BLIP: Non-interactive Differentially-Private Similarity Computation on Bloom filters

Mohammad Alaggan[1], Sébastien Gambs[1], and Anne-Marie Kermarrec[2]

[1] Université de Rennes 1 – INRIA/IRISA, Rennes, France
malaggan@irisa.fr, sebastien.gambs@irisa.fr
[2] INRIA Rennes Bretagne-Atlantique, Rennes, France
anne-marie.kermarrec@inria.fr

Abstract. In this paper, we consider the scenario in which the profile of a user is represented in a compact way, as a Bloom filter, and the main objective is to privately compute in a distributed manner the similarity between users by relying only on the Bloom filter representation. In particular, we aim at providing a high level of privacy with respect to the profile even if a potentially unbounded number of similarity computations take place, thus calling for a non-interactive mechanism. To achieve this, we propose a novel non-interactive differentially private mechanism called BLIP (for BLoom-and-flIP) for randomizing Bloom filters. This approach relies on a bit flipping mechanism and offers high privacy guarantees while maintaining a small communication cost. Another advantage of this non-interactive mechanism is that similarity computation can take place even when the user is offline, which is impossible to achieve with interactive mechanisms. Another of our contributions is the definition of a probabilistic inference attack, called the "Profile Reconstruction attack", that can be used to reconstruct the profile of an individual from his Bloom filter representation. More specifically, we provide an analysis of the protection offered by BLIP against this profile reconstruction attack by deriving an upper and lower bound for the required value of the differential privacy parameter ϵ.

1 Introduction

Consider a distributed network in which each node is an individual that has a profile representing his interests. In this context, many distributed applications, like recommender systems or private matching, require computing some kind of pairwise similarity between the profiles of different nodes[1]. Moreover, in a dynamic system, such computation takes place continuously as new nodes join the system. Some of the challenges that such systems would face include privacy and scalability issues. For instance, privacy concerns arise naturally due to the potentially sensitive nature of profiles, and some users may even refuse to participate in the similarity computation if they have no guarantees on the privacy of their profiles.

[1] For the sake of simplicity, we consider in the rest of the paper a user per machine, and we refer to it as a node.

A.W. Richa and C. Scheideler (Eds.): SSS 2012, LNCS 7596, pp. 202–216, 2012.

Dwork and Naor [12] have proved a strong impossibility result stating that for any privacy mechanism whose output has some utility (measured in terms of min-entropy[2]), there exists a piece of auxiliary information, which if available to an adversary, can cause a privacy breach whose amount of information provided on the original database (that was supposed to be protected) is at least of the same order than the utility. Therefore, Dwork and Naor [12, Section 3] recommended to depart from an *absolute* definition of privacy to a *relative* one such as *differential privacy*, which is the one that we adopt in this paper.

The main privacy guarantee provided by differential privacy is that for any privacy breach the adversary is able to cause with respect to the database, adding or removing a single row from the database will not significantly change the probability of success of causing this privacy breach for an adversary observing the output of the differentially-private mechanism. In general, in the literature the database model considered is such that each row contains the data of a particular individual and therefore the database effectively corresponds to the collection of the individuals' data. In this setting, protecting a row through differential privacy means protecting the privacy of a particular individual (which corresponds to a row). In our setting, the database is actually the profile of an individual, which is composed of the items he has liked, and therefore the guarantees provided by differential privacy protects the items contained in the profile.

One of the usual limits of differential privacy is that each time a differentially private interactive computation takes place, the user loses a little bit of privacy (as measured by the value of some privacy parameter ϵ). Therefore, if this computation takes place too many times, the user may spend all his privacy budget and remains with no privacy left (*i.e.*, the adversary will be able to reconstruct almost entirely the user's profile). However, in a dynamic system as the one we consider, there is no upper bound on the maximum number of similarity computations that can occur (as nodes continuously keep joining the system) and therefore an interactive mechanism would be of limited applicability.

