacy budget (*cf.* Section 2.3.1) is $r\varepsilon$, then the number of computations of this mechanism cannot exceed r computations due to the composition lemma (Lemma 2.1 shown in [42, 43]).

However, in a large scale system with potentially millions of users as the one we consider, setting a bound on the maximum number of similarity computations that can occur is typically not applicable for iterative gossip-based protocols (*cf.* Section 2.2). In particular, since new peers keep joining the system continuously, an interactive mechanism would be of limited applicability without extra assumptions (*cf.* Section 3.3.1).

To simultaneously address the privacy and scalability issues, we propose BLIP (for BLoom-then-flIP), a non-interactive differentially private mechanism, which computes a standard *Bloom filter* [109] from the profile of a peer, and then perturbs it prior to its public release in order to ensure high privacy guarantees. This randomized Bloom filter can be used an unbounded number of times to compute the similarity of this peer with other profile without breaching the privacy of the profiles. Moreover, this approach has exactly the same communication cost as *plain* (*i.e*, non-perturbed) Bloom filters, while offering much higher privacy guarantees, but at the cost of a slight decrease of utility.

In differential privacy, the trade-off between utility and privacy can be set through the privacy parameter ε. However, being able to choose an appropriate value for this parameter is still an open research question, which has not really been investigated, with a few exceptions [39, 28]. However, the Dwork-Naor impossibility proof [27] may suggest one way of choosing ε. Basically, they assume that the privacy mechanism $\mathcal{M}$ is non-trivial in the sense that an adversary observing $\mathcal{M}(x)$, for a private profile x, but without *any* auxiliary information, cannot compute a privacy breach about x, even if he has a prior distribution $\mathcal{D}$ on x. In this chapter, we set ε to the value that (partly) realizes this assumption. We do this by proposing two attacks where both try to create a different privacy breach[2], and neither of them assumes auxiliary information. Moreover, to keep things simple, we do not assume correlations exist between items (correlations are handled in Chapter 6). Therefore the value for ε for which our attacks succeed is merely an *indicative* upper bound, which is still considered progress given that we may have very little clue otherwise. The attacks are probabilistic inference attacks. The first attack is called the "Profile Reconstruction Attack" as it can be used to reconstruct the profile of a peer from its perturbed Bloom filter representation. The other attack is called the "Profile Dis-

[1]For comparison, the Laplacian mechanism [38] used in the previous chapters corresponds to $O(1) = o(n)$ additive noise, while the BLIP mechanism of this chapter incurs $O(\sqrt{n}) = o(n)$ additive noise with constant probability.

[2]For this, it is necessary to assume what constitutes a privacy breach, but in the context of Dwork-Naor impossibility result, it must be a piece of information about the database instance at hand. For the profile distinguishing game, this piece of information is whether the profile has the i^{th} bit set to one or not. For the other attack, it is an approximation of the profile.

tinguishing Game", in which the adversary tries to distinguish two profiles, differing only by one item, by observing the perturbed Bloom filters of both profiles. More specifically, we provide an analysis of the protection and the utility offered by BLIP against these attacks by deriving upper and lower bounds for the required value of the differential privacy parameter ε.

In short, the lower bound gives theoretical guarantees on the resulting approximation error generated by a specific value of the privacy parameter, while the upper bound demonstrates experimentally that the privacy parameter must be below a certain threshold to be able to prevent these attacks. Furthermore, we evaluate experimentally the trade-off between privacy and utility that can be reached with BLIP on GOSSPLE (*cf.* Section 2.2) in which peers are grouped based on the similarity between their profiles.

The chapter is organized as follows. First, in Section 5.2, we present background on Bloom filters and differential privacy necessary to understand our work. Then, we give an overview of the related work in Section 5.3. Afterwards, we propose BLIP, a non-interactive differentially private mechanism for randomizing Bloom filters in Section 5.4 and analyze in details the privacy guarantees it provides. In Section 5.5, we evaluate the impact of this mechanism on utility, as measured in terms of recall on GOSSPLE. In Section 5.6, we describe novel inference attacks, called the profile reconstruction attack, that can reconstruct a profile from its Bloom filter representation, and another one called the profile distinguish game, and show how BLIP can be used to drastically reduce its impact. Finally, we conclude in Section 5.8.

5.2 System model and background

In this chapter, we consider a computationally-unbounded adversary that can observe the flipped Bloom filter (the differentially-private version of the user's profile), but not the internal state of a peer owning that Bloom filter. This is unlike previous chapters where the adversary had to be computationally-bounded because of the use of cryptographic constructions.

5.2.1 Bloom filters

A Bloom filter [109] is a probabilistic data structure composed of a vector $\mathcal{B} \in \{0,1\}^m$. A set $\mathcal{H}$ of k independent hash functions is used to specify its operation. Each hash function maps an item to a uniform and independent random index $\{1,\ldots,m\}$ in the Bloom filter. Bloom filters form a compact representation of sets as they can represent any set with no more than m bits, for a given trade-off between m, the size of the structure, and the false positive probability (*i.e.*, the probability of an item is believed to belong to the Bloom filter while it is not). Bloom filters are often used for applications in which the storage space is limited or for protocols for which the communication cost has to be low but some false positives can be tolerable.

Bloom filters are associated with two operations: the `add` operation inserts an item into the Bloom filter while the `query` operation tests if an item is already present in it (some types of Bloom filters also support the removal of items [110] but we do

not consider them in this thesis). Both operations start by applying the k hash functions from $\mathcal{H}$ to the item in question in order to obtain a subset of $\{1, \ldots, m\}$ of indexes into $\mathcal{B}$. With an abuse of notation, we denote this step as $\mathcal{H}(\text{item})$, that is

$$\mathcal{H}(\text{item}) = \{i \mid i = h(\text{item}), h \in \mathcal{H}\} \ . \tag{5.1}$$

The add operation inserts the item by setting the bits corresponding to those indexes to one in $\mathcal{B}$, and does this independently of the previous values of these bits. For instance, the post condition to add(item) when applied to a Bloom filter $\mathcal{B}$ is that $\mathcal{B}_i = 1$ for every $i \in \mathcal{H}(\text{item})$, whereas all the other bits are left unchanged. The query operation checks if an item is included in the Bloom filter by verifying whether or not all the bits whose index is in $\mathcal{H}(\text{item})$ are set to one. If the item is included in the Bloom filter, query returns 1, while it returns 0 otherwise. For the rest of this chapter, we use the notation $i \in \mathcal{B}$ to express the statement $\mathsf{query}(i, \mathcal{B}) = 1$, or equivalently that i is a member of $\mathcal{B}$.

For example, consider a Bloom filter with a length of $m = 4$ bits, equipped with $k = 2$ hash functions $\mathcal{H}$. Fix for a particular item, say i_1, to be mapped by the hash functions to $\{2, 4\}$ (*i.e.* $\mathcal{H}(i_1) = \{2, 4\}$). Given an empty Bloom filter represented as $\mathcal{B} = (0, 0, 0, 0)$, the item i_1 is then inserted in the Bloom filter by setting the 2^{nd} and 4^{th} bits to one, ending up with $(0, 1, 0, 1)$. Verifying that the same item i_1 is present in a Bloom filter simply requires to check whether the 2^{nd} and 4^{th} bits are equal to one. As a different example, consider the scenario in which starting from an empty Bloom filter, two items are inserted, respectively to positions $\{1, 4\}$ and $\{1, 2\}$, which results in the corresponding Bloom filter $(1, 1, 0, 1)$. In this situation, the operation $\mathsf{query}(i_1, (1, 1, 0, 1))$ will lead to believe that i_1 was present in the Bloom filter although it was never explicitly inserted, thus causing a false positive.

The probability of such false positive is a function of m, k, and the number of items inserted in the Bloom filter. In general, the values of m and k are chosen according to a trade-off decision between the space usage and the false positive rate (an upper and lower bound for this trade-off is provided in [111]). The number of hash functions k does not imply a huge computational cost, given that it is possible to efficiently generate k hash values using only two hash operation [112].

5.2.2 Non-interactive differential privacy

Both the Laplacian mechanism (Theorem 2.1) and the *exponential mechanism*, proposed by McSherry and Talwar [113], which provides differential privacy for functions whose output is structured (*e.g.*, graphs or trees), are instances of *interactive* mechanisms. In particular, they both need to know the query before releasing the output. More specifically, they require interaction between the data holder and the user performing the query, hence the name.

On the other hand, a *non-interactive* mechanism releases a *sketch* of the data, which could be used later by anyone to compute a wide array of queries about the data (the queries belong to some class of queries) from that pre-computed sketch without requiring any interaction or access to the original data. This setting can be seen as a one-way communication protocol.

In a non-interactive setting, the answer to a query is not computed by the non-interactive mechanism itself. Rather, the answer can be computed from the output released by the mechanism. Thus, after publishing his output, the data owner may go offline.

Examples of non-interactive mechanisms for differential privacy include [26, 114, 45], which we mention in the related work section (*cf.* Section 5.3). Our proposition of differentially-private Bloom filters is based on the work of Beimel, Nissim and Omri [26] in which the authors address the "binary sum" problem, which deals with privately computing the number of ones in a bit vector. More precisely, we adapt their non-interactive mechanism to Bloom filters, with the goal of computing inner product between two Bloom filters. We review this related work in Section 5.3.

5.3 Related work

5.3.1 Non-interactive differential privacy

Most of the previous works studying non-interactive mechanisms in the context of differential privacy [115, 114, 45] have considered mechanisms that release a synthetic database that can be used to perform queries instead of the original one. The first work [115] is very inefficient due to its high computational complexity while [114] considers only input databases that have a sparse representation. However, the latter [114], depends on the exponential mechanism, which is inefficient, to choose its parameters.The Johnson-Lindenstrauss transform of [45] provides a relaxed variant of differential privacy, called (ε, δ)-differential privacy, which is strictly weaker than ε-differential privacy [69].

5.3.2 Privacy and Bloom filters

The literature on using Bloom filters for designing privacy-preserving techniques is quite diverse but often assumed a kind of client-server model, in which the owner of the Bloom filter (server) wants to answer some query asked by the client. In this context, some solutions focus of concealing the content of the query from the server, thus ensuring client's privacy (similarly to private information retrieval schemes), while others try to prevent the client from getting more information than the answer to its query, thus ensuring server's privacy (in the spirit of oblivious transfer). The application domains of these techniques include searching document indexes [116, 117, 118], private information retrieval [119], private matching [120], private publication of search logs [121] and anti-counterfeiting in supply chains [122]. The closest to our work is [121], which provides a probabilistic version of differential privacy (*i.e.*, the privacy guarantee holds for all except a small fraction of items). However, this technique only works for multi-sets and it is not straightforward it to apply on normal sets such as user profiles.

Both [116, 123] provide semantic security (indistinguishability) for the query sent by a peer, while [118, 119] require to salt the query with spurious bits (or hashes) with the approach developed in [119] also aims at setting a high number of possible

pre-images of the query. Some authors have considered the setting of an honest-but-curious third party that relies on cryptographic techniques [120], while other works propose a non-interactive solutions such as the release of a perturbed counting Bloom filter[3] [121] or non-interactive zero-knowledge proofs [122]. Moreover, these last two works provide information-theoretical security for the privacy of the client as we do with BLIP.

With respect to the privacy of the server (*i.e.*, protecting the content of the Bloom filter), some previous work [116] relies on the use of trapdoor function to limit the number of items that the client can query. Other techniques use a zero-knowledge proof issued by the server [122] or the obligation to have the approval of a trusted censor server [118] in order to limit the quantity of information (*i.e.*, queries) that the client can learn. Some approaches protect at the same time the client and the server privacy through the use of semi-trusted third party combined with cryptographic techniques [120] or provide simulation guarantees against malicious clients [123].

Some non-interactive mechanisms based on Bloom filters have also been proposed. For instance, [117] provides "probable innocence" (the probability that an adversary's claim that the server has an item i is true is in the range $(0, 0.5]$) while [121], that provides a probabilistic version of differential privacy (*i.e.*, the privacy guarantee holds for all except a small fraction of case), is the closest to our work. However, this technique only works for multi-sets as the release mechanism removes items whose count (after adding noise) is less than a predefined threshold. In our context, there is no notion of counting items, which make their work incomparable to ours. Moreover, we aim at achieving full differential privacy (*i.e*, in the sense of information-theoretic security) while [121] provides only probabilistic differential privacy (a strictly weaker guarantee).

5.4 The bit-flipping mechanism

We propose a novel approach relying on bit flipping to achieve differential privacy. The intuition behind our proposed mechanism is simple: before releasing a Bloom filter, each bit is flipped with a given probability so as to ensure differential privacy on the *items* of the profile from which the Bloom filter is derived.

If the profile is composed of n bits, in which n is the number of items in the system, then if n changes over time, it will be possible to breach the privacy of users just be observing the size of their flipped profile. For example, it would be possible to exclude certain items from being present in the user's profile just by learning that their index is greater than n. Bloom filters were chosen because the size of the flipped Bloom filter does not depend on n, allowing n to vary over time if the system supports it. Moreover, Bloom filters will be small even if n is huge, allowing less communication overhead. A challenge arises, however, because the mapping between bits and items is not one-to-one and is defined via hash functions, rendering it difficult to understand how flipping those bits relates to the privacy of

[3] A counting Bloom filter is an array of m buckets, in which each bucket can count from 0 up to some integer z. Buckets play a role similar to the bits in a "plain" Bloom filter, with the exception that instead to set the bit to one, the bucket value is increased by one.

the items.

In this section, we provide a formal definition of our mechanism and compute the flipping probability that has the least impact on utility while preserving differential privacy. A particular challenging task is to derive how the flipping of bits influences the privacy of the items themselves, which we address in Section 5.4.2.

5.4.1 Bloom-then-flip

More formally, the proposed non-interactive mechanism is a randomized function that for each bit of a Bloom filter, tosses a biased coin and based on the result, either outputs the original value of the bit or its opposite (*i.e.*, flip it). Since the mechanism *flips* the Bloom filter representation of a profile, we call it BLIP (for *Bloom-then-flip*). For example, the mechanism may take as input the Bloom filter $(0, 1, 1, 0)$ and randomly (with probability $0 < p < 1/2$ to be defined in Theorem 5.2) decide to flip the first two bits, thus outputting $(1, 0, 1, 0)$.

Let the function $b : \{0,1\}^n \to \{0,1\}^m$ be the function that maps a user's profile to its corresponding Bloom filter representation. That is, if $\mathcal{B} = b(\boldsymbol{d})$, then the bit $\mathcal{B}_i$ will be set to 1 if and only if i belongs to $\mathcal{H}(j)$ for some index j such that $\boldsymbol{d}_j = 1$, in which $\mathcal{H}$ is the set of hash functions used by b. Then, BLIP can be described as a $\mathcal{M}(\boldsymbol{d}) = F(b(\boldsymbol{d}))$, or alternative $\mathcal{M} = F \circ b$, in which $F : \{0,1\}^m \to \{0,1\}^m$ is a randomized function that flips each bit of its input with flipping probability p. More precisely, each bit output is the opposite as the corresponding bit of the input with probability p (otherwise the bit remains same). BLIP consists of (1) generating the Bloom filter representation of the profile first, and then (2) flipping the resulting Bloom filter (which is a binary vector). The flipping process may introduce permanent artifacts but the additive error is bounded by $O(\sqrt{m})$ with constant probability, in which m is the number of bits of the Bloom filter, as shown in Section 5.4.4.

The idea of randomizing each bit independently (as opposed to perturbing the final answer itself) is also known as the *randomized response* [124], and precedes the notion of differential privacy. The use of randomized response for differential privacy was previously studied [26] but the definition of differential privacy adopted ([26, Definition 2.4]) slightly differs from ours. In their model, each individual bit belongs to and is held by a different peer, which is very close to the setting of secure multiparty computation, as opposed to our model (of this chapter) where the input belongs to one and only one peer.

The following is the theorem that randomized response can ensure differential privacy.

Theorem 5.1 (Differential privacy (randomized response))**.** *Consider the randomized function $f : \{0,1\} \to \{0,1\}$ that is ε-differentially private. Thus, for all values $y, y' \in \{0,1\}$, and for all output $t \in \{0,1\}$:*

$$\Pr[f(y) = t] \leqslant \exp(\varepsilon) \Pr[f(y') = t] \ , \tag{5.2}$$

in which the probability is taken over the randomness of f, and exp *refers to the exponential function. Then, the function $F(x_1, \ldots, x_m) = (f(x_1), \ldots, f(x_m))$, also denoted as $F = \bigotimes_{i=1}^{m} f$, is ε-differentially private.*

Proof. Let $\boldsymbol{x} = \{0,1\}^m$, and let $(\boldsymbol{x}_{-i}, y)$ denotes the vector resulting from replacing the i-th coordinate of $\boldsymbol{x}$ with y, which means: $(\boldsymbol{x}_{-i}, y) = (x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_m)$. $\boldsymbol{x}$ and $(\boldsymbol{x}_{-i}, y)$ are neighbors (*i.e.* $\boldsymbol{x} \sim (\boldsymbol{x}_{-i}, y)$). Then, for any two neighboring vectors $\boldsymbol{x} \sim (\boldsymbol{x}_{-i}, y)$:

$$\frac{\Pr[F(\boldsymbol{x}) = \boldsymbol{t}]}{\Pr[F(\boldsymbol{x}_{-i}, y) = \boldsymbol{t}]} = \frac{\Pr[f(x_i) = t_i] \cancel{\prod_{j \neq i} \Pr[f(x_j) = t_j]}}{\Pr[f(y) = t_i] \cancel{\prod_{j \neq i} \Pr[f(x_j) = t_j]}} \leqslant \exp(\varepsilon) \ , \tag{5.3}$$

for all $\boldsymbol{t} = (t_1, \ldots, t_n) \in \{0,1\}^m$. In the above formula, the equality is obtained by independence. □

In the following section, we will analyze the function f in isolation instead of F.

5.4.2 The flipping probability

The local perturbation function f of Theorem 5.1 takes a bit x and outputs $1-x$ with probability $p < 1/2$ and x otherwise. If f inverts the value of a bit with probability $p = 1/2$, this provides the best privacy guarantees as each flipped bit conveys zero information, but this also destroys the utility. Therefore, the main challenge is to find the optimal probability p that f should use in order to maximize utility as a function of the privacy parameter ε. With respect to the utility, the smaller p is, the more accurate the output and therefore the best utility can be obtained by minimizing p.

Remember that BLIP mechanism $\mathcal{M} = F \circ b$, in which b is the function that encodes the user's profile as a Bloom filter, and $F = \bigotimes_{i=1}^{m} f$. Analyzing the differential privacy guarantees with respect to the input of $\mathcal{M}$ must take into account both F and b. More precisely, only analyzing F or equivalently f, without taking b into account, is not sufficient to prove that $\mathcal{M}$ provides differential privacy for the individual *items* of the profile that are encoded in the Bloom filter. For instance, recall that an item can impact up to k different bits due to the use of k different hash functions. In particular, differential privacy is guaranteed *for individual items* by the randomized mechanism $\mathcal{M}$ if for each pair of neighboring profiles $\boldsymbol{d} \sim \boldsymbol{d}'$ and for all bit strings $\boldsymbol{t} \in \{0,1\}^m$, the following condition holds (which is equivalent to the differential privacy condition in Definition 2.2):

$$\left| \ln \frac{\Pr(F(\mathcal{B}) = \boldsymbol{t})}{\Pr(F(\mathcal{B}') = \boldsymbol{t})} \right| \leqslant \varepsilon \ , \tag{5.4}$$

in which $\mathcal{B}$ is the Bloom filter of $\boldsymbol{d}$ and $\mathcal{B}'$ the Bloom filter of $\boldsymbol{d}'$.

Theorem 5.2 (Privacy guarantees for items)**.** *Setting the bit-flipping probability p to $1/(1+\exp(\varepsilon/k))$ satisfies condition* (5.4) *and thus provides ε-differential privacy for items. Thus, flipping the bits of a Bloom filter with this probability guarantees ε-differential privacy for the items encoded in this Bloom filter.*

Proof. Given a Bloom filter $\mathcal{B}$ equipped with a set $\mathcal{H}$ of hash functions, and an item i, let $T = \mathcal{H}(i)$ be the set of indexes whose corresponding bits in $\mathcal{B}$ are equal to one if i is in $\mathcal{B}$, and let $k' = |T| \leqslant k = |\mathcal{H}|$ be the number of those indexes.

We partition the Bloom filter $\mathcal{B}$ into two vectors: $\mathcal{B}^T = \bigotimes_{i\in T} \mathcal{B}_i \in \{0,1\}^{|T|}$ and $\mathcal{B}^{-T} = \bigotimes_{i\in\{1,\ldots,m\}\setminus T} \mathcal{B}_i \in \{0,1\}^{m-|T|}$, in which the former is the restriction to the bits whose indexes are in T, and the latter is the restriction to all other bits. A partition of any vector $\boldsymbol{t} \in \{0,1\}^m$ is defined similarity: $\boldsymbol{t}^T$ and $\boldsymbol{t}^{-T}$. We denote by $q = 1 - p$, the probability of not flipping a bit (it is always the case that $q > p$).

As the profiles (and therefore the Bloom filters as well) are universally quantified, hence they can be treated as constants and not random variables. As a consequence, they do not affect the independence property of $F = \bigotimes f$. Notice that if $\boldsymbol{d} \sim \boldsymbol{d}'$, then $\mathcal{B}^{-T} = \mathcal{B}'^{-T}$. The proof proceeds as follows. For all $\boldsymbol{t} \in \{0,1\}^m$:

$$\begin{aligned}
\left|\ln \frac{\Pr(F(\mathcal{B}) = \boldsymbol{t})}{\Pr(F(\mathcal{B}') = \boldsymbol{t})}\right| &= \left|\ln \frac{\Pr(F(\mathcal{B}^T) = \boldsymbol{t}^T)\Pr(F(\mathcal{B}^{-T}) = \boldsymbol{t}^{-T})}{\Pr(F(\mathcal{B}'^T) = \boldsymbol{t}^T)\Pr(F(\mathcal{B}'^{-T}) = \boldsymbol{t}^{-T})}\right| && \text{by independence} \\
&= \left|\ln \frac{\Pr(F(\mathcal{B}^T) = \boldsymbol{t}^T)}{\Pr(F(\mathcal{B}'^T) = \boldsymbol{t}^T)}\right| && \text{because } \mathcal{B}^{-T} = \mathcal{B}'^{-T} \\
&= \left|\ln \frac{\Pr(F(\overbrace{1,\ldots,1}^{k'}) = \boldsymbol{t}^T)}{\Pr(F(\mathcal{B}'^T) = \boldsymbol{t}^T)}\right| && \text{as } i \text{ is in } \mathcal{B} \\
&= \left|\ln \frac{p^z q^{k'-z}}{\delta}\right| = \left|\ln\left[\frac{q^{k'}}{\delta}\left(\frac{p}{q}\right)^z\right]\right| && \text{let } \delta = \Pr(F(\mathcal{B}'^T) = \boldsymbol{t}^T)\ ,
\end{aligned}$$

in which $z \in \{0,\ldots,k'\}$ is the number of zero bits in $\boldsymbol{t}^T$ ($i.e\ z = m - \|\boldsymbol{t}^T\|_1$).

If $p = 1/(1+\exp(\varepsilon/k'))$ then $q^x = (1-p)^x = \exp(\varepsilon x/k')p^x$, for $x \in (0,\infty)$, and $(p/q)^x = \exp(-\varepsilon x/k')$. Then since $p^{k'} \leqslant \delta \leqslant q^{k'}$ (recall that $\mathcal{B}'^T$ is a constant bit vector), $p^{k'} \leqslant \delta \leqslant \exp(\varepsilon)p^{k'}$ or $1 \leqslant \delta/p^{k'} \leqslant \exp(\varepsilon)$ implying that $1 \geqslant p^{k'}/\delta \geqslant \exp(-\varepsilon)$, and hence $0 \geqslant \ln(p^{k'}/\delta) \geqslant -\varepsilon$. Then

$$\left|\ln\left[\frac{q^{k'}}{\delta}\left(\frac{p}{q}\right)^z\right]\right| = \left|\ln\left[\frac{\exp(\varepsilon)p^{k'}}{\delta}\exp(-\varepsilon z/k')\right]\right| = \left|\ln(p^{k'}/\delta) + \varepsilon(k'-z)/k'\right| \ . \quad (5.5)$$

Then, we have

$$\ln(p^{k'}/\delta) + \varepsilon(k'-z)/k' \leqslant \ln(p^{k'}/p^{k'}) + \varepsilon(k'-z)/k' \leqslant \ln(1) + \varepsilon k'/k' = \varepsilon \ . \quad (5.6)$$

Afterwards, we want to show that $\ln(p^{k'}/\delta) + \varepsilon(k'-z)/k' \geqslant -\varepsilon$. Since $1 \geqslant p^{k'}/\delta \geqslant \exp(-\varepsilon)$ then

$$\ln(p^k/\delta) + \varepsilon(k-z)/k \geqslant \ln(\exp(-\varepsilon)) + \varepsilon(k-k)/k \geqslant -\varepsilon \ , \quad (5.7)$$

and hence

$$\left|\ln \frac{\Pr(F(\mathcal{B}) = \boldsymbol{t})}{\Pr(F(\mathcal{B}') = \boldsymbol{t})}\right| = \left|\ln(p^{k'}/\delta) + \varepsilon(k'-z)/k'\right| \leqslant \varepsilon \ , \quad (5.8)$$

thus proving the theorem. Moreover, equality can be obtained by setting $z = 0$, and having $\mathcal{B}'^T$ be the exact opposite of $\boldsymbol{t}^T$. The previous argument was for $p = 1/(1+\exp(\varepsilon/k')) \leqslant 1/(1+\exp(\varepsilon/k))$, which proves the theorem. □

Remark 5.1 (Optimality of p.). The case $k = 1$ reduces to protecting individual bits. In order to compute the values of p that ensures differential privacy *for a single bit*

(in accordance to Theorem 5.1), it is easy to verify that $p = \frac{1}{1+\exp(\varepsilon)}$ is the minimum value for which

$$\left| \ln \frac{\Pr(f(y) = t)}{\Pr(f(y') = t)} \right| \leqslant \varepsilon$$

holds for all t, y, y' in $\{0, 1\}$. Therefore, there exists a value for $k \geqslant 1$, in which f flips the input bit with probability p, for which $p < \frac{1}{1+\exp(\varepsilon/k)}$ does not ensure ε-differential privacy.

In the following section, we show how construct an unbiased estimator for inner product using BLIP.

5.4.3 Estimating inner product from flipped Bloom filters

Let $\mathcal{B}$ be a fixed Bloom filter before it is flipped, and $\widetilde{\mathcal{B}}$ be the *random variable* denoting the output of the BLIP mechanism applied on $\mathcal{B}$. Instead of publishing the original Bloom filter, the owner of $\mathcal{B}$ releases publicly $\widetilde{\mathcal{B}}$ instead. When a peer receives $\widetilde{\mathcal{B}}$, it can use it to *estimate* the similarity between $\mathcal{B}$ (which this peer does *not* know) and its own Bloom filter $\mathcal{B}'$. Since the computation is performed locally at the level of the peer that owns the Bloom filter $\mathcal{B}'$, this Bloom filter does not need to be perturbed. The similarity between $\mathcal{B}$ and $\mathcal{B}'$ is inferred by computing the perturbed inner product $\widetilde{SP} = \widetilde{\mathcal{B}} \cdot \mathcal{B}' = \sum_i \widetilde{\mathcal{B}}_i \mathcal{B}'_i$. Given the perturbed inner product, we introduce an unbiased estimator for the inner product and denote is as $\widehat{SP}$ in the following theorem.

Theorem 5.3 (Unbiased estimator for inner product)**.** *The function* $\widehat{SP} = \frac{\widetilde{SP} - p\|\mathcal{B}'\|_1}{1-2p}$ *is an unbiased estimator for* $SP = \mathcal{B} \cdot \mathcal{B}'$*, in which* p *is the flipping probability used by BLIP. Note that* $\|\mathcal{B}'\|_1 = \sum_i |\mathcal{B}'|$*, or equivalently in this context, the number of ones in* $\mathcal{B}'$*.*

Proof. To prove the theorem we need to show that $\mathbb{E}[\widehat{SP}] = SP$. Since $\mathbb{E}[\widehat{SP}] = (\mathbb{E}[\widetilde{SP}] - p\|\mathcal{B}'\|_1)/(1-2p)$ and $\mathbb{E}[\widetilde{SP}] = \mathbb{E}[\sum_i c_i] = \sum_i \mathbb{E}[c_i]$ in which $c_i = \widetilde{\mathcal{B}}_i \mathcal{B}'_i$ and noticing that we have

$$\mathbb{E}[c_i] = \mathcal{B}_i \mathcal{B}'_i (1-p) + (1 - \mathcal{B}_i)\mathcal{B}'_i p = \mathcal{B}'_i p + \mathcal{B}'_i \mathcal{B}_i (1-2p)$$

then substitution proves the theorem. □

5.4.4 Error bounds

In this section, we prove a concentration bound on the inner product that is estimated from the flipped Bloom filter. Let SP be the true inner product, and $\widehat{SP}$ be the random variable that represents the estimation, in which the randomness is taken over the flipping of the bits. Let $\widehat{sp}$ be the observed value of $\widehat{SP}$. We will prove that the probability that the error $\widehat{SP} - SP$ is larger than $\sqrt{m}$ is constant in m, in which m is the total number of bits (the size of the Bloom filter).

Given a random variable $\tilde{v}_i$ for a bit flipped with probability p and another constant bit b_i, an unbiased estimator $\widehat{sp_i}$ for $v_i b_i$ is $\widehat{sp_i} = (\tilde{v}_i - p)b_i/(1-2p)$. Recall that $\widehat{sp} = \sum_i \widehat{sp_i}$ is the unbiased estimator of the inner product SP from Theorem 5.3.

Theorem 5.4 (Hoeffding's Inequality [125]). *Let $X_1, \ldots, X_m$ be independent variables such that for $i \in \{1, \ldots, m\}, a \leqslant X_i \leqslant b$ for some $a < b \in \mathbb{R}$ independent of i. Let $X = \sum_i X_i$, then*

$$\Pr[|X - \mathbb{E}[X]| \geqslant t] \leqslant 2\exp\left(\frac{-2t^2}{m(b-a)^2}\right) .$$

Therefore we can prove the following theorem by setting $t = \sqrt{m}$ and substituting $a = p/(2p-1), b = (1-p)/(1-2p)$, since $p/(2p-1) \leqslant \widehat{sp_i} \leqslant (1-p)/(1-2p)$, in which $p < 1/2$.

Theorem 5.5 (BLIP error bound). *If $\widetilde{\mathcal{B}}$ is a flipped Bloom filter of m bits, in which each bit is flipped with probability $p = 1/(1+\exp(\varepsilon/k))$, and $\mathcal{B}'$ is an ordinary Bloom filter, and $\widehat{SP}$ is the unbiased estimator of $SP = \widetilde{\mathcal{B}} \cdot \mathcal{B}'$ introduced in Section 5.4.3, then*

$$\Pr\left[\left|\widehat{SP} - SP\right|\right] \leqslant \sqrt{m} \tag{5.9}$$

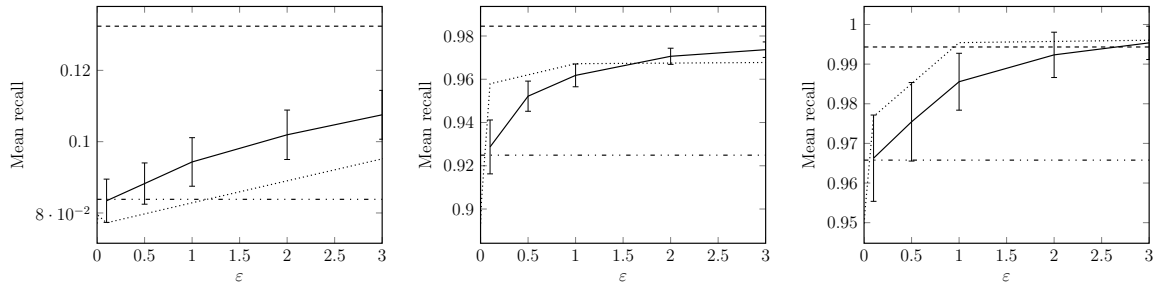
with probability at least $1 - 2\exp(-2\tanh(\varepsilon/2k)^2)$.

5.5 Utility evaluation

In this section, we evaluate experimentally how the utility is impacted by the BLIP mechanism by applying it on Gossple (described in Section 2.2). In this setting, we measure the utility in terms of the *recall*. Each profile is encoded as a Bloom filter of $m = 5000$ bits using 18 different hash functions. We also use the three datasets Delicious, Digg, and Survey, described in Section 2.4. We also compare the result to those obtained from the *noisy release* protocol from Chapter 3.

We provide in Figure 5.1 the results of the experiments for the recall, averaged over 100 runs, versus privacy parameter ε. On this plot, we use $p = 1/(1+\exp(\varepsilon/k))$ for the flipping probability ($k = 18$). The main plot displays the recall obtained when using the cosine similarity based on the bias-corrected inner product (Theorem 5.3). The other lines show the recall obtained for two different cases that act as a baseline: (1) when the similarity is computed on totally random Bloom filters (*i.e.*, Bloom filters whose bits are flipped with probability 0.5) and (2) when the similarity is computed with a plain Bloom filters that have not been flipped at all. Note that the intrinsic value of the recall that can be reached and the gap between the baseline (1) and baseline (2) are directly dependent of the characteristics of the dataset considered.

From these plots, we observe that the utility remains relatively high even for values of ε that are small, in comparison to the utility obtained with totally randomized Bloom filters (which leads to a random neighborhood for each peer). In general the utility obtained is far from the non-private solution (for Delicious the value is about 0.261) in which peers directly exchange their profiles. However, this is inherent to the fact that the similarity is computed based on the Bloom filters (and not the profiles themselves) and this is not a drawback due to our flipping mechanism. When combined with the results of the experiments described in the next section (resilience to inference attacks), it is possible to observe the trade-off



(a) Delicious (100 runs) (b) Digg (100 runs) (c) Survey (500 runs)

Differential privacy; Totally random Blooms; Plain Blooms; Laplacian;

Figure 5.1 – Recall obtained with BLIP. The bars correspond to the standard deviation. The dotted line called "Laplacian" is the *noisy release* from Chapter 3.

between privacy (*i.e.*, resilience against the profile reconstruction attack and the profile distinguish game) and utility (*i.e.*, recall).

In comparison to the interactive case (the noisy release protocol of Chapter 3) we expect the recall of the non-interactive case to be lower, since the perturbation introduced by BLIP is $\Theta(\sqrt{m})$, in which m is the number of bits in the Bloom filter, versus $\Theta(1)$ for the Laplacian mechanism used in the interactive case. The experiments confirm this for the Digg and Survey datasets, but not for Delicious, perhaps because of the trade-off between the sparsity of Delicious and the factors hidden by the asymptotic notation of the perturbation. Nonetheless, the difference, as observed, is not too big. Notice that for Survey the recall from the Laplacian mechanism for high epsilon is greater than recall achieved when Blooms are not flipped at all, which is due to the fact that the Blooms, even if not flipped, still introduce a small amount of perturbation.

5.6 Profile reconstruction attack

In this section, we try to answer some of the fundamental questions raised by the use of differential privacy such that "How to choose the value of ε?" or "What does it mean for ε to be equal to 0.5 or 1?" by considering a particular inference attack. The main objective of the adversary is to infer the description of the profile of a peer from its Bloom filter representation. We assume that in the same manner as other peers in the network, the adversary (which in fact could simply be one of these peers) can easily have access to the ε-differentially private Bloom filters released by the BLIP mechanism. We describe thereafter an inference attack, called the "profile reconstruction attack", by which the adversary produces as output a guess of the original profile behind a given Bloom filter. In doing so, we aim at empirically computing an upper bound on the privacy parameter that will prevent this attack from being effective. Another attack, called the "profile distinguishing game" is described in Section 5.7.

We consider a computationally-unbounded adversary. In the *profile reconstruc-*

tion attack, the adversary exhausts the Bloom filter by querying it on all the possible items of the application domain. More precisely, the attack works as follows: the adversary is given a Bloom filter whose bits are independently and randomly flipped with probability p, which is assumed to be public. Afterwards, the adversary performs some computation (possibly during an unbounded duration) and outputs a set of items that corresponds to its guess of the underlying profile behind the given Bloom filter. We measure the success of this attack by measuring how close the reconstructed profile is to the original one in terms of the cosine similarity (cosine similarity equals the square root of recall times precision). Note that due to the probability of false positives inherent to Bloom filters, the exact reconstruction of the original profile with 100% confidence may not always be possible.

Algorithm 6 ReconstructProfileFromFlippedBloom($\mathcal{B}$: Bloom filter, $\mathcal{H}$: set of hash functions, m: number of bits in the Bloom filter, n number of items in the system, $c \in (0, 1)$)

```
reconstructedProfile ← ∅
for all item i = 1 to n do
    Let k_0 ← |{z : B_z = 0 ∧ z ∈ H(i)}|
    if q(k_0, |H(i)| − k_0) > c then
        reconstructedProfile ← reconstructedProfile ∪ {i}
    end if
end for
return reconstructedProfile
```

The profile reconstruction attack is described in Algorithm 6. In a nutshell, the profile reconstruction attack works as follows. For each item in the system, the adversary checks its corresponding bits in the Bloom filter it observed. Then, k_0 is set to be the number of those bits that were found to be 0 while k_1 is the complementary quantity. Using those two values, the adversary calls a predicate q to determine if an item should be included or not in the reconstructed profile. More precisely, the predicate $q(k_0, k_1) > c$ is defined as $p^{k_0}(1-p)^{k_1}\binom{k_1+k_0}{k_0} > c$, for $0 < c < 1$ a constant and p the flipping probability applied by the BLIP mechanism. The main intuition behind the use of $q(k_0, k_1)$ is that it represents the probability that k_0 bits were flipped while k_1 bits were not flipped.

5.6.1 Experimental evaluation

In order to assess how the variation of the privacy parameter ε affects the success of the attack, we conduct an experiment on Delicious, Digg, and Survey datasets introduced earlier. The objective of this experiment is to empirically derive an upper bound on ε, such that for all c, the success of the adversary in reconstructing the profile through the inference attack is as low as possible. In this experiment, the adversary performs the profile reconstruction attack on each Bloom filter of each user for different values of ε and c. Finally, the cosine similarity is computed between the reconstructed profile and the original profile. All values of c between 0 and 1 in steps of 0.01 have been considered. Then, for each ε the adversary success is measured

by the maximum, over all values of c, of the expected similarity value, where the expectation is taken over the users.

Figure 5.2 shows the success of this attack on the three datasets. For instance, the smallest cosine similarity attained in the Survey dataset is about 0.56 whereas it is 0.04 in the Delicious dataset. This variation may stem from the difference in sparsity between the datasets (*i.e.*, the average number of items in a peer's profile compared to the total number of the items in the system). The baseline, which corresponds to the left-most point in the plot at $\varepsilon \to 0$ is the cosine similarity when the adversary performs a blind guess without observing the flipped Bloom filter. No privacy-preserving system can lead the adversary to output a guess with a similarity that is worse than the baseline, as the adversary could just ignore the output of the system and guess blindly. The success of BLIP in protecting the privacy should be measured relative to that baseline. Hence, we can conclude that BLIP successfully prevents this attack from producing a reconstructed profile having a cosine similarity significantly higher than the baseline when ε is less than 10, for all the considered datasets. Therefore, 10 would be a candidate for being the upper bound on ε for these three datasets.

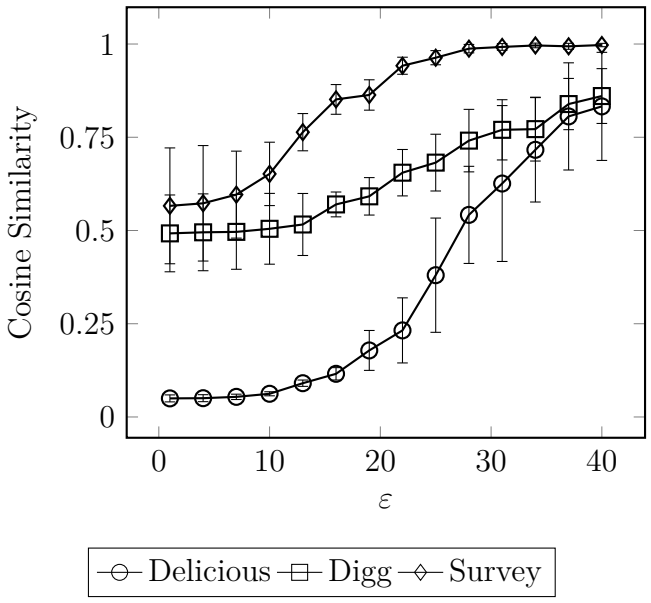


Figure 5.2 – Profile reconstruction attack. The vertical bars represent the standard deviation. The values on the y-axis are cosine similarity between the original profile and the reconstructed profile.

To summarize, the exact value of the similarity threshold above which the profile reconstructed attack is considered successful may depend not only upon the application considered but also upon the privacy preferences of the individual from which the Bloom filter is computed. However, choosing this value to be too conservative (*i.e.*, too low) is likely to decrease dramatically the utility of the output. Therefore, the achievable trade-off between utility and privacy should be set by also taking into account the error bounds discussed in Section 5.4.4.