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* [7] from the profile of a node, 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 node with other profile without breaching the privacy of the profiles. Moreover, this approach has exactly the same communication cost as "plain" (*i.e*, non-private) 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 [17,1]. In this paper, we address this issue by proposing an inference attack called the "Profile Reconstruction attack", that can be used to reconstruct the profile of a node from its perturbed Bloom filter representation. More specifically, we provide an analysis of the protection and the utility offered by BLIP against this

[2] We refer the interested reader to [1] for a discussion on the min-entropy of the output of a differentially-private mechanism.

attack, by deriving an upper and lower bound 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 this attack. Furthermore, we evaluate experimentally the trade-off between privacy and utility that can be reached with BLIP on a semantic clustering task in which nodes are grouped based on the similarity between their profiles.

The paper is organized as follows. First, in Section 2, we present the system model and background on Bloom filters and differential privacy necessary to understand our work. Afterwards, we propose BLIP, a non-interactive differentially private mechanism for randomizing Bloom filters in Section 3 and analyze in details the privacy guarantees it provides. In Section 4, we evaluate the impact of this mechanism on utility, as measured in terms of recall on a semantic clustering task. In Section 5, we describe a novel inference attack, called the profile reconstruction attack, that can reconstruct a profile from its Bloom filter representation and show how BLIP can be used to drastically reduce its impact. Finally, we give an overview of the related work in Section 6 before concluding in Section 7.

2 System Model and Background

2.1 System Model

We consider a distributed system of nodes, in which nodes are characterized by their profile representing the associated user's interests. For example, the profile of a node can be a vector of items that have been tagged using a collaborative platform [2] such as Delicious[3], or queries that have been performed on a search engine or ratings on movies. We denote the profile of a user as c, a set of items, which is a subset of an application domain $\mathcal{I}$. While $\mathcal{I}$ may be infinite (*e.g.*, the set of all possible URLs), a specific profile is always finite.

We assume that the computation of similarity between pairs of nodes takes place on a regular basis as nodes in the network get in contact with new nodes, and try to dynamically maintain a list of the most similar nodes in the network. For instance, in the case of a distributed semantic clustering algorithm [6], nodes need to compute regularly their similarity in order to update their semantic neighborhood. In our constructions, we also assume that a node never releases directly its profile, but rather a Bloom filter representation [7] (*cf.* Section 2.2) of it. When some node a wants to compute its similarity with another node b, it will do so by using b's public Bloom filter.

The profile is a personal and private information that should be protected, and therefore our main concern is *how to compute the similarity measure while preserving its privacy*. In this context, this means not revealing the contents of the profile and restricting the possibility for an adversary observing the Bloom filter to infer the presence or absence of a particular item in this profile. We consider a computationally-unbounded adversary that can observe the released Bloom filter, but not the internal state of a node

[3] `http://delicious.com/`

while computing the Bloom filter. In particular, the adversary does not have access to the random coins used by the node when computing a perturbed version of the Bloom filter's representation of its profile.

2.2 Bloom Filters

A Bloom filter [7] is a probabilistic data structure composed of an array of m bits along with a set $\mathcal{H}$ of k hash functions. Each hash function maps an item to a uniform random position $\{0, \ldots, m-1\}$ in the Bloom filter. Bloom filters form a compact representation of sets as they can represent any set with just m bits, for a given trade-off between the size of the structure and the false positive probability (*i.e.*, the probability of an item being considered to be in 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 are 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 [23] but we do not consider them in this paper). Both operations start by applying the k hash functions from $\mathcal{H}$ to the item in question to obtain a subset of $\{0, \ldots, m-1\}$ of positions in the Bloom filter. The add operation inserts the item by setting the bits corresponding to those positions to one in the Bloom filter, and does this independently of the previous values of these bits. The query operation checks if an item is included in the Bloom filter by verifying whether or not all those bits are set to one.

In short, the probability of false positive is a function of m, k, and n 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 [9]).

2.3 Similarity Measure

A *similarity measure* sim is a function that takes as input two sets A and B representing the profiles of two nodes and outputs a value in the range between 0 and 1 (*i.e.*, $\mathsf{sim}(A, B) \in [0, 1]$), where 0 indicates that the sets are entirely different (the profiles have no items in common) while 1 means that the sets are identical (the nodes can be considered as sharing exactly the same interests). Recall that in this paper, the profiles are represented as Bloom filters and therefore the similarity is computed directly on the Bloom filters and not on the profiles themselves. As a concrete instance, consider the *cosine similarity* that is commonly used to assess the similarity between two sets [6] and can be seen as a normalized overlap between the sets. The cosine similarity is defined as

$$\mathsf{cos\_sim}(A, B) = \frac{|A \cap B|}{\sqrt{|A| \times |B|}} \ , \tag{1}$$

where $|A|$ and $|B|$ are the sizes of the sets A and B. The *size of the set intersection* between A and B (*i.e.*, $|A \cap B|$) is equivalent to the *scalar product*[4] in the case where the sets are represented as binary vectors (*i.e.*, characteristic function).