5.7 Profile distinguishing game

Thereafter, we also describe another inference attack in the form of a game that highlights the risk of adding an item to a profile in terms of whether an adversary would be able to guess its presence (or absence). The motivation behind this attack is that protecting the presence or absence of a single item is the main objective of differential privacy. In order to verify for which values of ε this promise holds in practice, we design the profile distinguishing game. The profile distinguishing game works as follows:

1. Given a profile $\boldsymbol{d}$, pick an index i at random such that $\boldsymbol{d}_i = 1$.
2. Define a new profile $\boldsymbol{d}'$, such that $\boldsymbol{d}'_j = \boldsymbol{d}_j$ for $j \neq i$ and $\boldsymbol{d}'_i = 0$.
3. BLIP both profiles, which generates respectively $\mathcal{B}$ and $\mathcal{B}'$.
4. Give the adversary either ($\mathcal{B}$,$\mathcal{B}'$) or ($\mathcal{B}'$,$\mathcal{B}$) uniformly at random.
5. The adversary should guess the Bloom filter representing $\boldsymbol{d}$.
6. $\mathcal{A}$ outputs 1 if it guessed the Bloom filter at position 1, and outputs 0 if it guessed it was the Bloom filter at position 2.

Algorithm 7 Guess($\mathcal{B}$: Bloom filter, $\mathcal{H}$: set of hash functions, m: number of bits in the Bloom filter, $c \in (0,1)$, i: item to distinguish)

Let $k_0 \leftarrow |\{z : \mathcal{B}_z = 0 \wedge z \in \mathcal{H}(i)\}|$
return $q(k_0, |\mathcal{H}(i)| - k_0) > c$

Algorithm 8 DistinguishProfile($\mathcal{B}_1, \mathcal{B}_2$: Bloom filter, $\mathcal{H}$: set of hash functions, m: number of bits in the Bloom filter, $c \in (0,1)$, i: item to distinguish)

guessB1 $\leftarrow$ Guess($\mathcal{B}_1, \mathcal{H}, m, c, i$)
guessB2 $\leftarrow$ Guess($\mathcal{B}_2, \mathcal{H}, m, c, i$)
if guessB1 = guessB2 **then**
  return choose uniformly at random from $\{\mathcal{B}_1, \mathcal{B}_2\}$
end if
if guessB1 **then**
  return $\mathcal{B}_1$
else
  return $\mathcal{B}_2$
end if

The best privacy guarantees occurs when the probability of making the right guess is close to 0.5, which is the value that an adversary would obtain by guessing completely at random by tossing a fair coin. In general, the adversary may use any strategy it wants and can have arbitrary background information.However, we have designed a heuristic that would at least provide (empirically) a lower bound on the

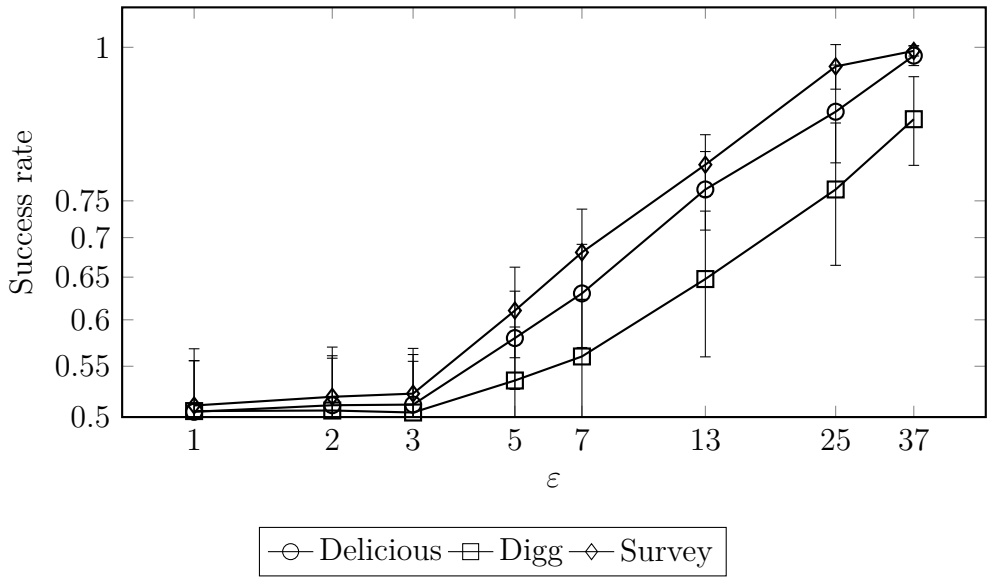


Figure 5.3 – Profile Distinguishing Game. The bars represent the standard deviation.

success rate of the guessing probability of the adversary. This strategy follows closely the strategy used in the profile reconstruction attack (Section 5.6), in the sense that it relies on the same predicate q indicating whether or not an item belongs to a BLIPed profile. The full algorithm is in details in Algorithm 8.

The proposed algorithm have been tested experimentally on the same three datasets used throughout the chapter. In these experiments, the adversary performs the profile distinguishing game on each Bloom filter of each user for different values of ε and c (the value used by the predicate q, similar to Section 5.6.1). For each user and each setting, the BLIPping is done 100 times, each time with different random coins. The average number of successes is taken to be the expected success probability for this user and this setting. This value is averaged over all users and the standard deviation is taken over all users as well. After that, for each ε the maximum success rate is taken over all value for c (therefore, for different values of ε, a different value for c may be the one chosen). This last value, the maximum success rate, is the one plotted against ε in Figure 5.3.

From the figure we can see figure out that if we want the success probability to be less than 55% we should set ε to be below 3.6. Therefore this attack gives a better upper bound for the privacy parameter than the previous attack.

5.8 Conclusion

In this chapter, we proposed BLIP (for BLoom-then-flIP), a differentially private non-interactive mechanism that releases a randomized version of the Bloom filter representation of a profile. The randomized Bloom filter offers high privacy guarantees (in the sense of differential privacy) while still maintaining a good level of utility. For instance, the differentially private Bloom filter can be used to compute a similarity measure, such as cosine similarity or inner product with another Bloom

filter in a non-interactive manner. We have demonstrated how the BLIP mechanism affects the privacy of the underlying profile and how to guarantee privacy at the level of items (as opposed to bits of the Bloom filter) by tuning the flipping probability. We have also described generic inference attacks against the flipped Bloom filter, including the profile distinguishing game and the profile reconstruction attack, the latter enabling reconstruction of the full original profile (almost) if applied on a plain (*i.e.*, non-randomized) Bloom filter but does no longer work on the perturbed Bloom filter if the value of the parameter ε is chosen wisely. For future research it would be interesting to investigate variations that provide more accuracy without sacrificing much privacy or efficiency.

5.8.1 Interactive vesrus Non-interactive

On the question of whether to use the interactive protocols of Chapter 3 of the non-interactive protocol of this chapter, we note that each one has is pros and cons. The interactive protocol provide higher accuracy, although not much higher. However, they also support dynamic profiles and thus are more suitable for applications where the list of items a user has liked change very often, such as micro-blogging platforms for instance. On the other hand, the non-interactive protocol might only have a limited support for dynamic profiles; where older items are removed and a new release of the profile occurs only when the current profile is disjoint from the one associated with the previous release, which make the non-interactive protocol more relevant for applications where the user does not add new items to his favorites except rarely, such as an application in which users express their interest in Renaissance artists. Nonetheless, other advantage of the non-interactive protocols is that is incurs much less computational cost, rendering them more suitable for weak devices or devices in which the emphasis is on preserving energy, such as mobile phones. Moreover, non-interactive protocols would not require the active participation of the user in computing similarity with other users. In particular, the user may deposit his flipped Bloom filter somewhere, either with other users or on a centralized server[4], and turn offline. Additionally, since there is no need to hide the identity of the user when using the non-interactive protocol (as there is not privacy budget issue), it is easier to save the identities of the users in the current clustering view over several sessions; saving the time needed to find them again when the user reconnects. It also makes it possible to exchange the users from the clustering view with other users from the clustering view of other users.

[4]It is even possible in this case to delegate the responsibility of finding similar peers to the centralized server without worrying about privacy, in which case this is a *hybrid system* between centralized and decentralized architectures.

6
Inference Attacks

Abstract

Differential privacy has been widely used in recent years as a method for providing strong privacy guarantees for query answers over binary vectors. In particular, it guarantees that the query answers are insensitive to individual bits. Therefore, while protecting privacy of individual bits, it still captures global information about the binary vector as a whole. This privacy guarantee holds relative to a strong adversary who knows all bits except one, along with arbitrary auxiliary information. However, somehow counter-intuitively, the privacy guarantees are not as strong for a more common and weaker adversary who does not know the exact value of any bit, but instead is aware of the correlations between bits. The inter-dependencies between bits is leveraged to cluster bits together into *landmarks*. Then, we study two attacks that this adversary could use to reconstruct the binary vectors given differentially-private answers to a few chosen counting queries defined by those landmarks.

6.1 Introduction

Kifer and Machanavajjhala [84] raised the concern of the existence of dependencies between the bits of a profile when considering the guarantees provided by differential privacy for binary vectors [84]. In particular, they demonstrated an artificial example of correlations enabling an adversary to fully reconstruct the private data given a differentially-private query answer. The real issue underpinning this is not the presence of correlations *per se*, but rather that the adversarial model typically assumed by differential privacy is *too strong*. By "strong", we mean that the adversary is assumed to know too much about the private data. Their example succeeds in breaching the privacy only for a weak adversary, but not for the strong one assumed by the differential privacy definition. Informally, a privacy breach occurs when an adversary gains non-negligible amount of information after observing a privacy-preserving query answer. If the adversary is so strong that it already knows everything about the private data, then even if the entire data was released then no privacy breach takes place as the adversary did not gain any new information. However, if the adversary turns out to be weaker instead, then a privacy breach can take place. By analogy, in differential privacy the adversary is assumed to know every bit except one (*i.e.* every bit in a binary vector except one), while in reality an adversary is much weaker. Relative to the assumption that an adversary knows every bit but one, then if the adversary ends up knowing every bit except one after the release, no privacy breach occurs. However, whether a privacy breach actually occurs or not is relative to what the adversary knew before the release. Thus, if the adversary did not know anything but ended up learning each bit except one, a privacy breach does occur, although differential privacy as a condition on the release mechanism alone still holds. The ability of the adversary that does not know any bit (a *blind* adversary) to learn bits it could not learn before release depends on the auxiliary knowledge it has. In this chapter, we assume a type of auxiliary knowledge that is useless on its own (*i.e.*, before release), but is nonetheless realistic to assume

that an adversary has, which is the correlations between bits. Although correlations are not necessary for this breach to occur, they act as an amplifier that allows to obtain a higher success rate. We develop two attacks that employ correlations.

Non-interactive private similarity. Dinur and Nissim noticed that if the amount of perturbation to the inner product is $o(\sqrt{n})$, then $k = n\log^2(n)$ queries are sufficient to reconstruct the binary vector with arbitrary precision [30]. In this chapter, we analyse non-interactive differential privacy mechanisms [44]. Two non-interactive differentially private mechanisms has been specifically designed for estimating pairwise similarity and distance. The first is the BLIP mechanism (*cf.* Section 5) and the second is the Johnson-Lindenstrauss mechanism [45]. The former is based on randomized-response and is designed to estimate inner product. The latter is based on sketching (using dimensionality-reduction) and is designed to estimate Euclidean distance. Since Euclidean distance can be converted to inner product[1], our attack employs inner product and is agnostic to the underlying mechanism.

The chapter is organized as follows. In Section 6.2 we describe the theory behind the phenomenon and the attack. Then in Section 6.3 we present the results of the experiments before concluding in Section 6.4.

6.2 Attacks

We describe several attacks exploiting item correlations in this section. The adversary is attacking a collaborative platform with m users and n items, thus User profiles are n-dimensional binary vectors. The adversary is assumed to have knowledge of all the profiles within the platform except one and he uses this knowledge to extract the correlations between items as shown in the following section. The aim of the adversary is to reconstruct the profile of a particular user given only a sketch of the user's profile. The sketch can be used to estimate similarity between the user's profile and arbitrary binary vectors chosen by the adversary, such that these similarities are ε-differentially private. Those binary vectors are called *landmarks* and we explain how the adversary chooses them in the next section.

6.2.1 Landmarks

The attacks start by collecting correlated items together into clusters, called *landmarks*. The items are collected into landmarks using a hierarchical clustering technique [126]. To compute the clustering, a distance function between pairs of items is needed. In our setting, we use the Hamming distance between the corresponding profiles of the items as the distance function. An item profile is the binary vector indexed by users, assigning ones to users who have the item and zero to users who do not have it. The Hamming distance captures the number of users who have exactly one of the two items but not the other, therefore insisting on the fact that we would like the two items to be co-occurring (*i.e.* correlated). The criterion used to merge the clusters is the Ward's criterion [126], which minimizes the inter-cluster variance.

[1] Via the identity $2\langle x, y\rangle = \|x\|_2^2 + \|y\|_2^2 - \|x - y\|_2^2$.

We set the target number of clusters to $\lfloor n/k \rfloor$, resulting in an average cluster size of k item, a parameter to be chosen. In particular, the number of similarity values (queries) the adversary uses is bounded from above by $\lfloor n/k \rfloor$. However, since the differentially-private mechanism is non-interactive, the perturbation is done once and randomness cannot be averaged out, independently of the number of queries were. This means that the parameter k only affects the success of the attack because of properties inherent to the attack semantics itself and not the release mechanism. The formal definition of a landmark is provided next.

Definition 6.1 (Landmark)**.** A landmark is a binary vector $\ell \in \{0,1\}^n$ satisfying the following conditions: (a) at least 2 bits are set to one, and (b) if a profile p is drawn uniformly at random from the dataset and $z = \{i|\ell_i = 1\}$ then the distribution on $x = \sum_{i \in z} p_i$ has most of its probability weight in the tails. Similarly, we can say that it is more likely to find the bits $(p_i)_{i \in z}$ agreeing on a common value than otherwise.

Consider an arbitrary profile $p \in \{0,1\}^n$, if p contains a landmark ℓ or equivalently $\langle p, \ell \rangle = \langle \ell, \ell \rangle$ then we say that $\ell \in p$. If p does not contain any part of ℓ or equivalently $\langle p, \ell \rangle = 0$ then we say that $\ell \notin p$. Both statements can be simultaneously false, for instance, if p contains some bits of ℓ but not all of them. If a landmark ℓ has n_ℓ ones, that is $\langle \ell, \ell \rangle = n_\ell$, then the inner product $\langle p, \ell \rangle$ to a profile p may produce up to $n_\ell + 1$ distinct values, namely $\{0, \ldots, n_\ell\}$. The adversary is trying to decide whether p contains ℓ or not. More precisely, the adversary is trying to distinguish the two cases $\ell \in p$ and $\ell \notin p$. Let $z = \{i|\ell_i = 1\}$ be the set of indexes whose corresponding value in ℓ is 1, then let $p^{(1)} \in \{0,1\}^n$ be an arbitrary binary vector such that $\ell \in p^{(1)}$, and let $p^{(2)}$ be such that $\ell \notin p^{(2)}$ but $p_i^{(1)} = p_i^{(2)}$ for $i \notin z$. Both profiles are identical except for the fact that one of them contains all the bits of ℓ while the other contains none. If $\mathcal{M} : \{0,1\}^n \to \mathbb{R}$ is an ε-differentially private mechanism then the following statement [43] follows directly from the definition of differential privacy for all $S \subseteq \mathbb{R}$:

$$\exp(-\varepsilon n_\ell) \leqslant \frac{\Pr[\mathcal{M}(p^{(1)}) \in S]}{\Pr[\mathcal{M}(p^{(2)}) \in S]} \leqslant \exp(\varepsilon n_\ell) \ , \tag{6.1}$$

since the Hamming distance between $p^{(1)}$ and $p^{(2)}$ is n_ℓ. This fact will be used later in Section 6.2.3.

Thus, for a ε-differentially private inner product $\widetilde{\langle p, \ell \rangle}$ between landmark ℓ and a profile p we have

$$\frac{\Pr[\widetilde{\langle p, \ell \rangle} \subseteq S | \ell \notin p]}{\Pr[\widetilde{\langle p, \ell \rangle} \subseteq S | \ell \in p]} \leqslant \exp(\varepsilon n_\ell) \ ,$$

for all $S \subseteq \mathbb{R}$. Hence, using Bayes rule [127] we have for all $S \subseteq \mathbb{R}$

$$\frac{\Pr[\widetilde{\langle p,\ell\rangle} \subseteq S \mid \ell \notin p]}{\Pr[\widetilde{\langle p,\ell\rangle} \subseteq S \mid \ell \in p]} = \frac{\Pr[\widetilde{\langle p,\ell\rangle} \subseteq S \wedge \ell \notin p]}{\Pr[\widetilde{\langle p,\ell\rangle} \subseteq S \wedge \ell \in p]} \times \frac{\Pr[\ell \in p]}{\Pr[\ell \notin p]} \tag{6.2}$$

$$= \frac{\Pr[\ell \notin p \mid \widetilde{\langle p,\ell\rangle} \subseteq S]}{\Pr[\ell \in p \mid \widetilde{\langle p,\ell\rangle} \subseteq S]} \times \frac{\cancel{\Pr[\widetilde{\langle p,\ell\rangle} \subseteq S]}}{\cancel{\Pr[\widetilde{\langle p,\ell\rangle} \subseteq S]}} \times \frac{\Pr[\ell \in p]}{\Pr[\ell \notin p]} \tag{6.3}$$

$$= \frac{\Pr[\ell \notin p \mid \widetilde{\langle p,\ell\rangle} \subseteq S]}{\Pr[\ell \in p \mid \widetilde{\langle p,\ell\rangle} \subseteq S]} \times \frac{\Pr[\ell \in p]}{\Pr[\ell \notin p]} \ , \tag{6.4}$$

then by (6.1)

$$\exp(-\varepsilon n_\ell) \times \frac{\Pr[\ell \notin p]}{\Pr[\ell \in p]} \leqslant \frac{\Pr[\ell \notin p \mid \widetilde{\langle p,\ell\rangle} \subseteq S]}{\Pr[\ell \in p \mid \widetilde{\langle p,\ell\rangle} \subseteq S]} \leqslant \exp(\varepsilon n_\ell) \times \frac{\Pr[\ell \notin p]}{\Pr[\ell \in p]} \ , \tag{6.5}$$

which means that the posterior of the adversary given its prior probability would be bounded for *any* attack on any differentially-private mechanism.

6.2.2 Attack 1: Threshold per landmark

Consider an adversary that has access to a set of profiles $P \in \{0,1\}^{(m-1)\times n}$ different from profile p that is currently being attacked. The adversary uses P to generate a set L of landmarks in which each landmark ℓ belongs to $\{0,1\}^n$. Let the number of ones in the landmark ℓ be n_ℓ (*i.e,*, $n_\ell = \|\ell\|_1$). The adversary computes a threshold value τ_ℓ for each landmark ℓ, as described in the following paragraph. To reconstruct the unknown profile p, the adversary computes the differentially-private inner product between p and each landmark in L. Denote such perturbed inner product as $\widetilde{\langle p,\ell\rangle}$. and by C be the set of landmarks whose perturbed inner product with p is greater than their corresponding thresholds (*i.e.* $C = \{\ell \in L \mid \widetilde{\langle p,\ell\rangle} > \tau_\ell\}$). The adversary's guess is the smallest (with the least number of ones) profile that contains all the landmarks in C.

Threshold computation. The adversary constructs two sets of profiles for each landmark $A_\ell = \{p \in P \mid \langle p,\ell\rangle = 0\}$ and $B_\ell = \{p \in P \mid \langle p,\ell\rangle = n_\ell\}$. The first set A_ℓ is the set of profiles whose true inner product with the landmark is zero, and the other set B_ℓ is the set of profiles whose true inner product with the landmark is maximal (equal to the number of ones in the landmark). If $B_\ell = \varnothing$, ignore ℓ in which case the bits in ℓ never occur together; that they are strongly anti-correlated. This can occur for large k, and may affect the success rate of the attack because those ignored bits will never be predicted.

Next, the adversary queries the differentially-private mechanism for the perturbed inner product between the landmarks and each of these profiles, ending up with $\tilde{A}_\ell = \{\widetilde{\langle p,\ell\rangle} \mid p \in A_\ell\}$ and $\tilde{B}_\ell$ (defined similarly) representing the perturbed inner products. The threshold τ_ℓ is then chosen as the value that separates $\tilde{A}_\ell$ and $\tilde{B}_\ell$. Let $d_A(x) = \sum_{a\in\tilde{A}_\ell}|x-a|/|\tilde{A}_\ell|$ be the average distance from y to every value

in $\tilde{A}_\ell$ ($d_B(x)$ is defined similarly). Then the threshold value for the landmark ℓ is chosen as:

$$\tau_\ell = \arg\min_y \left(d_A(y) + d_B(y)\right) \quad . \tag{6.6}$$

6.2.3 Attack 2: Hypothesis testing

Given a profile p and a landmark ℓ, then with high probability either p contains all of ℓ or none of it, due to the definition of landmark. This is equivalent to saying that with high probability $\langle p, \ell \rangle$ is either 0 or n. This attack exploits this observation by testing two null hypotheses: $H_> \stackrel{\text{def}}{=} [\langle p, \ell \rangle > 0]$ (versus the alternative hypothesis $H_\notin \stackrel{\text{def}}{=} [\langle p, \ell \rangle = 0]$) and $H_< \stackrel{\text{def}}{=} [\langle p, \ell \rangle < n]$ (versus the alternative hypothesis $H_\in \stackrel{\text{def}}{=} [\langle p, \ell \rangle = n]$). The tests used are likelihood ratio tests, explained in the following section. If either of the two tests rejects, then the attack decides and terminates. If both fail to reject (inconclusive case) then the adversary rejects $H_<$ (*i.e.*, decide that ℓ belongs to p, the profile under attack) with probability equal to the prior probability of the landmark ℓ. The prior probability of a landmark ℓ is the fraction of profiles in the training set that contain it. Due to the way they are constructed, both tests cannot reject simultaneously.

Likelihood ratio test. Let $\widetilde{\langle p, \ell \rangle}$ be the perturbed inner product obtained from a differentially-private mechanism. Then, the likelihood ratio [127] of the event $\ell \in p$ given the perturbed inner product s is

$$\Lambda_\in(s) = \frac{\Pr[\widetilde{\langle p, \ell \rangle} = s \mid \langle p, \ell \rangle = n_\ell]}{\Pr[\widetilde{\langle p, \ell \rangle} = s \mid \langle p, \ell \rangle = 0]} = \frac{\mathcal{L}[\langle p, \ell \rangle = n_\ell \mid s]}{\mathcal{L}[\langle p, \ell \rangle = 0 \mid s]} = \frac{\mathcal{L}[\ell \in p \mid s]}{\mathcal{L}[\ell \notin p \mid s]} \quad , \tag{6.7}$$

in which $\mathcal{L}(\theta \mid x) \stackrel{\text{def}}{=} \Pr[x \mid \theta]$ is the likelihood function. Moreover, the likelihood ratio of the event $\ell \notin p$ given the perturbed inner product s is

$$\Lambda_\notin(s) = 1/\Lambda_\in(s) = \frac{\mathcal{L}[\ell \notin p \mid s]}{\mathcal{L}[\ell \in p \mid s]} \quad . \tag{6.8}$$

According to the likelihood ratio test the adversary should reject the null hypothesis $H_<$ if the likelihood ratio $\Lambda_\in(S)$ is less than some threshold $\eta_\in$. This threshold is chosen according to α, the desired probability of Type I error (the error of incorrectly rejecting a null hypothesis) using the Neyman-Pearson lemma [127]. More precisely, we fix α and then choose $\eta_\in$ that satisfies the following equation

$$\Pr[\Lambda_\in(s) \leqslant \eta_\in \mid \ell \in p] = \alpha \quad . \tag{6.9}$$

The adversary will perform the two hypothesis tests using $\alpha/2$ for each, so that total probability of Type I error is α. If $\widetilde{\langle p, \ell \rangle}$ is a random variable satisfying differential privacy, then both $\Lambda_\in$ and $\Lambda_\notin$ cannot be lower than $\exp(-\varepsilon n)$ as shown in (6.1). In addition, if $\eta_\in < \exp(-\varepsilon n)$, then the test will always fail to reject. Hence, ε sets a lower bound on the Type I error probability α.

If the likelihood function of the particular differential privacy mechanism used is not known or is difficult to compute, the adversary can use kernel density estimation

[128, 129] to estimate it and based on it obtain the required thresholds $\eta_{\in}$ and $\eta_{\notin}$ needed for the likelihood ratio test. This approach requires the knowledge of the true inner product between each landmarks and each profile in the training set.

6.3 Experimental evaluation

To evaluate the impact of having correlations in the dataset, we have tested our attacks on the Survey dataset (*cf.* Section 2.4) and a synthetic dataset generated out of the survey dataset with similari priors for the bits but each bit is otherwise generated independently from the others (thus its bits are uncorrelated). We compare with a Naïve Bayes attack (adapted from 5.6 to incorporate item priors). We also carry out our attack using two different differentially private mechanisms, BLIP (*cf.* Section 5) and Johnson-Lindenstrauss [45], which we call JL for brevity.

Our attacks use both mechanisms, BLIP and JL, as a black box, considering only the perturbed inner product computed by both of them. The Naïve Bayes attack, on the other hand, is specific to BLIP and uses the flipped Bloom filter output by BLIP as a white box by taking into account the mechanics of the Bloom filter, however without exploiting correlations. Thus, the Naïve Bayes attack is not a baseline but rather an incomparable attack.

We measure the success of the attack by how close the reconstructed profile is to the true profile in terms of cosine similarity (*cf.* Section 2.2). For comparison, we also plot the cosine similarity between the constant binary vector of all ones $(1, \ldots, 1)$ and the true profile.

In Figure 6.1, we evaluate the effectiveness of the attacks with respect to the parameter k at $\varepsilon = 100$. It is clear that for the threshold per landmark (TPL) attack, the higher k is, the less effective the attack. However, the hypothesis testing (HT) attack maintains its effectiveness even as k varies. The value of $\varepsilon = 100$ provides very low privacy as evidenced by the 100% success rate of the Naïve Bayes attack, but TPL and HT attacks are not able to catch up except for the TPL attack on the BLIP dataset at $k = 2$. We can also observe that the success rate dramatically decreases, for the TPL attack on both the BLIP and JL mechanisms when the same attacks are run on an uncorrelated but similar dataset, indicating that the correlations are key to its performance, while the HT attack does not seem to depend that much on correlations. The uncorrelated dataset is generated from the original dataset by randomly generating profiles whose bits have the same prior probability as the original dataset but are instead generated randomly of other bits, hence the bits are uncorrelated to one another.

For Figure 6.2, we plot the effectiveness of the attacks for $k = 5$ while ε varies. We observe that the attacks seem to achieve better results as ε increases (less privacy) for the correlated dataset, while they seem insensitive to ε in the uncorrelated dataset.

6.4 Conclusion

In this chapter, we analyzed the privacy guarantees provided by differential privacy when the adversary is weaker than what is generally assumed. The rationale be-

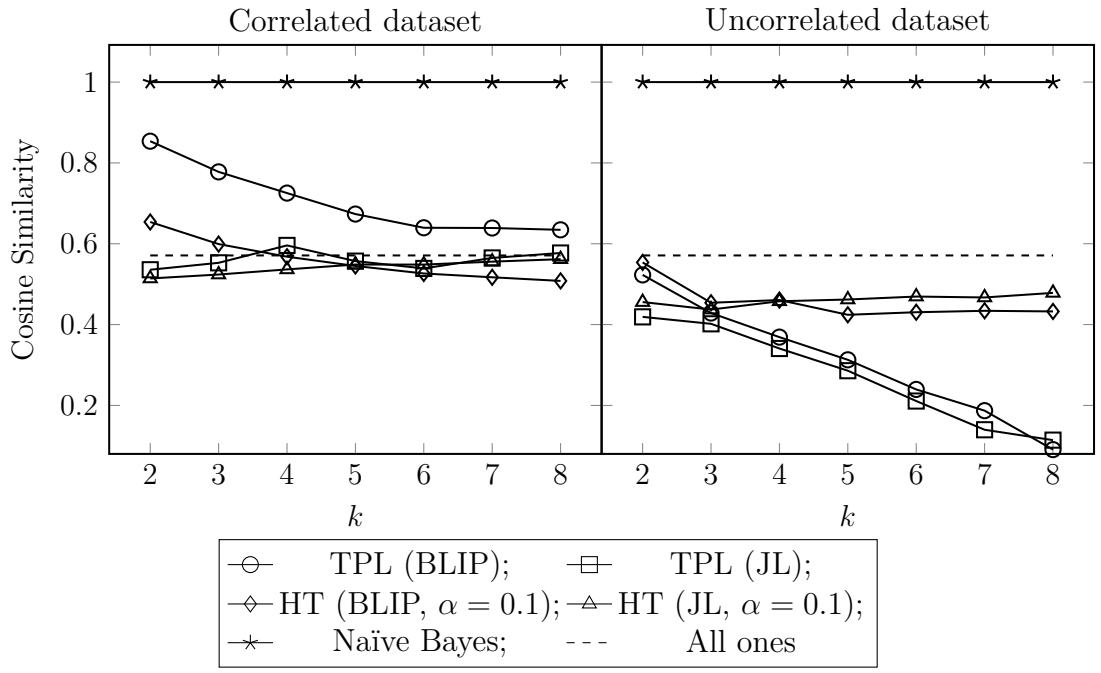


Figure 6.1 – Survey at $\varepsilon = 100$. The plot on the left is for the survey dataset, while the plot on the right is for an uncorrelated version of the survey data, where each item is generated according to its probability of occurrence. TPL (Threshold-Per-Landmark) is the attack from Section 6.2.2, while HT (Hypothesis Testing) is the attack from Section 6.2.3.

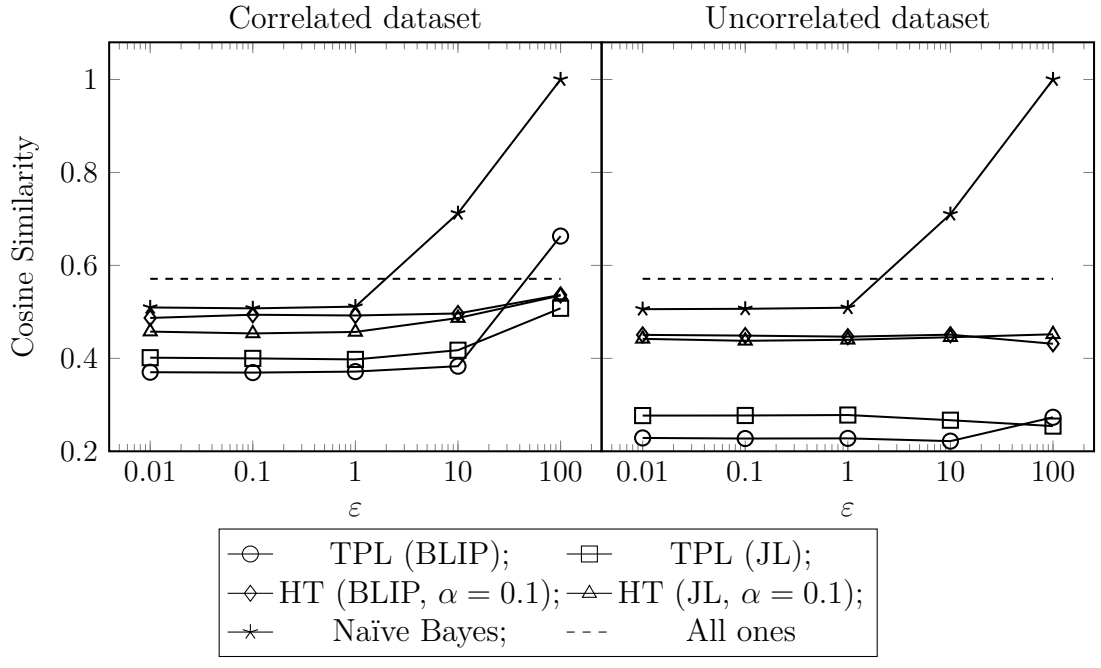


Figure 6.2 – Survey at $k = 5$. The plot on the left is for the survey dataset, while the plot on the right is for an uncorrelated version of the survey data, where each item is generated according to its probability of occurrence.

hind the analysis is that a piece of information may be useless to a knowledgeable adversary, causing no breach, but valuable to a weaker adversary, resulting in an unexpected privacy breach. In particular, the traditional adversary in differential privacy knows every bit in the private binary vector except for one, but we consider an adversary with knowledge only about the correlations between the bits but not the values of the bits themselves. This type of adversary is especially realistic in the context of collaborative filtering systems. We designed two attacks to exploit these correlations given access to a non-interactive differentially private mechanism. We also considered two different non-interactive differentially private mechanisms, one that is provided in this thesis and another one from the literature. We evaluate our attacks against a real life dataset and a stripped-down version of the same dataset but without correlations.

We conclude that an adversary that knows correlations between bits has an advantage over a blind adversary. Such an adversary is able to reconstruct a significant portion of the profile, even when a relatively stronger adversary cannot guess the single bit that is unknown to him (*cf.* Section 5.7). We also observe that a white box adversary targeting the mechanics of a particular mechanism outperforms a black box adversary, thus understanding what is possible in white box versus black box attacks is an interesting research direction.

7
Conclusion

This chapter concludes the thesis by summarizing the main results of the thesis and highlighting potential directions for future work.

Summary

In this thesis, we have addressed the problem of privately computing pair-wise similarity in a large-scale and dynamic peer-to-peer collaborative platform.

In Chapter 3, we described a secure two-party implementation of a differentially-private mechanism for this task, that does not reveal the similarity value if it is below a certain threshold. Then, we evaluated the resulting protocols and empirically shown that it is feasible to maintain privacy without hurting too much the utility. A theoretical analysis of the false negative rate and false positive rate was provided and empirically validated. We also addressed the privacy budget issue, which otherwise would have imposed a serious restriction on dynamic peer-to-peer systems, by setting a fixed upper bound on the number of interactions for similarity computations a peer can have. To solve this problem we proposed two solutions, one of them is through implementing a non-interactive differentially-private mechanism in Chapter 5. The other one in the interactive setting is via introducing the concept of bidirectional anonymous channels and two realizations of it for passive and malicious adversaries.

Heterogeneity of privacy expectations. In Chapter 4, we jointly address the problem of diversity of privacy expectations among users, and the problem of diversity of sensitivity among the items of one user. More precisely, we described a variant of differential privacy that can capture the diversity of privacy valuations among items. The valuations of one user is on the same scale of the valuations of another user, and so is directly comparable. Providing heterogeneous privacy expectations among users is thus a byproduct of choosing an upper bound for the privacy valuations of one's items on the common privacy valuation scale. We presented the stretching mechanism that can fulfill this variant of differential privacy with some distortion in the output. Our experiments showed that if the percentage of overly strict users is small, the system is capable of providing good utility.

Efficiency. In Chapter 5, we introduce a highly efficient non-interactive protocol for similarity computation. Beside being efficient it also lifts the needs for the bidirectional anonymous channel, which strengthens the robustness of the peer-to-peer network. For instance, peers can log off and then log back in without worrying about losing connections to their personalized but anonymous neighbors. Moreover, peers can computer their similarity with other peers who are offline as long as their BLIPed profiles exist in the network.

The meaning of the privacy parameter. The Dwork-Naor impossibility result [27], states that for *any* privacy-preserving mechanism, if we do not restrict the auxiliary information accessible to the adversary, then a privacy breach (of size equal to the min-entropy of the utility) always occurs. Not only this motivated the notion of differential privacy, it also highlighted that the first step to understanding

privacy (as lack of privacy breaches) is to understand what auxiliary information is accessible to an actual adversary and what the adversary can do with it. In this spirit, we investigated the meaning and implications of the value of the privacy parameter ε, in Chapter 5 and Chapter 6. In particular, we investigated three models for auxiliary information. The first model assumed no auxiliary information and is used a baseline (the profile reconstruction attack), while the second considered the knowledge of all the profile except one item (the profile distinguishing game), and the last one considered the correlations between bits.

Perspectives.

The meaning of the privacy parameter. While the results we obtained when we studied the meaning of the privacy parameter were informative for the context of this thesis, it still may not be entirely accessible to the average user who may be asked by the application's user interface for the value for ε. In particular, the average user should not have to understand the attacks or the technicalities of particular assumptions about the adversary. Attempting to provide an intuitive explanation about ε that is tangible to the user may not be entirely technical. In fact, it may include social and philosophical aspects, which may differ substantially between different communities and cultures. Building a unified model that is capable of capturing these aspects as parameters is a great challenge, but is crucial for wide adoption of differential private mechanisms. Such a unified model should not be restricted to a specific differentially-private mechanism and should provide insight into any such mechanism.

Distortion in HDP mechanisms. Although the stretching mechanism can be applied to a wealth of functions, it is nonetheless not directly applicable to some natural functions, such as the ℓ_0-norm or the minimum. Indeed, when computing the ℓ_0 norm (*i.e.*, the number of non-zero coordinates in a given vector), each coordinate contributes either zero or one regardless of its value. Since the stretching mechanism modifies this value, this mechanism would always output the exact value as long as no privacy weight has been set to zero. For the case of the minimum, due to the fact that the stretching mechanism shrinks each coordinate by a factor corresponding to its privacy weight, the resulting output may not have anymore a relation to the intended semantics of the function min.

Another challenge is to enable users to estimate the amount of distortion in the output that they received out of an heterogenous differentially private mechanism. For instance, for functions such as the sum, recipients will not be able to estimate the correct value without being given the distortion. Although the distortion has an upper bound given by Theorem 4.3, the information needed to compute the upper bound is private. Therefore, releasing the distortion (or even its upper bound) would constitute a violation of privacy. We believe this issue could be solved partially by releasing an upper bound using the traditional Laplacian mechanism at an additional cost of ε amount of privacy.

An important future work includes the characterization of functions that have a

low and high distortion when the Stretching Mechanism is applied to them. Indeed, functions having a high distortion are not really suitable for our HDP mechanism. We also leave as open the question of designing a different mechanism than the Stretching Mechanism achieving HDP with a lower distortion or no distortion at all.

Dynamic profiles for non-interactive mechanisms. In this thesis we considered fixed profiles (*i.e.*, profiles that do not change over time). Dealing with dynamic profiles means that if some peer A updates his profile then he may need to recompute his similarity with some peers in his view or to gradually remove the oldest, in order to keep his view evolving. Recomputing the similarity over the same channel means running into the privacy budget issue and thus the number of similarity computations over the same channel will have to be bounded eventually, restricting the number of times the profile of the users may be updated. As an alternative, removing old peers from the view, perhaps after they run out of their quota of recomputations, is a solution. Nonetheless, it does not work for non-interactive mechanisms such as BLIP because a user who releases two different flipped Bloom filters consumes twice as much from his privacy budget than if he only released one. The number of releases of BLIP would have to be bounded. Although [130] provide a solution for what they call "continual observation", the total number of such observations still has to be bounded. Hence, addressing the problem of dynamic profiles for mechanisms that do not employ the bidirectional anonymous channel, such that the number of releases is unbounded as needed by peer-to-peer systems, remains open.

Résumé en Français

La manière dont les gens interagissent sur Internet a fondamentalement changé au cours de la dernière décennie, car le web est devenue une plate-forme collaborative. C'est devenu une plate-forme collaborative, avec laquelle les utilisateurs peuvent interagir et qu'ils peuvent modifier en exprimant leurs préférences et en partageant des informations.

En outre, les réseaux sociaux, comme Facebook et Twitter, ont été conçus pour offrir une expérience sociale centrée sur l'utilisateur. Les utilisateurs sont les principaux producteurs de contenu et leurs amis sont les principaux consommateurs. Le contenu qu'ils produisent, outre de leurs informations personnelles (date de naissance, travail, loisirs, ***alma mater***, etc.), sont généralement de nature personnelle : leurs photos, d'autres photos qu'ils ont aimées, des articles d'information qu'ils ont trouvé intéressants, et ainsi de suite. Amis et connaissances sont habituellement choisis explicitement par les utilisateurs, et une relation d'amitié peut être symétrique (comme dans Facebook) ou asymétrique (comme dans Twitter).

Beaucoup d'autres sites ne dépendent pas d'un réseau social explicite, mais permettent tout de même aux utilisateurs d'exprimer leurs intérêts, comme Delicious, une plate-forme collaborative de bookmarking et Digg, un agrégateur collaboratif de news. Les informations fournies par les utilisateurs peuvent être exploitées pour extraire les tendances et les sujets d'actualité et ensuite des services personnalisés pourraient être fournis aux utilisateurs en fonction de leurs préférences [1, 2, 3, 4, 5]. En outre, les similarités entre les intérêts des utilisateurs pourraient être identifiées et utilisées pour recommander des éléments en utilisant le filtrage collaboratif. Pour tirer parti de ces informations, le calcul de similarité est au cur de tout système de personnalisation.

Vie privée. Même si les réseaux sociaux ont le potentiel d'améliorer l'expérience des utilisateurs, ils soulèvent également des problèmes importants en terme de vie privée. Les utilisateurs partagent leurs informations et leurs intérêts personnels, dont certains qu'ils pourraient vouloir garder secret. Par exemple, un utilisateur de Digg pourrait exprimer son opinion pour des informations liées au cancer afin d'obtenir plus de ces informations dans son fil de nouvelles personnalisé, mais il pourrait ne pas vouloir que quelqu'un d'autre sache qu'il est intéressé par les nouvelles liées au cancer parce qu'il considère que les informations sur sa santé sont sensibles. Le site hébergeant les données privées de l'utilisateur est supposé mettre en uvre des mécanismes de contrôle d'accès appropriés, par exemple en permettant à l'utilisateur de contrôler qui peut accéder à ses données. Néanmoins, l'échec de ces mécanismes de contrôle d'accès n'est pas rare, y compris du à des failles de sécurité. En outre, le récent programme PRISM de la National Security Agency (NSA) [6] donne accès à la NSA aux renseignements personnels des utilisateurs collectées par des géants d'Internet tel que Facebook ou Google.