2.4 Differential Privacy

In this paper, we are interested in a recent notion of privacy, called *differential privacy* [10]. Differential privacy aims at providing strong privacy guarantees with respect to the input of some computation by randomizing the output of this computation, and this independently of the auxiliary information that the adversary may have gathered. In our setting, the input of the computation is the raw profile of a node and the randomized output will be a perturbed version of the Bloom filter representation of this profile.

Two profiles $\mathbf{x}$ and $\mathbf{x}'$ are said to *differ in at most one element* or said to be *neighbors* if they are equal except for possibly one entry.

Definition 1 (Differential privacy [11]). *A randomized function $\mathcal{F} : \mathcal{D}^n \rightarrow \mathcal{D}^n$ is ϵ-differentially private, if for all neighboring profiles $\mathbf{x}, \mathbf{x}' \in \mathcal{D}^n$ and for all $\mathbf{t} \in \mathcal{D}^n$:*

$$\Pr[\mathcal{F}(\mathbf{x}) = \mathbf{t}] \leqslant e^{\epsilon} \cdot \Pr[\mathcal{F}(\mathbf{x}') = \mathbf{t}] \ .$$

This probability is taken over all the coin tosses of $\mathcal{F}$ and e is the base of the natural logarithm.

The parameter ϵ is public and may take different values depending on the application (for instance it could be 0.01, 0.1 or even 0.25). The smaller the value of ϵ, the higher the privacy but also, as a result, the higher the impact might be on the utility of the resulting output.

Dwork, McSherry, Nissim and Smith have designed a generic technique, called the *Laplacian mechanism* [11], that achieves ϵ-differential privacy for a function f by adding random noise to the true answer of f before releasing it. Subsequently, McSherry and Talwar have proposed another mechanism, called the *exponential mechanism* [19] that, unlike the Laplacian mechanism that works only for functions with numerical output, provides differential privacy for functions whose output is more structured (*e.g.*, graphs or trees). These mechanisms are *interactive* as they require a two-way communication protocol between the curator (the entity in charge of the database) and the client performing the query. Therefore, during this computation, the curator has to be online in order to receive the query and prepare the associate response to this query.

On the other hand, a *non-interactive* mechanism computes some function from the original database and releases it once and for all, which corresponds to a one-way communication protocol. The output released by the non-interactive mechanism can later by used by anyone to compute the answer to a particular class of queries (usually not just a single specific query), without requiring any further interactions with the curator. It is important to understand that the answer is not computed by the non-interactive mechanism, but rather that the answer can be computed from the output released by

[4] The scalar product of two vectors of length l, $a = (a_1, \cdots, a_\ell)$ and $b = (b_1, \cdots, b_\ell)$, is defined as $\sum_{i=1}^{\ell} a_i b_i$.

the non-interactive mechanism, thus after publishing this output the curator can go offline. Examples of non-interactive mechanisms for differential privacy include [4,18]. Our proposition of differentially-private Bloom filter is based on the work of Beimel, Nissim and Omri [4] in which the authors address the "binary sum" problem, which deals with the computation of the number of ones in a private binary vector. More precisely, we adapt their non-interactive mechanism to Bloom filters rather than binary vectors, with the goal of computing scalar product between two Bloom filters rather than computing the number of ones.

3 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. 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.