De plus, étant donné que les amis s'influencent les uns les autres, les données privées des utilisateurs peuvent être exploitées pour afficher de la publicité aux amis

de l'utilisateur [7, 8]. Par exemple, le site pourrait présenter un film à un utilisateur en montrant que de ses amis, Alice et Bob, l'ont aimé, ce qui peut être considéré comme une violation de la vie privée pour Alice et Bob. Même un internaute égaré, recevant des recommandations publiques et apparemment inoffensives, et quelques informations auxiliaires, peut encore atténuer le mécanisme de contrôle d'accès et déduire les informations personnelles des utilisateurs. Par exemple, Calandrino, Kilzer, Narayanan, Felten et Shmatikov [9] ont montré que, rien qu'en observant les recommandations publiques d'un objet sur Amazon, ils pouvaient déduire les achats privés de certains utilisateurs. En général, la personnalisation implique souvent une certaine forme de violation de la vie privée car les données privées de l'utilisateur doivent être utilisées dans le processus [10].

En outre, même si l'utilisateur n'a pas associé sa véritable identité à son profil sur un site donné, il est toujours possible, étant donné les informations privées qu'il a donné au site, de relier son compte utilisateur à un autre site Web auquel il aurait donné sa vrai identité [11], et donc de le dé-anonymiser.

Réseau pair-à-pair. A partir de cela nous pouvons voir que déléguer les données privées d'un utilisateur à des tiers pour le stockage ou le traitement n'est pas forcément sûr, car les tiers ne sont pas toujours fiables. Au lieu de cela, il est préférable de déployer les réseaux sociaux et les applications sur un réseau Pair-à-Pair (P2P). Contrairement aux systèmes centralisés classiques qui sont composés d'un serveur central traitant les requêtes des clients, dans lesquels les clients communiquent uniquement avec le serveur et ne sont pas conscients de l'existence d'autres clients dans le système, les systèmes P2P sont distribués et le service est fourni via la collaboration explicite et proactive entre les clients. Chaque client est généralement une machine dans le système qui partage une partie de la charge de la fourniture du service. Le réseau P2P est le graphe superposé sur la topologie du réseau physique, et dont les nuds représentent les clients et les arêtes représentent une collaboration directe entre deux clients (ce qui signifie généralement qu'ils peuvent communiquer directement). Les clients dans les systèmes P2P n'ont généralement qu'une connaissance partielle du réseau, ainsi P2P systèmes sont connus pour leur capacité à passer à l'échelle et leur robustesse [12].

Étant donné que les applications dont nous avons parlé sont centrés sur l'utilisateur, les système P2P sont des candidats naturels pour la mise en uvre de ces applications. En particulier, chaque utilisateur, avec ses informations personnelles, équivaut à un noeud du réseau P2P. De cette façon, l'utilisateur va stocker ses données sur sa propre machine, gardant le contrôle complet sur ses données, et n'a pas besoin de faire confiance à quelqu'un d'autre qu'elle-même (étant donné, bien sûr, que son logiciel et son matériel soit fiables). Étant donné que le nombre d'utilisateurs des réseaux sociaux augmente très rapidement, la nécessité de passer à l'échelle donne également une forte incitation pour déployer des applications de manière décentralisée et en P2P [13, 14], au prix cependant d'une complexité accrue du système et un surcout de synchronisation.

Similarité. Dans ce cadre distribué, calculer efficacement des recommandations ou des résultats de recherche personnalisés tout en gardant le trafic réseau raison-

nable est d'une importance primordiale. De nombreux systèmes P2P existants y parviennent en associant un utilisateur uniquement à d'autres utilisateurs avec lesquels il partage des intérêts communs [1, 15, 16, 17] (c'est-à-dire, dont sa similarité est relativement plus élevée qu'il aurait avec d'autres utilisateurs.) En outre, la similarité entre les utilisateurs peut être vue comme une forme de mesure de distance, ce qui est une primitive fondamentale dans de nombreuses applications de fouille de données. Ces distances peuvent être utilisées pour effectuer des tâches de clustering, de calcul des plus proches voisins, ou d'un intérêt direct pour notre contexte, du filtrage collaboratif, qui permet de fournir des recommandations personnalisées.

Par conséquent, avoir une primitive qui fournit des valeurs de similarité entre les utilisateurs est la clé pour assurer de la personnalisation sociale dans les plateformes de P2P collaboratif. Cette thèse est consacrée à la construction d'une telle primitive d'une manière qui assure le respect de la vie privé des utilisateurs, tout en étant efficace en termes de coûts de communication et de calcul. Dans la section suivante, nous détaillons plus en profondeur la problématique de la vie privée dans un tel contexte.

Vie privée

Afin d'effectuer le filtrage collaboratif, les informations concernant les données de plusieurs utilisateurs doivent être combinées. Dans notre cas, cette information est la similarité d'utilisateur à utilisateur. Les deux utilisateurs impliqués dans le calcul de leur similarité seront engagés dans un protocole distribué afin de mener à bien cette tâche. Cependant, il y a deux aspects complémentaires dans l'exercice d'un tel protocole. Considérons tout d'abord le contexte dans lequel la vie privée est définie. En particulier, il faut examiner 1) une fonction à calculer, 2) les entrées de la fonction, dont la vie privée doivent être protégée, et 3) la valeur résultant de la fonction (sortie). S'il n'y avait qu'une seule partie détenant les entrées et effectuant le calcul, il n'y aurait pas de problème de vie privée. Le problème se pose lorsqu'il y a plusieurs parties détenant des différentes parts de l'entrée, dans lequel la part détenue par une partie donné est considérée comme privée pour lui.

Un exemple fondateur de Yao [18] est celui de deux millionnaires, chacun considérant sa richesse comme entrée privée, qui veulent calculer la fonction "supérieur à" pour savoir lequel d'entre eux est plus riche. Les deux parties de cet exemple doivent être engagées dans un protocole afin de combiner leurs entrées et de calculer une sortie, qui est ensuite révélée aux deux parties. Au cours de ce processus, un participant observe 1) la transcription du protocole et 2) la sortie de la fonction. La définition traditionnelle de calcul multiparti sécurisé [19] ne se préoccupe que de garantir que la transcription de protocole ne divulgue pas d'information à propos des entrées des participants. C'est à dire, pour un adversaire observant la transcription du protocole, aucune information sur l'entrée privée des utilisateurs honnêtes ne devrait être divulguée. Dans cette définition, il n'existe aucune garantie sur ce que la valeur de sortie (par opposition à la transcription du protocole seul) peut révéler sur les valeurs d'entrée.

En accord avec cette définition, la fouille de données [20] préservant la vie privée

(comme [21]), ne s'est préoccupée que de ce que la valeur de sortie peut révéler sur l'entrée, et non sur la manière dont la sortie est elle-même calculée. Par exemple, les données peuvent être confiées à un tiers parti de confial centralisé qui effectue le calcul nécessaire avant de publier la sortie. Dans ce cadre, au lieu de garantir que la sortie est la réponse exacte, une légère déviation de la vraie réponse est permise afin de gagner en terme de respect de la vie privée.

Il est donc naturel d'aspirer à empêcher à la fois la transcription du protocole et la valeur de sortie de laisser fuire de l'information sur les entrées de participants. Des constructions générales pour les protocoles sécurisés multipartis pour simuler toute fonctionnalité existent [22, 23, 24] donc il serait simple de mettre en uvre l'un des nombreux mécanismes de differential-privacy en utilisant une telle construction. Toutefois, ces constructions générales sont très coûteuses et irréalisables. Des protocoles spécialisés peuvent être construits pour des fonctions spécifiques afin d'être plus efficace [25, 26]. Cette thèse étend cette ligne de travail dans le monde des systèmes distribués, ce qui soulève un ensemble unique de défis, tels que la dynamicité des systèmes large échelle, les déconnections possibles des pairs, la calcul continu (détaillé plus loin que la question du budget de privacy), et l'hétérogénéité des attentes de la vie privée des différents utilisateurs.

Differential privacy

Une fonction de préservation de la vie privée idéale est une fonction dont la sortie ne relâche aucune information sur son entrée. En particulier, ce qui pourrait être appris après la publication de la sortie de la fonction devrait également être trivialement appris sans observer la sortie de la fonction [27]. Toutefois, si la sortie fournit des informations sur l'entrée (c'est-à-dire, dont l'utilité est mesurée en terme de min-entropy [28]), il a été longtemps soupçonné [29], démontré [30], et enfin prouvé dans [27] que cet idéal est impossible à réaliser. En particulier, Dwork et Naor [27] ont prouvé que si la sortie de la fonction préservation de la vie privée a n bits de min-entropie, alors une faille de n bits de vie privée peut se produire, étant donné certaines connaissances auxiliaires de l'adversaire. Dans le même papier Dwork et Naor suggèrent, comme une solution, de passer du précédent concept absolu de la vie privée à un relatif, qu'ils ont nommé differential privacy. En outre, la nouvelle notion ne dépend pas d'hypothèses concernant la connaissance auxiliaire de l'adversaire. En particulier, la propriété de differential privacy est une propriété de la seule fonction et non de son entrée, ni de l'adversaire, qui a abouti à un cadre fort et théoriquement attrayant qui a été largement adopté par la suite dans la communauté de protection de la vie privée.

L'esprit de la differential privacy est qu'un petit changement dans l'entrée devrait ne pas induire plus qu'un petit changement sur la sortie de la production. La définition exacte de la différential privacy dépend de la façon dont la notion de "petit" est défini formellement. Dans le littérature traditionnelle de la differential privacy, la fonction de calcul prend un vecteur en entrée ; soit un vecteur de zéros et de uns, ou un vecteur d'informations individuelles. Dans ce contexte, une petite variation de l'entrée s'élève à un changement de tout au plus une position dans le vecteur. Par

exemple, pour les vecteurs de zéros et de uns, un petit changement est représenté par une modification de la distance de Hamming par 1. Cette notion a été généralisée pour des distances arbitraires dans [31].

La différential privacy a commencé dans le cadre du contrôle de la divulgation statistique au niveau des bases de données [27]. Dans ce contexte, l'entrée de la fonction est un vecteur de données d'individus, et un petit changement dans l'entrée est la présence ou l'absence des données d'un seul individu. Ainsi, un utilisateur fournissant ses données, par exemple, est assuré que la sortie de la fonction représente les tendances globales entre tous les utilisateurs et contient très peu d'informations qui lui sont spécifiques. Ainsi, tout bris global de vie privée a un impact minimal sur sa propre vie privée. En outre, même le simple fait d'avoir contribué aux données, est protégé lui aussi. De plus, cette garantie tient toujours même si l'adversaire connait tous les autres utilisateurs qui ont fourni leurs données.

Les avantages pour la vie privée de l'utilisateur dans le modèle de base de données statistiques mentionné dans le paragraphe précédent est clair. Cependant, elle nécessite quelques précisions pour montrer les mêmes avantages dans le cas des micro-données dans laquelle l'entrée représente les données d'une seule personne, ce qui est la situation considérée tout au long de cette thèse. Dans cette configuration, un petit changement est défini par rapport à l'absence ou la présence d'un élément dans le profil d'un individu. La différential privacy garantira que toutes les informations qui pourraient être déduites de la sortie ne dépend pas d'un élément particulier. Par conséquent, l'utilisateur n'a aucune raison de s'inquiéter du fait d'ajouter un article en particulier a son profil ou non. Cependant, les tendances globales sur le profil de l'utilisateur, tels que les grandes catégories de ses intérêts peuvent être dévoilées.

Contributions de la thèse

L'objectif de cette thèse est d'aborder le problème du calcul de similarité de manière privée entre les utilisateurs de plateformes collaboratives de type pair-à-pair. Nos contributions sont résumées ici.

Canal anonyme bidirectionnelle Notre première contribution se penche sur la question du budget de confidentialité. Le problème du budget de confidentialité apparait lorsqu'un adversaire est en mesure d'obtenir k calculs de similarité indépendants d'un mécanisme préservant la vie privée. Si le mécanisme respecte la differential privacy, alors k doit avoir une limite de part le lemme de composition [42, 43]. En outre, si le mécanisme est efficace (c'est-à-dire qu'il s'exécute en temps polynomial), alors k doit être $\widetilde{O}(n^2)$, où n est le nombre d'éléments dans le système [131]. En outre, si le mécanisme ajoute $o(\sqrt{n})$ de bruit, alors k doit être $\widetilde{O}(n)$ [30]. De plus, pour le mécanisme de Laplace, qui offre la differential privacy pour $O(1)$ bruit, k doit en plus être contraint à être sous-linéaire $o(n)$ [108]. Dans notre cas, k est le nombre de fois où le pair calcule sa similarité avec un autre pair. Par conséquent, si k est borné, alors les pairs ne seront pas en mesure de calculer leur similarité avec de nouveaux pairs rejoignant le système. Cette limitation est très contraignant pour les systèmes pair-à-pair, où de nouveaux pairs continuent de rejoindre le système tout le temps.

Pour contourner cette limitation, nous remarquons que la réalisation d' un nombre illimité de calculs de similarité, en soi, n'est pas suffisant pour soulever la question du budget de vie privée. Par ailleurs, la question du budget de la confidentialité n'aura pas besoin d'être considéré même si l'adversaire a obtenu les sorties de tous ces calculs de similarité. En effet, l'adversaire doit également relier tous ces calculs de similarité au même paire. Par conséquent, nous introduisons le concept de canal anonyme bidirectionnelle pour empêcher l'adversaire de relier la sortie de calculs de similarité effectués sur différents canaux anonymes au même peer. Si le calcul de similarité se déroule sur un canal privé entre exactement deux pairs, alors cet objectif, que nous nommons non-chaînabilité, est réalisé en fournissant l'anonymat aux deux pairs.

Le terme "anonymat" se définit comme la séparation de l'identité (comme l'adresse IP) du profil privé d'un pair [1, 132], ou des services qu'il fournit, comme dans les services cachés de Tor [59]. Ces systèmes, en fait, offrent des pseudonymes plutôt que l'anonymat. En effet, utiliser un pseudonyme peut assurer la séparation entre l'identité (IP) et un profil privé. Toutefois, cela ne répond pas à l'objectif de non-chaînabilité et donc ne résout pas la question du budget de vie privée. Par conséquent, ce que nous entendons dans cette thèse en utilisant le terme "anonymat" est strictement plus fort que les pseudonymes. En particulier, l'anonymat assure la non-chaînabilité alors que les pseudonymes ne le font pas. Par exemple, alors que FreeRec [132] fournit des pseudonymes pour les deux pairs aux extrémités d'un canal de communication, a la fois les services cachés de Tor [59] et AP3 système éditeur-abonné anonyme [64] fournissent des pseudonymes pour un des pairs et l'anonymat pour l'autre, car ils sont conçus pour le modèle client-serveur. Toutefois, comme dans FreeRec, étant donné que les pairs sont dans un système symétrique, notre canal anonyme bidirectionnel permet l'anonymat a la fois pour l'expéditeur et le destinataire du canal de communication.

Nous décrivons deux protocoles pour la mise en uvre du canal anonyme bidirectionnel, une pour les adversaires passifs et une autre pour les adversaires actifs. Un adversaire passif suit la recette du protocole et ne triche pas, mais il est seulement capable de lire les messages qui passent à travers les pairs qu'il contrôle. A l'inverse, un adversaire actif est capable de modifier les messages, d'effacer des messages, voire d'en créer de nouveaux.

La première méthode pour construire des canaux anonymes bidirectionnels sécurisé contre des adversaires passifs, utilise un pair supplémentaire entre l'émetteur et le récepteur agissant comme un anonymiseur, en supposant qu'il n'est de connivence avec aucun des deux pairs. Supposons qu'un canal anonyme bidirectionnel doit être construit entre Alice et Bob. Alors Alice utilise un service d'échantillonnage aléatoire par les pairs [12] afin de choisir l'anonymiseur uniformément au hasard dans l'ensemble du réseau. Étant donné que l'anonymiseur est choisi uniformément au hasard, la probabilité qu'il s'agisse d'un adversaire est égal au rapport des pairs contrôlés par l'adversaire sur le nombre total de pairs. L'anonymiseur choisit ensuite aléatoirement un autre pair Bob et permet à Alice et Bob d'effectuer un échange de clés [58] à travers lui sans révéler leurs identités mutuelles. Si l'anonymiseur s'avère être un adversaire, alors il est un adversaire passif et ne peut pas monter une attaque de l'homme du milieu afin de compromettre l'échange de clés et donc, une fois l'échange

de clé effectué, il n'est pas capable de lire le messages échangés entre Alice et Bob. Ce canal n'est valable que pour une seule utilisation et ne doit pas être réutilisée afin de fournir la non-chaînabilité et ainsi éviter la question du budget de vie privée. La même exigence vaut également pour la deuxième méthode de construction de canal anonyme bidirectionnel, qui est sûre contre des adversaires actifs.

La seconde méthode utilise une technique d'échantillonnage aléatoire des pairs qui résiste à des adversaires actifs [33]. De plus, il y aura deux pairs impliqués dans le canal anonyme bidirectionnel entre Alice et Bob, respectivement le proxy d'Alice et le proxy de Bob. Alice choisit son proxy uniformément au hasard à l'aide du protocole d'échantillonnage de pairs aléatoire [33], puis en utilisant la clé publique du proxy, elle lancera un canal anonyme vers lui en utilisant la technique décrite dans [60]. Le canal anonyme entre Alice et son proxy fournit l'anonymat pour Alice mais pas pour son proxy. Bob réalise les mêmes étapes pour lancer un canal anonyme à son proxy. Par la suite, lorsque les deux proxys se rencontrent, par échantillonnage aléatoire, le proxy de Alice envoie la clé publique et l'adresse IP du proxy de Bob à Alice, et le mandataire de Bob envoie la clé publique et l'adresse IP du proxy d'Alice à Bob. En utilisant cette information, Alice et Bob conduisent alors un protocole d'échange de clés d'une manière particulière, pour éviter que leurs proxys puissent monter une attaque de l'homme du milieu, aussi longtemps que l'un des deux proxy est honnête. Autrement dit, le proxy d'Alice et le proxy de Bob ne seront pas en mesure de monter une attaque de l'homme du milieu à moins qu'ils soient en collusion, ce qui implique que les deux soient contrôlés par le même adversaire actif. La probabilité que deux proxys appartiennent au même adversaire, puisque le service d'échantillonnage aléatoire fournit des pairs de manière aléatoire uniforme, est $p^2 < p$, dans lequel p est le rapport entre les pairs contrôlés par l'adversaire pour le nombre total de pairs dans le système, en supposant bien sûr que Alice et Bob soient honnêtes. Le protocole d'échange de clé commence par Alice initiant un canal anonyme à sens unique[1] vers le proxy de Bob, ainsi évitant complètement son proxy. Alice utilise ce canal pour envoyer un message a Bob via le canal anonyme et le proxy de Bob, selon le protocole d'échange de clés Diffie-Hellman. Ensuite Bob effectue les calculs nécessaires et envoie la réponse à Alice via un autre canal de communication a sens unique qu'il construit entre lui et le proxy d'Alice. Ce protocole empêche les proxys d'effectuer une attaque de l'homme du milieu car une telle attaque nécessite de controler dans les deux sens le protocole d'échange de clefs, ce qui ne peut pas être fait a moins que les deux proxys ne collaborent. Ensuite, Alice et Bob peuvent communiquer de façon privée via le canal anonyme bidirectionnel alors que leurs proxys ne peuvent pas lire leurs messages. Nous répétons que dans cette thèse tous nos protocoles considèrent un adversaire passif, néanmoins nous décrivons cette construction pour démontrer la faisabilité du canal anonyme bidirectionnel.

Enfin, la question du budget de la vie privée peut aussi être évitée en utilisant des mécanismes non interactifs de differential privacy, car ils libèrent leur sortie une fois pour toutes (c'est-à-dire $k = 1$). La sortie d'un mécanisme de protection des renseignements personnels non interactif est habituellement une esquisse qui peut être utilisé plus tard pour estimer la similarité avec le pair possédant ce croquis,

[1]Les canaux anonymes a sens unique peuvent être moins couteux a construire que les canaux bidirectionnels car ils n'ont pas besoin de fournir au destinataire un moyen de répondre.

autant de fois que souhaité, au prix d'un taux d'erreur élevé. Nous décrivons un tel mécanisme plus tard sous le nom de BLIP.

Protocole en deux parties pour similarité de seuil. Comment font deux pairs pour calculer la similarité entre leurs profils ? Dans le système GOSSPLE, sur lequel est basé notre système, les pairs envoient leurs profils les uns aux autres [1]. De toute évidence, cela ne préserve pas la vie privée puisque les profils sont des données privées. Au lieu de cela, en utilisant le calcul multiparti sécurisé, qui emploie des techniques cryptographiques, les pairs pourraient émuler un tiers de confiance qui reçoit leurs profils privés en entrée et envoie en sortie la similarité entre les deux profils aux pairs [19]. En outre, cela est possible tout en ne révélant rien d'autre sur les profils privés autres que leur similarité. Pour plus de sécurité, nous exigeons également que le protocole ne révèle pas la valeur de la similarité, mais plutôt un seul bit d'information précisant si elle est supérieure à un seuil donné ou non. Non seulement libérer un seul bit d'information est plus privé que de libérer la valeur de similarité, mais il rend aussi plus difficile pour un adversaire de deviner le profil d'un pair particulier, par exemple dans la situation dans laquelle l'adversaire devine des articles afin d'augmenter progressivement la similarité avec un pair particulier jusqu'à ce qu'il reconstitue entièrement son profil. Dans ce cas, le seuil de similarité rend les choses plus difficiles pour l'adversaire, car il n'obtient aucun indice et donc son estimation initiale du profil doit être très bonne.

Des résultats généraux de faisabilité montrant la possibilité de construire un protocole multiparti sécurisé pour toutes les fonctionnalités existent [18, 22, 23]. Ces résultats de faisabilité pourraient être appliquées aveuglément pour obtenir un protocole multiparti de tout mécanisme differentially private tels que le mécanisme de Laplace [38]. Toutefois, ces résultats généraux sont extrêmement inefficaces pour une utilisation pratique. Au lieu de cela, il y a d'autres techniques qui sont plus adaptées, mais ils doivent être adaptés pour des fonctionnalités spécifiques. Ces techniques sont divisées en deux grandes catégories : le chiffrement homomorphe et procédé de partage de secrets [47]. Le partage de secret est plus efficace que le cryptage homomorphe et n'a pas besoin d'hypothèses cryptographiques, fournissant la privacy même face à un adversaire disposant d'une puissance de calcul illimité. Cependant, elle nécessite plus de deux parties et n'est donc pas adaptée à nos besoins, où il n'y a que deux parties qui sont intéressées dans le calcul de leur similarités. Nous employons donc le cryptage homomorphe à la place. Le cryptage homomorphe est un chiffrement qui permet aux pairs d'effectuer des calculs sur le texte chiffré, sans décryptage. Par exemple, un cryptage additif-homomorphes dispose d'une fonction qui prend en entrée deux nombres chiffrés et la clé publique utilisée pour chiffrer et émet un autre numéro crypté, qui, si décrypté donnera la somme des deux nombres originaux. Il peut également permettre la multiplication de numéros non chiffrés, une opération appelée scalaring. $\mathsf{Enc}_{pk}(a)$ désigne le chiffrement du message a en vertu de la clé publique pk, pendant que $\mathsf{Dec}_{sk}(a) = a$ est le déchiffrement de ce message avec la clé secrète sk. Afin de simplifier la notion, nous allons laisser tomber les indices et écrire $\mathsf{Enc}(a)$ au lieu de $\mathsf{Enc}_{pk}(a)$ et $\mathsf{Dec}(a)$ au lieu de $\mathsf{Dec}_{sk}(a)$. Il y a une operation efficace $\oplus$ sur deux messages cryptés de telle sorte que $\mathsf{Dec}(\mathsf{Enc}(a) \oplus \mathsf{Enc}(b)) = a+b$ il y a également une opération efficace de scalaring $\odot$ prenant en entrée un texte

chiffré et un texte clair tels que $\mathsf{Dec}(\mathsf{Enc}(c) \odot a) = c \times a$.

En plus des opérations élémentaires précédentes qui peuvent être effectuées sur place par un pair, il existe des techniques plus sophistiquées pour lesquels il existe des protocoles multipartis, mais nécessitent la collaboration active de plusieurs pairs. Par exemple, l'opération consistant à déterminer lequel de deux nombres cryptés est supérieure à l'autre, sans nous révéler d'autres informations sur les numéros [48, 49, 50]. D'autres exemples incluent la multiplication de deux nombres cryptés [51], ou l'extraction du bit le moins significatif [49], encore sous forme chiffrée, à partir d'un nombre entier crypté.

Le cryptosystème de Paillier [52] est une instance d'un schéma de chiffrement homomorphe. En outre, le système de chiffrement de Paillier est également sémantiquement sûr. Cela signifie qu'un adversaire avec une capacité de calcul bornée ne peut pas tirer des informations non triviales sur un texte clair a compte tenu de son cryptage $\mathsf{Enc}(a)$ et la clé publique pk seulement. Par exemple, un adversaire avec une capacité de calcul bornée qui est reçoit deux textes chiffré différents chiffrés avec la même clé, ne peut même pas décider avec une probabilité non négligeable si les deux textes chiffrés correspondent au cryptage du même texte en clair ou pas. En particulier, le cryptosystème sémantiquement sûr est par essence probabiliste, ce qui signifie que même si le même message est chiffré deux fois, les deux textes chiffrés résultants seront différents, sauf avec une probabilité négligeable. La propriété de la sécurité sémantique est essentielle pour prouver que nos protocoles sont sécurisés contre un adversaire passif. Pour nos protocoles, nous allons aussi utiliser une version a seuil de la cryptographie de Paillier [53] qui exige la coopération active des pairs pour décrypter. Un cryptosystème a seuil (t, n) est un système à clé publique dans lequel au moins $t > 1$ pairs sur n ont besoin de coopérer activement afin de déchiffrer un message chiffré. En particulier, aucune collusion de $t - 1$ ou moins pairs ne peut déchiffrer un texte chiffré. Cependant, tout pair peut chiffrer une valeur lui même en utilisant la clé publique pk. Après que le système de cryptage de seuil ait été mis en place, chaque pair i obtient sa propre clé secrète sk_i (pour $i \in \{1, \ldots, n\}$), qui est inutile en soi, mais doit être utilisé comme entrée d'un protocole de décryptage à seuil par t pairs ou plus pour réussir à déchiffrer un texte chiffré. Dans la définition précédente, lorsque nous disons que les pairs coopèrent pour décrypter cela signifie qu'ils s'engagent dans un protocole interactif : le protocole de décryptage, qui fait partie de la définition du système de chiffrement de seuil. Dans un tel protocole l'entrée de chaque pair est une partie de la clé secrète, avec le texte chiffré. Après que les pairs aient échangé certains messages selon le protocole, le résultat est le texte brut correspondant au texte chiffré.

Nous décrivons deux protocoles, l'un pour le calcul de produit scalaire et l'autre pour le calcul de similarité cosinus. Notre approche ne se limite pas à la similarité cosinus et peut être appliquée à toute métrique de similarité binaire, comme la similarité Jaccard, la distance de Hamming, la distance euclidienne et d'autres, puisque toute métrique de similarité binaire peut être implémentée en utilisant les produits scalaires seul [36]. Le protocole de produit scalaire est utilisé comme un sous-protocole par le protocole de similarité cosinus. Par conséquent, nous allons les décrire comme un protocole de similarité cosinus avec seuil. Comme il est difficile de mettre en uvre la racine carrée nécessaire pour la similarité cosinus dans le cal-

cul multiparti sécurisé, nous mettons en uvre la similarité cosinus carré à la place. Monter au carré la similarité cosinus n'a aucune incidence sur l'utilité parce que la fonction carré est monotone.

Le protocole est composé de plusieurs étapes, y compris le calcul du produit scalaire, l'élever au carré, l'utiliser pour calculer la similarité cosinus carré, puis enfin comparer si la similarité cosinus carré est supérieure à un seuil prédéterminé. Nous les décrivons dans l'ordre. Il existe plusieurs types de protocoles qui peuvent être utilisés pour calculer le produit scalaire entre deux vecteurs binaires. Le premier type de protocoles sont les protocoles calculant directement le produit scalaire tandis que le deuxième type est les protocoles d'intersection d'ensembles de cardinalité. Un protocole d'intersection d'ensemble de cardinalité prend deux ensembles d'entiers en entrées et retourne le nombre d'entiers qu'ils ont en commun. Ce protocole peut aussi être utilisé pour calculer le produit scalaire de deux vecteurs binaires car chaque vecteur binaire peut être représenté comme un ensemble de nombres entiers correspondant aux positions dans le vecteur qui sont égales à 1. Les protocoles d'intersection d'ensembles de cardinalité pourrait fournir un avantage de performance quand n est grand et les vecteurs binaires sont peu denses (à savoir le nombre de bits égal à 1 est faible par rapport au nombre de bits à zéro). Il y a des avantages et des inconvénients pour les approches basées sur les intersections d'ensembles et les produit scalaires. À notre connaissance, tous les protocoles d'intersection d'ensembles sécurisés existants [70, 71, 72, 73] comportent des étapes proportionnels au nombre de uns dans les vecteurs binaires, révélant ainsi cette information. D'autre part, les protocoles de produits scalaires [74, 75, 76, 77] nécessitent intrinsèquement une complexité de communication proportionnelle à n, qui est indépendante du nombre de uns dans les vecteurs binaires, au prix cependant d'une augmentation de la communication. Du point de vue de la complexité de calcul, l'approche à base de produit scalaire a une complexité de calcul linéaire tandis que l'approche de a base d'intersection d'ensembles a une complexité quadratique. Étant donné que notre intérêt principal est la vie privée, nous avons choisi l'approche a base de produit scalaire car il cache le nombre de uns. La thèse présente un aperçu des différents protocoles à base de produits scalaires dans la littérature ainsi que des protocoles d'intersection d'ensembles. Nous allons utiliser le protocole à base de produit scalaire présenté dans [74, 75]. A des fin d'illustration, nous appelons le premier pair Alice et le second pair Bob. Alice crypte chaque bit dans son vecteur binaire et les envoie à Bob, qui choisit uniquement les bits chiffrés d'Alice dont la position dans son propre vecteur binaire est 1. Bob ajoute les textes clairs choisis, en utilisant la propriété homomorphes de la cryptographie afin d'obtenir ainsi un cryptage du produit scalaire. Puis Alice et Bob utilisent la porte de multiplication de [51] pour obtenir une clé de cryptage du produit scalaire au carré (soit, en multipliant le produit scalaire par lui-même). Le produit scalaire au carré est le numérateur de la similarité cosinus carré. Le dénominateur est calculé séparément lorsque Alice envoie à Bob une valeur chiffrée du nombre d'éléments de son profil, puis Bob scalarise cette valeur chiffrée en utilisant le nombre d'éléments dans son profil. La valeur de seuil est alors également scindée en un numérateur et un dénominateur (ce qui est toujours possible puisque le seuil est représenté avec arithmétique à précision finie). L'affirmation selon laquelle la similarité cosinus carré est supérieur au seuil pourrait alors être reformulée en faisant

multipliant les deux dénominateurs des deux côtés de l'autre côté. Cela peut être fait aussi par des opérations de scalarisation parce que le seuil est un paramètre public et n'est pas crypté. Ensuite, le protocole de comparaison de nombre entier de [49] est utilisé pour obtenir le bit de sortie indiquant si la similarité cosinus carré est supérieure au seuil. Dans la thèse, nous examinons également les protocoles de comparaison d'entiers sécurisés dans la littérature. Dans la pratique, la valeur du seuil τ dépend de l'application et est fixé de façon empirique de manière à être nettement supérieur à la similarité moyenne dans la population. Néanmoins, nous proposons une euristique pour la sélection du seuil en fonction du taux d'acceptation voulue, en utilisant un modèle statistique pour le produit scalaire de deux vecteurs binaires.

La génération de bruit distribué pour la differential privacy. La differential privacy nécessite de l'aléatoire pour fonctionner, et ce caractère aléatoire doit rester secret pour le parti qui reçoit la sortie differentially private. Sinon, le bruit pourrait être enlevé et la garantie de differential privacy serait violée. Par conséquent, étant donné que les deux pairs impliqués dans le calcul à deux parties sont aussi ceux recevant sa sortie, alors aucun d'eux ne devrait connaitre l'aléatoire utilisé par le mécanisme. Plus précisément, la tâche de la génération du bruit ne peut être déléguée à l'un d'eux, ils doivent s'engager dans un protocole interactif pour générer du bruit. En outre, le bruit qu'ils engendrent en collaboration doit être généré sous une forme cryptée.

Les protocoles utilisés pour produire des pièces aléatoires sont connus comme des protocoles pile ou face. On sait que dans le cas des deux partis, un adversaire malveillant peut biaiser la pièce avec un additif $\Theta(1/r)$, dans lequel r est le nombre de tours du protocole pile ou face [66]. Comme nous supposons un adversaire passif dans cette thèse, nous sommes en mesure de produire des pièces non biaisées, sinon nous pourrions utiliser le protocole pile ou face optimal des deux partis de [66] pour générer des pièces biaisées et ensuite utiliser le mécanisme de differential privacy de [67] qui peut assurer la differential privacy, même pour des pièces non biaisés.

Il existe plusieurs protocoles pile ou face pour le cas multi-partis (majorité honnête) [51, 25, 50, 68], qui ne s'appliquent pas nécessairement au cas à deux parties de cette thèse. Ainsi, nous ne mentionnons que le protocole multi-parties de [25] (Cadre ODO) qui est conçu pour générer un bruit spécifique pour la differential privacy. ODO n'est toujours pas nécessairement applicable à la configuration en deux partis de cette thèse. Le framework ODO peut générer des variables aléatoires privées Binomiales et de Poisson, approximant respectivement les bruits Gaussiens et Laplaciens, respectivement. Toutefois, le bruit qu'ils génèrent dans une variante détendue de l'ε-differential privacy appelé (ε, δ)-differential privacy ce qui est une définition strictement plus faible [69]. Pour fournir ε-differential privacy à la place, les réponses aux requêtes doivent être borné. Nous décrivons un protocole de génération de bruit des deux parties semi-honnête qui satisfait ε-differential privacy sans borner les réponses des requêtes. En outre, au lieu de générer un bruit conjointement comme [25], dans notre protocole, chaque partie va générer du bruit au niveau local et l'intégrer avec le bruit de l'autre partie par addition, ce qui rend l'étape de génération de bruit plus efficace, mais est sécurisée seulement contre un adversaire passif.

Au lieu de dépendre d'un tiers pour générer du bruit, nous offrons deux possibilités de génération de bruit distribué. Le premier utilise deux variables aléatoires exponentielles. Cette méthode utilise l'observation que la différence de deux variable aléatoire exponentielle est une variable aléatoire de Laplace. Alice et Bob genèrent (chacun de leur côté) deux variables aléatoires exponentielles n_1 et n_2. Ensuite, pendant le protocole, Alice ajoute n_1 à la similarité cosinus carré cryptée tandis que Bob soustrait n_2. Si la valeur de similarité est révélée, un pair honnête mais curieux peut retirer son bruit exponentiel de la valeur révélée. Par exemple, si la valeur de sortie était $x = s + n_1 - n_2$, où s est la vraie similarité, le pair qui a généré n_2 pourrait le retirer de x une fois qu'il l'a reçue. La valeur avec laquelle il termine est $x + n_2 = s + n_1$ qui satisfait une variante plus faible de la differential privacy appelée (ε, δ)-differential privacy [25], en conséquence cette méthode est seulement recommandée pour la similarité par seuil dans laquelle la valeur de similarité perturbé n'est pas révélé. La deuxième méthode consiste à utiliser deux variables aléatoires de Laplace. Supposons que Alice et Bob veulent publier le résultat du produit scalaire entre les deux profils. A la fin du protocole Alice et Bob pourraient tous les deux simplement ajouter du bruit de Laplace aléatoire généré indépendamment en utilisant la propriété homomorphe du schéma de chiffrement. Ensuite, ils pourraient coopérer pour effectuer le décryptage de seuil et ils seraient tous deux arrivés à apprendre le produit intérieur perturbé. Ensuite, Alice peut soustraire son propre bruit de la sortie libérée pour récupérer une version du produit intérieur qui ont été perturbés qu'avec le bruit de Bob (qui elle ne peut pas supprimer).

L'impact de la différential privacy au protocole de similarité avec seuil, telle que mesurée par les faux positifs et de faux négatifs est également analysé dans la thèse avec un modèle théorique. Le modèle est également validé par l'évaluation expérimentale.

Differential privacy hétérogène. Nous proposons une variante de la differential privacy, appelé differential privacy hétérogène, qui peut fournir diverses garanties de confidentialité pour les différents pairs avec des attentes différentes de confidentialité. Par ailleurs, un seul utilisateur peut choisir différents niveaux de confidentialité pour différents articles dans son profil. Nous décrivons également un nouveau mécanisme, appelé "mécanisme d'étirement" et prouvons formellement qu'il satisfait la différential privacy hétérogène. Chaque élément du profil est associé à un poids de confidentialité entre 0 et 1 décrivant sensibilité, 0 étant le niveau de confidentialité le plus élevé. Le poids de confidentialité doit également être protégé de peur que leur magnitude ne révèle des informations à propos des items dont ils décrivent la sensibilité. Nous prouvons que notre mécanisme d'étirement fournit la differential privacy classique pour ces poids de confidentialité. Notre mécanisme d'étirement modifie la sensibilité de la fonction par l'application d'une transformation linéaire de l'entrée, avant d'appliquer le mécanisme de Laplace standard. Étant donné que le mécanisme modifie la fonction, de la distorsion est introduite à la sortie. Nous dérivons une borne sur la distorsion additionnelle et montrons qu'elle est de $\widetilde{O}(\sqrt{n})$ lorsque le poids de la vie privée est $1 - \widetilde{O}(1/n^{3/2})$, pour la fonction de produit scalaire.

Nous avons effectué une évaluation expérimentale et démontré que la differential privacy hétérogène peut assurer des niveaux différents de privacy tout en maintenant

un bon niveau de service. Nous considérons également que si les pairs sont divisés en trois groupes avec des attentes différentes de la vie privée : un groupe qui est pointilleux sur sa vie privée (fondamentalistes), un groupe qui est moins pointilleux mais attend tout de même un peu de vie privée (pragmatiques), et un groupe qui est plus indifférent, et d'étudier comment les variations dans les proportions de ces groupes les uns aux autres affectent les garanties d'utilité de chacun. Par exemple, comment serait affectée l'utilité fournie au groupe des indifférents si le groupe fondamentaliste formait la majorité des pairs dans le système. Nous concluons de nos expériences que (1) les pragmatiques et les indifférents ont toujours un meilleur rappel que les fondamentalistes et (2) les indifférents ont souvent un meilleur rappel que les pragmatiques, mais pas toujours. Cela semble indiquer que le groupe se souciant le plus de la vie privée est en général puni plus (c'est-à-dire, son utilité est faible) que les groupes qui sont plus ouvert en ce qui concerne les attentes de la vie privée. Ce n'est pas vraiment surprenant car un faible poids de vie privée se traduira par les utilisateurs du groupe de fondamentalistes se séparant des autres utilisateurs dans le regroupement, au point qu'ils n'auront pas nécessairement de voisins significatifs à leurs yeux. Enfin, à la question de savoir si (ou non) des groupes plus libéraux seront punis par des groupes conservateurs, la réponse semble être négative. En effet, il peut être vu à partir des résultats de ces expériences que les groupes conservateurs sont punis plus que les groupes libéraux.

Bien que le mécanisme d'étirement peut être appliquée à une multitude de fonctions, il n'est cependant pas directement applicable à certaines fonctions naturelles, telles que le ℓ_0-norme ou le minimum. En effet, lors du calcul de la norme ℓ_0 (soit le nombre de coordonnées non nulles dans un vecteur donné), chaque coordonnée contribue zéro ou un peu de sa valeur. Comme le mécanisme d'étirement modifie cette valeur, ce mécanisme retournerait toujours la vraie valeur tant qu'aucun poids de confidentialité n'a été mis exactement à zéro. Pour le cas du minimum, en raison du fait que le mécanisme d'étirage rétrécit chaque coordonnée par un facteur correspondant à son poids de confidentialité, la sortie résultante peut ne plus avoir aucun rapport à la sémantique de la fonction minimum.

Un autre défi est de permettre aux utilisateurs d'estimer la quantité de distorsion dans la sortie qu'ils ont reçu d'un mécanisme de differential privacy hétérogène. Par exemple, pour des fonctions telles que la somme, les bénéficiaires ne seront pas en mesure d'estimer la valeur correcte sans recevoir la distorsion. Bien que la distorsion ait une limite supérieure, les informations nécessaires pour calculer cette limite supérieure sont privés. Par conséquent, publier la distorsion (ou même sa borne supérieure) constituerait une violation de la vie privée. Nous pensons que ce problème pourrait être résolu partiellement en libérant une limite supérieure en utilisant le mécanisme de Laplace traditionnel avec un cout supplémentaire de ε montant de vie privée.

Un important travail futur comprend la caractérisation des fonctions qui ont une distorsion basse et haute quand le mécanisme d'étirage est appliquée sur eux. En effet, les fonctions comportant une forte distorsion ne sont pas vraiment adaptés à notre mécanisme d'étirement. Nous laissons aussi ouverte la question de savoir s'il est possible d'atteindre la differential privacy hétérogène avec une distorsion inférieure à celle du mécanisme d'étirement ou sans distorsion du tout.

BLIP : calcul de similarité non-interactif pour filtres de Bloom. Nous décrivons un mécanisme de differential privacy non-interactif pour le calcul de similarité en pair à pair. Etant non-interactif, il permet d'éviter la question du budget de la vie privée, qui impose une limite sur le nombre de fois que la similarité peut être calculé, car le mécanisme est calculée une seule fois. Par conséquent, le mécanisme évite la nécessité du canal anonyme bidirectionnel, fournissant des calcul de similarité plus efficace en termes de cout de calcul et de communication, car il n'est pas nécessaire d'utiliser des outils cryptographiques, et aussi que sa sortie peut être mis en cache. Un autre avantage de ce mécanisme non-interactif est que le calcul de similarité peut avoir lieu même lorsque l'utilisateur est hors ligne, ce qui est impossible à réaliser avec des mécanismes interactifs. Notre nouveau mécanisme de protection des renseignements personnels est appelé BLIP (pour BLoom-and-flIP). Pour s'attaquer simultanément aux problèmes de confidentialité et d'évolutivité, nous proposons BLIP (Bloom-and-FLIP), un mécanisme de differential privacy non-interactif, qui calcule un filtre de Bloom standard [109] du profil d'un pair, puis le perturbe avant sa diffusion publique en vue d'assurer de hautes garanties de confidentialité. Un filtre de Bloom est une structure de donnés compacte de taille fixe permettant de représenter des ensembles, économisant de l'espace contre un taux de faux positifs. Ce filtre de Bloom aléatoire peut être utilisé un nombre illimité de fois pour estimer la similarité d'un pair avec un autre profil sans violer la vie privée des profils. En outre, cette approche a exactement le même cout de communication que les simples (c'est-à-dire, non perturbé) filtres de Bloom, tout en offrant des garanties de confidentialité beaucoup plus élevés, mais au prix d'une légère diminution de l'utilité. Par exemple, le filtre de Bloom differentially private peut être utilisé pour calculer une mesure de similarité, par exemple en similarité cosinus ou en produit scalaire avec un autre filtre de Bloom de façon non interactive.

Nous fournissons une analyse théorique du compromis entre la vie privée et d'utilité dans la forme de l'erreur d'estimation de la similarité pour un paramètre de confidentialité donnée. En particulier, nous montrons que l'erreur ajoutée au produit scalaire entre deux filtres de Bloom est délimitée par $\sqrt{m}$, dans lequel m est la taille du filtre de Bloom, avec une probabilité d'au moins $1 - 2\exp(-2\tanh(\varepsilon/2k)^2)$, dans lequel k est le nombre de fonctions de hachage utilisées pour le filtre de Bloom. En outre, nous évaluons expérimentalement le compromis entre la vie privée et l'utilité qui peut être atteint avec BLIP et montront qu'il offre un bon niveau d'utilité.

Comme travaux futurs, il serait intéressant d'étudier les variations qui offrent une plus grande précision sans pour autant sacrifier la vie privée ou l'efficacité.

Choisir le paramètre de confidentialité. Habituellement, la valeur du paramètre de confidentialité de differential privacy ε est difficile à choisir, et être capable de choisir une valeur appropriée pour ce paramètre est toujours une question ouverte, qui n'a pas vraiment été étudiée, à quelques exceptions près [39, 28]. Nous abordons ce problème en fournissant un moyen de choisir une valeur pour le paramètre de confidentialité ε pour BLIP, notre protocole de non-interactive, en termes d'utilité et de vie privée. Alors que l'utilité est mesurée par la borne théorique décrite plus tôt, la vie privée est analysée au travers de deux attaques. La valeur de ε devrait être choisi de telle sorte qu'elle présente une utilité acceptable tout en évitant ces deux

attaques. En particulier, en utilisant ces attaques nous fournissons expérimentalement une borne supérieure au paramètre de vie privée afin d'empêcher ces attaques. La justification est de garantir que le mécanisme n'est au moins pas ouvertement non-privé [133]. Nous faisons cela en proposant deux attaques où chacun essaie de créer une brèche différente de la vie privée, et aucune des deux ne suppose la disponibilité d'informations auxiliaires. Par ailleurs, pour simplifier les choses, nous ne supposons pas que des corrélations existent entre les éléments (les corrélations sont traitées dans une autre attaque plus tard). Par conséquent, la valeur de ε pour lesquels nos attaques réussissent est simplement une limite supérieure indicative, qui est toujours considéré comme un progrès, étant donné que nous pouvons avoir très peu d'indices autrement. Les attaques sont des attaques d'inférence probabiliste. La première attaque est appelée l'"Attaque par reconstruction de profil" car elle peut être utilisé pour reconstruire le profil d'un pair a partir de sa représentation en filtre de Bloom perturbé. L'autre attaque est appelé le "Jeu de distinction de profil", dans Lequel l'adversaire tente de distinguer deux profils différents sur un seul point, en observant les filtres de Bloom perturbé de deux pairs. Plus précisément, nous proposons une analyse de la protection et de l'utilité offert par BLIP contre ces attaques, en dérivant une borne supérieure a la valeur requise pour le paramètre de differential privacy ε qui garantit que les deux attaques échouent. Nous avons démontré que le mécanisme de BLIP est capable de garantir la confidentialité du profil sous-jacent à la valeur recommandée pour le paramètre de confidentialité tout en offrant de l'utilité.

Attaques d'inférence. Kifer and Machanavajjhala [84] ont soulevé la préoccupation de l'existence de dépendances entre les bits d'un profil lors de l'examen des garanties prévues par la differential privacy pour les vecteurs binaires. En particulier, ils ont montré un exemple artificiel de corrélations permettant à un adversaire de reconstruire entièrement les données privées étant donné la réponse à requête de façon differentially private. La vraie question qui sous-tend n'est pas la présence de corrélations en soi, mais plutôt que le modèle d'adversaire habituellement assumée par la differential privacy est trop fort. Par "fort", nous voulons dire que l'adversaire est supposé en savoir trop sur les données privées. Leur exemple ne réussit à violer la vie privée que pour un adversaire faible, mais pas pour le fort pris par la définition de la differential privacy. De manière informelle, une violation de la vie privée se produit quand un adversaire gagne quantité non négligeable d'information après avoir observé la réponse à une requête préservant la vie privée. Si l'adversaire est si fort qu'il sait déjà tout sur les données privées, alors même si l'ensemble des données a été libéré aucune violation de la vie privée ne se produit car l'adversaire n'a pas gagné de nouvelle information. Toutefois, si l'adversaire se révèle en fait être plus faible, alors une une violation de la vie privée peut avoir lieu. Par analogie, en differential privacy l'adversaire est censé connaitre tous les bits sauf un (c'est à dire tous les bits dans un vecteur binaire sauf un), alors qu'en réalité l'adversaire est beaucoup plus faible. Par rapport à l'hypothèse selon laquelle l'adversaire connait chaque bit sauf un, alors si l'adversaire finit par connaitre tous les bits sauf un après la publication, aucune violation de la vie privée ne se produit. Toutefois, le fait qu'une violation de la vie privée se produise réellement ou pas est relatif à ce que l'adversaire savait avant

la publication. Ainsi, si l'adversaire ne savait rien, mais a fini par apprendre chaque bit, sauf un, une violation de la vie privée se produit, même si les conditions de differential privacy sur le mécanisme de publication seul tient toujours. La capacité de l'adversaire qui ne connait aucun bits (un adversaire aveugle) à apprendre des bits qu'il ne pouvait pas apprendre avant la publication dépend de la connaissance auxiliaire dont il dispose. Nous supposons un type de connaissance auxiliaire qui est inutile en soit même (c'est-à-dire, avant libération), mais qu'il est néanmoins réaliste de supposer que l'adversaire a, qui sont les corrélations entre les bits. Bien que les corrélations ne soient pas nécessaires pour cette violation de vie privée se produise, elles agissent comme un amplificateur qui permet d'obtenir un taux de réussite plus élevé. Nous employons des corrélations entre les bits afin de regrouper en points de repères. Ensuite, nous étudions deux attaques qui utilisent ces points de repères pour reconstituer les vecteurs binaires étant donné les réponses differentially private a quelques requêtes définies par ces points de repère. Nous montons nos attaques contre deux mécanismes non interactifs differentially private, BLIP, que nous avons introduit plus tôt, et le mécanisme Johnson-Lindenstrauss [45]. Les mécanismes produisent des réponses perturbées aux requêtes des produits scalaires. Nos attaques emploient le produit scalaire et est agnostique au mécanisme sous-jacent.

Nous évaluons nos attaques sur un jeu de données réelles et une version allégée du même ensemble de données, mais sans corrélations. Nous concluons qu'un adversaire qui connait les corrélations entre les bits a un avantage sur un adversaire aveugle. Un tel adversaire est capable de reconstruire une partie importante du profil, même lorsqu'un adversaire relativement plus fort ne peut pas deviner le seul bit qui lui est inconnu. Nous observons également qu'un adversaire boite blanche, ciblant le fonctionnement d'un mécanisme particulier, fait mieux qu'un adversaire boite noire, étant agnostique au mécanisme, donc comprendre ce qu'il est possible dans les attaques boite blanche par rapport aux attaques boites noires est un axe de recherche intéressant.

Bibliography

[1] M. Bertier, D. Frey, R. Guerraoui, A.-M. Kermarrec, and V. Leroy, "The Gossple Anonymous Social Network," in Proceedings of the 11th International Middleware Conference (Middleware'10), ACM/IFIP/USENIX, Bangalore, India, November 2010, pp. 191–211.

[2] Y. Zeng, N. Zhong, X. Ren, and Y. Wang, "User Interests Driven Web Personalization Based on Multiple Social Networks," in International Workshop on Web Intelligence & Communities, (WI&C'12), R. Akerkar, P. Maret, and L. Vercouter, Eds. Lyon, France: ACM, April 2012, pp. 9:1–9:4.

[3] X. Zhou, Y. Xu, Y. Li, A. Josang, and C. Cox, "The State-of-the-Art in Personalized Recommender Systems for Social Networking," Artificial Intelligence Review, vol. 37, no. 2, pp. 119–132, 2012.

[4] Z. Wen and C.-Y. Lin, "How Accurately Can One's Interests Be Inferred from Friends?" in Proceedings of the 19th International Conference on World Wide Web (WWW'10), M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, Eds. Raleigh, North Carolina, USA: ACM, April 2010, pp. 1203–1204.

[5] F. Liu, C. Yu, and W. Meng, "Personalized Web Search for Improving Retrieval Effectiveness," IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 1, pp. 28–40, January 2004.

[6] G. Greenwald and E. MacAskill, "NSA Prism program taps in to user data of Apple, Google and others," The Guardian, June 2013. [Online]. Available: http://www.guardian.co.uk/world/2013/jun/06/us-tech-giants-nsa-data

[7] P. World, "Facebook's Beacon more intrusive than previously thought," November 2007. [Online]. Available: http://www.pcworld.com/article/140182/article.html

[8] H. D'Andrade, "MySpace and Facebook plan to use personal data for "targeted advertising"," September 2007. [Online]. Available: https://www.eff.org/deeplinks/2007/09/myspace-and-facebook-plan-use-personal-data-targeted-advertising

[9] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov, ""You Might Also Like:" Privacy Risks of Collaborative Filtering," in 32nd IEEE Symposium on Security and Privacy, S&P 2011. Berkeley, California, USA: IEEE Computer Society, May 2011, pp. 231–246.

[10] E. Toch, Y. Wang, and L. Cranor, "Personalization and Privacy: a Survey of Privacy Risks and Remedies in Personalization-Based Systems," User Modeling and User-Adapted Interaction, vol. 22, no. 1-2, pp. 203–220, 2012, 10.1007/s11257-011-9110-z.

[11] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," in 29th IEEE Symposium on Security and Privacy, S&P 2008. Oakland, California, USA: IEEE Computer Society, May 2008, pp. 111–125.

[12] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "Gossip-Based Peer Sampling," ACM Transactions on Computer Systes (TOCS'07), vol. 25, no. 3, August 2007.

[13] V. Leroy, "Distributing social applications," Ph.D. dissertation, IRISA, Decemter 2010.

[14] A.-M. Kermarrec, "Towards a Personalized Internet: a Case for a Full Decentralization." Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences, vol. 371, no. 1987, March 2013.

[15] X. Bai, R. Guerraoui, A.-M. Kermarrec, and V. Leroy, "Collaborative personalized top-k processing," ACM Transactions on Database Systems, vol. 36, no. 4, p. 26, 2011.

[16] A. Boutet, D. Frey, R. Guerraoui, A. Jégou, and A.-M. Kermarrec, "WhatsUp Decentralized Instant News Recommender," in Proceedings of the 27th IEEE International Parallel & Distributed Processing Symposium (IPDPS'13). Boston, Massachusetts, USA: IEEE Computer Society, May 2013, pp. 741–752.

[17] D. Frey, A. Jégou, and A.-M. Kermarrec, "Social Market: Combining Explicit and Implicit Social Networks," in Proceedings of the 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'11), ser. Lecture Notes in Computer Science, X. Défago, F. Petit, and V. Villain, Eds., vol. 6976. Grenoble, France: Springer, October 2011, pp. 193–207.

[18] A. C.-C. Yao, "Protocols for Secure Computations (Extended Abstract)," in Proceedings of 23rd Annual Symposium on Foundations of Computer Science (FOCS'82). Chicago, Illinois, USA: IEEE Computer Society, November 1982, pp. 160–164.

[19] O. Goldreich, "Cryptography and cryptographic protocols," Distributed Computing, vol. 16, no. 2–3, pp. 177–199, 2003.

[20] C. C. Aggarwal and P. S. Yu, Eds., Privacy-Preserving Data Mining: Models and Algorithms, ser. Advances in Database Systems. Springer US, 2008, vol. 34.

[21] F. McSherry and I. Mironov, "Differentially Private Recommender Systems: Building Privacy into the Netflix Prize Contenders," in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and

Data Mining (KDD'09), J. F. E. IV, F. Fogelman-Soulié, P. A. Flach, and M. J. Zaki, Eds. Paris, France: ACM, June 2009, pp. 627–636.

[22] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)," in Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC'88), J. Simon, Ed. Chicago, Illinois, USA: ACM, May 1988, pp. 1–10.

[23] O. Goldreich, S. Micali, and A. Wigderson, "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority," in Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC'87), A. V. Aho, Ed. New York, New York, USA: ACM, 1987, pp. 218–229.

[24] D. Chaum, C. Crépeau, and I. Damgård, "Multiparty Unconditionally Secure Protocols (Extended Abstract)," in Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC'88), J. Simon, Ed. Chicago, Illinois, USA: ACM, May 1988, pp. 11–19.

[25] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our Data, Ourselves: Privacy via Distributed Noise Generation," in Proceedings of the 25th International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT'06), ser. Lecture Notes in Computer Science, S. Vaudenay, Ed., vol. 4004. St. Petersburg, Russia: Springer, May 2006, pp. 486–503.

[26] A. Beimel, K. Nissim, and E. Omri, "Distributed Private Data Analysis: on Simultaneously Solving How and What," in Proceedings of the 28th Annual International Cryptology Conference - Advances in Cryptology (CRYPTO'08), ser. Lecture Notes in Computer Science, D. Wagner, Ed., vol. 5157. Santa Barbara, CA, USA: Springer, August 2008, pp. 451–468.

[27] C. Dwork and M. Naor, "On the difficulties of disclosure prevention in statistical databases or the case for differential privacy," Journal of Privacy and Confidentiality, vol. 2, no. 1, pp. 93–107, 2010.

[28] M. S. Alvim, M. E. Andrés, K. Chatzikokolakis, and C. Palamidessi, "On the Relation Between Differential Privacy and Quantitative Information Flow," in Part II of the Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP'11), ser. Lecture Notes in Computer Science, L. Aceto, M. Henzinger, and J. Sgall, Eds., vol. 6756. Zurich, Switzerland: Springer, July 2011, pp. 60–76.

[29] T. Dalenius, "Towards a Methodology for Statistical Disclosure Control," Statistik Tidskrift, vol. 15, pp. 429–444, 1977.

[30] I. Dinur and K. Nissim, "Revealing Information while Preserving Privcy," in Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'03), F. Neven, C. Beeri, and T. Milo, Eds. San Diego, California, USA: ACM, June 2003, pp. 202–210.

[31] K. Chatzikokolakis, M. E. Andrés, N. E. Bordenabe, and C. Palamidessi, "Broadening the Scope of Differential Privacy Using Metrics," in Proceedings of the 13th International Symposium on Privacy Enhancing Technologies (PETS'13), ser. Lecture Notes in Computer Science, E. D. Cristofaro and M. Wright, Eds., vol. 7981. Bloomington, IN, USA: Springer, July 2013, pp. 82–102.

[32] A. Moin, "Recommendation and Visualization Techniques for Large Scale Data," Ph.D. dissertation, Université Rennes 1, July 2012.

[33] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: Byzantine Resilient Random Membership Sampling," Computer Networks, vol. 53, no. 13, pp. 2340–2359, 2009.

[34] A.-M. Kermarrec and M. van Steen, "Gossiping in distributed systems," Operating Systems Review, vol. 41, no. 5, pp. 2–7, 2007.

[35] J. Imbrie and E. G. Purdy, "Classification of Modern Bahamian Carbonate Sediments," Classification of Carbonate Rocks, a Symposium, pp. 253–272, 1962.

[36] S.-S. Choi, S.-H. Cha, and C. C. Tappert, "A Survey of Binary Similarity and Distance Measures," Journal on Systemics, Cybernetics and Informatics, vol. 8, no. 1, pp. 43–48, 2010.

[37] C. Dwork, "Differential Privacy," in Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP'06), Part II, ser. Lecture Notes in Computer Science, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds., vol. 4052. San Servolo, Venice, Italy: Springer, July 2006, pp. 1–12.

[38] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in Proceedings of the 3rd Theory of Cryptography Conference (TCC'06), ser. Lecture Notes in Computer Science, S. Halevi and T. Rabin, Eds., vol. 3876. New York, NY, USA: Springer, March 2006, pp. 265–284.

[39] J. Lee and C. Clifton, "How Much is Enough? Choosing ε for Differential Privacy," in Proceedings of the 14th International Information Security Conference (ISC'11), ser. Lecture Notes in Computer Science, X. Lai, J. Zhou, and H. Li, Eds., vol. 7001. Xi'an, China: Springer, October 2011, pp. 325–340.

[40] M. Alaggan, S. Gambs, and A.-M. Kermarrec, "BLIP: Non-Interactive Differentially-Private Similarity Computation on Bloom Filters," in Proceedings of the 14th International Symposium on Stabilization, Safety, and Securityof Distributed Systems (SSS'12), ser. Lecture Notes in Computer Science, A. W. Richa and C. Scheideler, Eds., vol. 7596. Toronto, Canada: Springer, October 2012, pp. 202–216.

[41] P. Dandekar, N. Fawaz, and S. Ioannidis, "Privacy Auctions for Inner Product Disclosures," CoRR, vol. abs/1111.2885, 2011.

[42] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What Can We Learn Privately?" in Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS'08). Philadelphia, Pennsylvania, USA: IEEE Computer Society, October 2008, pp. 531–540.

[43] F. D. McSherry, "Privacy Integrated Queries: an Extensible Platform for Privacy-Preserving Data Analysis," in Proceedings of the ACM SIGMOD International Conference on Management of Data, (SIGMOD'09), U. Çetintemel, S. B. Zdonik, D. Kossmann, and N. Tatbul, Eds. Providence, Rhode Island, USA: ACM, June 2009, pp. 19–30.

[44] D. Leoni, "Non-Interactive Differential Privacy: a Survey," in Proceedings of the 1st International Workshop on Open Data (WOD'12), G. Raschia and M. Theobald, Eds. Nantes, France: ACM, May 2012, pp. 40–52.

[45] K. Kenthapadi, A. Korolova, I. Mironov, and N. Mishra, "Privacy via the Johnson-Lindenstrauss Transform," CoRR, vol. abs/1204.2606, 2012.

[46] O. Goldreich, Foundations of Cryptography: Volume 2, Basic Applications. New York, NY, USA: Cambridge University Press, 2004.

[47] A. Shamir, "How to share a secret," Communications of the ACM, vol. 22, no. 11, pp. 612–613, 1979.

[48] J. A. Garay, B. Schoenmakers, and J. Villegas, "Practical and Secure Solutions for Integer Comparison," in Problem of the 10th International Conference on Practice and Theory in Public-Key Cryptography (PKC'07), ser. Lecture Notes in Computer Science, T. Okamoto and X. Wang, Eds., vol. 4450. Beijing, China: Springer, April 2007, pp. 330–342.

[49] T. Nishide and K. Ohta, "Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol," in Problem of the 10th International Conference on Practice and Theory in Public-Key Cryptography (PKC'07), ser. Lecture Notes in Computer Science, T. Okamoto and X. Wang, Eds., vol. 4450. Beijing, China: Springer, April 2007, pp. 343–360.

[50] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft, "Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation," in Proceedings of the 3rd Theory of Cryptography Conference (TCC'06), ser. Lecture Notes in Computer Science, S. Halevi and T. Rabin, Eds., vol. 3876. New York, NY, USA: Springer, March 2006, pp. 285–304.

[51] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty Computation from Threshold Homomorphic Encryption," in Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, Advances

in Cryptology (EUROCRYPT'01), ser. Lecture Notes in Computer Science, B. Pfitzmann, Ed., vol. 2045. Innsbruck, Austria: Springer, May 2001, pp. 280–299.

[52] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT'99), ser. Lecture Notes in Computer Science, J. Stern, Ed., vol. 1592. Prague, Czech Republic: Springer, May 1999, pp. 223–238.

[53] I. Damgård and M. Jurik, "A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System," in Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography (PKC'01), ser. Lecture Notes in Computer Science, K. Kim, Ed., vol. 1992. Cheju Island, Korea: Springer, February 2001, pp. 119–136.

[54] O. Goldreich, Foundations of Cryptography: Basic Tools. New York, NY, USA: Cambridge University Press, 2000.

[55] J. R. Douceur, "The Sybil Attack," in Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), ser. Lecture Notes in Computer Science, P. Druschel, M. F. Kaashoek, and A. I. T. Rowstron, Eds., vol. 2429. Cambridge, Massachusetts, USA: Springer, March 2002, pp. 251–260.

[56] A. Pfitzmann and M. Waidner, "Networks without User Observability: Design Options," in Proceedings of the Workshop on the Theory and Application of of Cryptographic Techniques, Advances in Cryptology - (EUROCRYPT'85), ser. Lecture Notes in Computer Science, F. Pichler, Ed. Linz, Austria: Springer, April 1985, vol. 219, pp. 245–253.

[57] A. Pfitzmann and M. Köhntopp, "Anonymity, Unobservability, and Pseudonymity – A Proposal for Terminology," in Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Designing Privacy Enhancing Technologies, ser. Lecture Notes in Computer Science, H. Federrath, Ed., vol. 2009. Berkeley, CA, USA: Springer, July 2000, pp. 1–9.

[58] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed. New York, NY, USA: John Wiley & Sons, Inc., 1995.

[59] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The Second-Generation Onion Router," in Proceedings of the 13th USENIX Security Symposium (USENIX'04). San Diego, CA, USA: USENIX, August 2004, pp. 303–320.

[60] G. Danezis, C. Díaz, E. Käsper, and C. Troncoso, "The Wisdom of Crowds: Attacks and Optimal Constructions," in Proceedings of the 14th European Symposium on Research in Computer Security (ESORICS'09), ser. Lecture Notes in Computer Science, M. Backes and P. Ning, Eds., vol. 5789. Saint-Malo, France: Springer, September 2009, pp. 406–423.

[61] M. Shaneck, Y. Kim, and V. Kumar, "Privacy Preserving Nearest Neighbor Search," in Proceedings of the 6th IEEE International Conference on Data Mining (ICDM'06). Hong Kong, China: IEEE Computer Society, December 2006, pp. 541–545.

[62] F. McSherry and I. Mironov, "Differentially Private Recommender Systems: Building Privacy into the Netflix Prize Contenders," in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09), J. F. E. IV, F. Fogelman-Soulié, P. A. Flach, and M. J. Zaki, Eds. Paris, France: ACM, June 2009, pp. 627–636.

[63] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," Communications of the ACM, vol. 24, no. 2, pp. 84–88, February 1981.

[64] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach, "AP3: Cooperative, Decentralized Anonymous Communication," in Proceedings of the 11st ACM SIGOPS European Workshop, Y. Berbers and M. Castro, Eds. Leuven, Belgium: ACM, September 2004.

[65] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," ACM Trans. Inf. Syst. Secur., vol. 1, no. 1, pp. 66–92, 1998.

[66] T. Moran, M. Naor, and G. Segev, "An Optimally Fair Coin Toss," in Proceedings of the 6th IACR Theory of Cryptography Conference (TCC'09), ser. Lecture Notes in Computer Science, O. Reingold, Ed., vol. 5444. San Francisco, California, USA: Springer, March 2009, pp. 1–18.

[67] Y. Dodis, A. López-Alt, I. Mironov, and S. Vadhan, "Differential Privacy with Imperfect Randomness," in Proceedings of the 32nd Annual Cryptology Conference – Advances in Cryptology (CRYPTO'12), ser. Lecture Notes in Computer Science, R. Safavi-Naini and R. Canetti, Eds., vol. 7417. Santa Barbara, California, USA: Springer, August 2012, pp. 497–516.

[68] M. O. Rabin, "Randomized Byzantine Generals," in Proceedings of the 24th Annual Symposium on Foundations of Computer Science (FOCS'83). Tucson, Arizona, USA: IEEE Computer Society, November 1983, pp. 403–409.

[69] A. De, "Lower Bounds in Differential Privacy," in Proceedings of the 9th Theory of Cryptography Conference (TCC'12), ser. Lecture Notes in Computer Science, R. Cramer, Ed., vol. 7194. Taormina, Sicily, Italy: Springer, March 2012, pp. 321–338.

[70] G. S. Narayanan, T. Aishwarya, A. Agrawal, A. Patra, A. Choudhary, and C. P. Rangan, "Multi Party Distributed Private Matching, Set Disjointness and Cardinality of Set Intersection with Information Theoretic Security," in Proceedings of the 8th International Conference on Cryptology and Network Security (CANS'09), ser. Lecture Notes in Computer Science, J. A. Garay, A. Miyaji, and A. Otsuka, Eds., vol. 5888. Kanazawa, Japan: Springer, December 2009, pp. 21–40.

[71] R. Li and C. Wu, "An Unconditionally Secure Protocol for Multi-Party Set Intersection," in Proceedings of the 5th International Conference on Applied Cryptography and Network Security (ACNS'07), ser. Lecture Notes in Computer Science, J. Katz and M. Yung, Eds., vol. 4521. Zhuhai, China: Springer, June 2007, pp. 226–236.

[72] L. Kissner and D. X. Song, "Privacy-preserving set operations," in Proceedings of the 25th Annual International Cryptology Conference, Advances in Cryptology (CRYPTO'05), ser. Lecture Notes in Computer Science, V. Shoup, Ed. Santa Barbara, California, USA: Springer, August 2005, vol. 3621, pp. 241–257.

[73] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient Private Matching and Set Intersection," in Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT'04), ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Interlaken, Switzerland: Springer, May 2004, pp. 1–19.

[74] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen, "On Private Scalar Product Computation for Privacy-Preserving Data Mining," in Revised Selected Papers from the 7th International Conference on Information Security and Cryptology (ICISC'04), ser. Lecture Notes in Computer Science, C. Park and S. Chee, Eds. Seoul, Korea: Springer, December 2004, vol. 3506, pp. 104–120.

[75] R. N. Wright and Z. Yang, "Privacy-Preserving Bayesian Network Structure Computation on Distributed Heterogeneous Data," in Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04), W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, Eds. Seattle, Washington, USA: ACM, August 2004, pp. 713–718.

[76] A. Amirbekyan and V. Estivill-Castro, "A New Efficient Privacy-Preserving Scalar Product Protocol," in Proceedings of the 6th Australasian Data Mining and Analytics Conference (AusDM'07), ser. CRPIT, P. Christen, P. J. Kennedy, J. Li, I. Kolyshkina, and G. J. Williams, Eds., vol. 70. Gold Coast, Queensland, Australia: Australian Computer Society, December 2007, pp. 209–214.

[77] C. A. Melchor, B. A. Salem, and P. Gaborit, "A Collusion-Resistant Distributed Scalar Product Protocol with Application to Privacy-Preserving Computation of Trust," in Proceedings of The 8th IEEE International Symposium on Networking Computing and Applications (NCA'09). Cambridge, Massachusetts, USA: IEEE Computer Society, July 2009, pp. 140–147.

[78] I.-C. Wang, C.-H. Shen, J. Zhan, T. sheng Hsu, C.-J. Liau, and D.-W. Wang, "Toward empirical aspects of secure scalar product," IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol. 39, no. 4, pp. 440–447, 2009.

[79] A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino, "Private record matching using differential privacy," in Proceedings of the 13th International Conference on Extending Database Technology (EDBT'10), ser. ACM International Conference Proceeding Series, I. Manolescu, S. Spaccapietra, J. Teubner, M. Kitsuregawa, A. Léger, F. Naumann, A. Ailamaki, and F. Özcan, Eds., vol. 426. Lausanne, Switzerland: ACM, March 2010, pp. 123–134.

[80] A. C.-C. Yao, "How to Generate and Exchange Secrets (Extended Abstract)," in Proceedings of the 27th Annual Symposium on Foundations of Computer Science (FOCS'86). Toronto, Canada: IEEE Computer Society, October 1986, pp. 162–167.

[81] H.-Y. Lin and W.-G. Tzeng, "An Efficient Solution to the Millionaires' Problem Based on Homomorphic Encryption," in Proceedings of the 3rd International Conference on Applied Cryptography and Network Security (ACNS'05), ser. Lecture Notes in Computer Science, J. Ioannidis, A. D. Keromytis, and M. Yung, Eds., vol. 3531, New York, NY, USA, June 2005, pp. 456–466.

[82] B. Schoenmakers and P. Tuyls, "Efficient Binary Conversion for Paillier Encrypted Values," in Proceedings of the 25th International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT'06), ser. Lecture Notes in Computer Science, S. Vaudenay, Ed., vol. 4004. St. Petersburg, Russia: Springer, May 2006, pp. 522–537.

[83] ——, "Practical Two-Party Computation Based on the Conditional Gate," in Proceedings of the 10th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT'04), ser. Lecture Notes in Computer Science, P. J. Lee, Ed., vol. 3329. Jeju Island, Korea: Springer, December 2004, pp. 119–136.

[84] D. Kifer and A. Machanavajjhala, "No Free Lunch In Data Privacy," in Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'11), T. K. Sellis, R. J. Miller, A. Kementsietsidis, and Y. Velegrakis, Eds. Athens, Greece: ACM, June 2011, pp. 193–204.

[85] P.-A. Fouque, J. Stern, and J.-G. Wackers, "CryptoComputing with Rationals," in Proceedings of the 6th International Conference on Financial Cryptography (FC'02), Revised Papers, ser. Lecture Notes in Computer Science, M. Blaze, Ed., vol. 2357. Southampton, Bermuda: Springer, March 2002, pp. 136–146.

[86] I. Mironov, "On Significance of the Least Significant Bits for Differential Privacy," in Proceedings of the ACM conference on Computer and Communications Security (CCS'12), T. Yu, G. Danezis, and V. D. Gligor, Eds. Raleigh, North Carolina, USA: ACM, October 2012, pp. 650–661.

[87] I. Gazeau, D. Miller, and C. Palamidessi, "Preserving differential privacy under finite-precision semantics," INRIA, Research Report, 2013. [Online]. Available: http://hal.inria.fr/hal-00780774

[88] J. Bar-Ilan and D. Beaver, "Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction," in Proceedings of the 8th Annual ACM Symposium on Principles of Distributed Computing (PODC'89), P. Rudnicki, Ed. Edmonton, Alberta, Canada: ACM, August 1989, pp. 201–209.

[89] W. L. Harkness, "Properties of the Extended Hypergeometric Distribution," The Annals of Mathematical Statistics, vol. 36, no. 3, pp. 938–945, 1965.

[90] M. Alaggan, S. Gambs, and A.-M. Kermarrec, "Private Similarity Computation in Distributed Systems: From Cryptography to Differential Privacy," in Proceedings of the 15th International Conference on the Principles of Distributed Systems (OPODIS'11), ser. Lecture Notes in Computer Science, A. F. Anta, G. Lipari, and M. Roy, Eds., vol. 7109. Toulouse, France: Springer, December 2011, pp. 357–377.

[91] S. Venkatasubramanian, Privacy-Preserving Data Mining, ser. Advances in Database Systems. Springer US, 2008, vol. 34, ch. Measures of Anonymity, pp. 81–103.

[92] C. Dwork, "Differential Privacy: a Survey of Results," in Proceedings of the 5th International Conference on Theory and Applications of Models of Computation (TAMC'08), ser. Lecture Notes in Computer Science, M. Agrawal, D.-Z. Du, Z. Duan, and A. Li, Eds., vol. 4978. Xi'an, China: Springer, April 2008, pp. 1–19.

[93] I. Mironov, O. Pandey, O. Reingold, and S. P. Vadhan, "Computational Differential Privacy," in Proceedings of the 29th Annual International Cryptology Conference – Advances in Cryptology (CRYPTO'09), ser. Lecture Notes in Computer Science, S. Halevi, Ed., vol. 5677. Santa Barbara, CA, USA: Springer, August 2009, pp. 126–142.

[94] A. McGregor, I. Mironov, T. Pitassi, O. Reingold, K. Talwar, and S. Vadhan, "The Limits of Two-Party Differential Privacy," Electronic Colloquium on Computational Complexity, Tech. Rep. 106, August 2011.

[95] S. Preibusch and A. R. Beresford, "Privacy-Preserving Friendship Relations for Mobile Social Networking," in Proceedings of the W3C Workshop on the Future of Social Networking, Barcelona, Spain, January 2009.

[96] Y. Liu, K. P. Gummadi, B. Krishnamurthy, and A. Mislove, "Analyzing Facebook Privacy Settings: User Expectations vs. Reality," in Proceedings of the Internet Measurement Conference, Berlin, Germany, November 2011, pp. 61–70.

[97] D. Zwick and N. Dholakia, "Models of Privacy in the Digital Age: Implications for Marketing and E-Commerce," September 1999.

[98] N. Zhang and W. Zhao, "Privacy-Preserving Data Mining Systems," Computer, vol. 40, no. 4, pp. 52–58, April 2007.

[99] A. Jeffrey, Matrix Operations for Engineers and Scientists: An Essential Guide in Linear Algebra. Springer Netherlands, 2010, ch. Linear Transformations and the Geometry of the Plane, pp. 239–272.

[100] K. Das, K. Bhaduri, and H. Kargupta, "Multi-Objective Optimization Based Privacy Preserving Distributed Data Mining in Peer-to-Peer Networks," Peer-to-Peer Networking and Applications, vol. 4, no. 2, pp. 192–209, 2011.

[101] R. Kumar, R. D. Gopal, and R. S. Garfinkel, "Freedom of Privacy: Anonymous Data Collection with Respondent-Defined Privacy Protection," INFORMS Journal on Computing, vol. 22, no. 3, pp. 471–481, 2010.

[102] L. Sweeney, "k-Anonymity: A Model for Protecting Privacy," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, no. 5, pp. 557–570, 2002.

[103] A. Ghosh and A. Roth, "Selling Privacy at Auction," in Proceedings of the 12th ACM Conference on Electronic Commerce (EC-2011), Y. Shoham, Y. Chen, and T. Roughgarden, Eds. San Jose, CA, USA: ACM, June 2011, pp. 199–208.

[104] K. Nissim, S. Raskhodnikova, and A. Smith, "Smooth Sensitivity and Sampling in Private Data Analysis," in Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC'07), D. S. Johnson and U. Feige, Eds. San Diego, California, USA: ACM, June 2007, pp. 75–84.

[105] H. Dym, Linear Algebra in Action. Weizmann Institute of Science - AMS, 2007.

[106] C. Jensen, C. Potts, and C. Jensen, "Privacy Practices of Internet Users: Self-Reports versus Observed Behavior," International Journal of Human-Computer Studies, vol. 63, no. 1-2, pp. 203–227, July 2005.

[107] Harris Interactive, "The Harris Poll® ♯17: Most People are 'Privacy Pragmatists' who, while Concerned about Privacy, will Sometimes Trade it off for Other Benefits," 2003. [Online]. Available: http://www.harrisinteractive.com/harris_poll/index.asp?PID=365

[108] A. Roth, "New algorithms for preserving differential privacy," Ph.D. dissertation, Carnegie Mellon University, July 2010.

[109] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," Communications of the ACM, vol. 13, no. 7, pp. 422–426, 1970.

[110] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of Bloom filters for distributed systems," IEEE Communications Surveys & Tutorials, vol. PP, no. 99, pp. 1–25, April 2011.

[111] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid, and Y. Tang, "On the false-positive rate of Bloom filters," Information Processing Letters, vol. 108, no. 4, pp. 210–213, 2008.

[112] A. Kirsch and M. Mitzenmacher, "Less hashing, same performance: Building a better bloom filter," in Proceedings of the 14th Annual European Symposium on Algorithms (ESA'06), ser. Lecture Notes in Computer Science, Y. Azar and T. Erlebach, Eds., vol. 4168. Zurich, Switzerland: Springer, September 2006, pp. 456–467.

[113] F. McSherry and K. Talwar, "Mechanism Design via Differential Privacy," in Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), Providence, RI, USA, October 2007, pp. 94–103.

[114] Y. D. Li, Z. Zhang, M. Winslett, and Y. Yang, "Compressive mechanism: utilizing sparse representation in differential privacy," CoRR, vol. abs/1107.3350, 2011.

[115] A. Blum, K. Ligett, and A. Roth, "A Learning Theory Approach to Non-Interactive Database Privacy," in Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC'08), C. Dwork, Ed. Victoria, British Columbia, Canada: ACM, May 2008, pp. 609–618.

[116] E.-J. Goh, "Secure indexes," Cryptology ePrint Archive 2003/216, Tech. Rep., March 2004.

[117] M. Bawa, R. J. Bayardo, R. Agrawal, and J. Vaidya, "Privacy-preserving indexing of documents on the network," The VLDB Journal, vol. 18, no. 4, pp. 837–856, August 2009.

[118] S. M. Bellovin and W. R. Cheswick, "Privacy-enhanced searches using encrypted Bloom filters," Columbia University CUCS-034-07, Tech. Rep., 2007.

[119] R. K. Pon and T. Critchlow, "Performance-Oriented Privacy-Preserving Data Integration," in Proceedings of the 2nd International Workshop on Data Integration in the Life Sciences (DILS'05), ser. Lecture Notes in Computer Science, vol. 3615. San Diego, CA, USA: Springer, July 2005, pp. 240–256.

[120] A. Shikfa, M. Önen, and R. Molva, "Broker-Based Private Matching," in Proceedings of the 11th International Symposium on Privacy Enhancing Technologies (PETS'11), ser. Lecture Notes in Computer Science, vol. 6794. Waterloo, ON, Canada: Springer, July 2011, pp. 264–284.

[121] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke, "Privacy in search logs," CoRR, vol. abs/0904.0682, 2009.

[122] F. Kerschbaum, "Public-Key Encrypted Bloom Filters with Applications to Supply Chain Integrity," in Proceedings of the 25th Annual WG 11.3 Conference on Data and Applications Security and Privacy XXV (DBSec'11), ser. Lecture Notes in Computer Science, vol. 6818. Richmond, VA, USA: Springer, July 2011, pp. 60–75.

[123] R. Nojima and Y. Kadobayashi, "Cryptographically secure Bloom-filters," Transactions on Data Privacy, vol. 2, no. 2, pp. 131–139, 2009.

[124] S. L. Warner, "Randomized response: a survey technique for eliminating evasive answer bias," Journal of the American Statistical Association, vol. 60, no. 309, pp. 63–69, March 1965.

[125] D. Dubhashi and A. Panconesi, Concentration of Measure for the Analysis of Randomized Algorithms, 1st ed. New York, NY, USA: Cambridge University Press, 2009.

[126] J. Ward, Joe H., "Hierarchical grouping to optimize an objective function," Journal of the American Statistical Association, vol. 58, no. 301, pp. 236–244, March 1963.

[127] J. A. Rice, Mathematical Statistics and Data Analysis. Cengage Learning, 2006.

[128] M. Rosenblatt, "Remarks on Some Nonparametric Estimates of a Density Function," The Annals of Mathematical Statistics, vol. 27, no. 3, pp. 832–837, 1956.

[129] E. Parzen, "On Estimation of a Probability Density Function and Mode," The Annals of Mathematical Statistics, vol. 33, no. 3, pp. 1065–1076, 1962.

[130] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum, "Differential Privacy Under Continual Observation," in Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC'10), June 2010, pp. 715–724.

[131] J. Ullman, "Answering $n^{2+o(1)}$ Counting Queries with Differential Privacy is Hard," in Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC'13), D. Boneh, T. Roughgarden, and J. Feigenbaum, Eds. Palo Alto, California, USA: ACM, June 2013, pp. 361–370.

[132] A. Boutet, D. Frey, A. Jégou, A.-M. Kermarrec, and H. B. Ribeiro, "FreeRec: An Anonymous and Distributed Personalization Architecture," in Proceedings of the 1st International Conference on Networked Systems (NETYS'13), ser. Lecture Notes in Computer Science, V. Gramoli and R. Guerraoui, Eds., vol. 7853. Marrakech, Morocco: Springer, May 2013, pp. 58–73.

[133] C. Dwork, F. D. McSherry, and K. Talwar, "The Price of Privacy and the Limits of LP Decoding," in Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC'07), D. S. Johnson and U. Feige, Eds. San Diego, California, USA: ACM, June 2007, pp. 85–94.

[134] S. Vaudenay, Ed., Proceedings of the 25th International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT'06), ser. Lecture Notes in Computer Science, vol. 4004. St. Petersburg, Russia: Springer, May 2006.

[135] T. Okamoto and X. Wang, Eds., Problem of the 10th International Conference on Practice and Theory in Public-Key Cryptography (PKC'07), ser. Lecture Notes in Computer Science, vol. 4450. Beijing, China: Springer, April 2007.

[136] J. Simon, Ed., Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC'88). Chicago, Illinois, USA: ACM, May 1988.

[137] D. S. Johnson and U. Feige, Eds., Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC'07). San Diego, California, USA: ACM, June 2007.

[138] S. Halevi and T. Rabin, Eds., Proceedings of the 3rd Theory of Cryptography Conference (TCC'06), ser. Lecture Notes in Computer Science, vol. 3876. New York, NY, USA: Springer, March 2006.

[139] J. F. E. IV, F. Fogelman-Soulié, P. A. Flach, and M. J. Zaki, Eds., Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09). Paris, France: ACM, June 2009.