3.1 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 decides to flip the first two bits with some bias, thus outputting $(1, 0, 1, 0)$. BLIP can be described as a randomized function $\mathcal{F} : \{0, 1\}^n \rightarrow \{0, 1\}^n$, where each bit of the output is the opposite as the corresponding bit of the input with probability p (the flipping probability), while otherwise it remains the same. Therefore, 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 idea of randomizing each bit independently (as opposed to perturbing the final answer itself) is also known as the *randomized response technique* [24], and precedes the notion of differential privacy. The application of randomized response for differential privacy was previously studied [4] but the definition of differential privacy adopted ([4, Definition 2.4]) slightly differs from ours. Indeed, in the model adopted in this previous work [4], each individual bit belongs to and is held by a different node. Therefore, this model is very close to the setting of secure multiparty computation, while in our model all the bits of the Bloom filter belong to the same node. It is worth mentioning that the node owning this Bloom filter will publishes the perturbed version of the Bloom filter *once and for all*. This process may introduce permanent artifacts but the error they induced is bounded by $O(\sqrt{n})$ with constant probability. Due to space constraints, we leave the details of the proof of this bound to the full version of the paper.

The following is the definition for the local model of differential privacy that we also use in the setting of non-interactive differential privacy.

Definition 2 (Differential privacy (local model)). *A randomized function* $\mathcal{F}(\mathbf{x}) = (f(x_1), \ldots, f(x_n))$*, where* $f : \mathcal{D} \to \mathcal{D}$ *and* $\mathbf{x} = (x_1, \ldots, x_n)$*, is* ϵ*-differentially private, if for all values* $y, y' \in \mathcal{D}$*, and for all output* $t \in \mathcal{D}$*:*

$$\Pr[f(y) = t] \leqslant e^{\epsilon} \cdot \Pr[f(y') = t] \ ,$$

where the probability is taken over the randomness of f*.*

Definition 2 originally introduced in [4] is equivalent to Definition 1 as shown by Lemma 1. In this *local variant* of the definition, the perturbation is applied individually to each row of the database, and then the function is computed on the collection of perturbed rows. In contrast to the original (*i.e.*, global) model of differential privacy in which the function is computed first and then the answer is perturbed before releasing it. Moreover, in the local model if the dataset is split among several nodes, each node can perform the randomization locally before releasing its perturbed part of the data.

Lemma 1 (Equivalence of definitions.). *Definition 1 is equivalent to Definition 2.*

Proof. Let $(\mathbf{x}_{-i}, y)$ denotes the vector resulting from replacing the i-th coordinate the $\mathbf{x}$ with y (*e.g.*, $(x_1, \ldots, y, \ldots, x_n)$). We proceed by first proving that the proposed definition implies Definition 1 along the lines of [4].

Definition 2 $\implies$ *Definition 1.* For any two neighboring vectors $\mathbf{x}$ and $(\mathbf{x}_{-i}, y)$:

$$\frac{\Pr[\mathcal{F}(\mathbf{x}) = \mathbf{t}]}{\Pr[\mathcal{F}(\mathbf{x}_{-i}, y) = \mathbf{t}]} = \frac{\Pr[f(x_i) = t_i]}{\Pr[f(y) = t_i]} \cdot \frac{\cancel{\prod_{j \neq i} \Pr[f(x_j) = t_j]}}{\cancel{\prod_{j \neq i} \Pr[f(x_j) = t_j]}} \leqslant e^{\epsilon} \ ,$$

where $\mathbf{t} = (t_1, \ldots, t_n)$. In the above formula, the equality is obtained by independence while the inequality results from Definition 2.

Definition 1 $\implies$ *Definition 2.* For all x_i, y:

$$\frac{\Pr[f(x_i) = t_i]}{\Pr[f(y) = t_i]} = \frac{\Pr[f(x_i) = t_i]}{\Pr[f(y) = t_i]} \cdot \frac{\prod_{j \neq i} \Pr[f(x_j) = t_j]}{\prod_{j \neq i} \Pr[f(x_j) = t_j]} = \frac{\Pr[\mathcal{F}(\mathbf{x}) = \mathbf{t}]}{\Pr[\mathcal{F}(\mathbf{x}_{-i}, y) = \mathbf{t}]} \leqslant e^{\epsilon} \ ,$$

where $\mathbf{t} = (t_1, \ldots, t_n)$. In the above formula, the last equality is obtained by independence while the inequality results from Definition 1. □

Clearly, the result holds only in case $\mathcal{F}$ could be decomposed to applications of a function f to each row independently. One of the consequence of this lemma is that it is possible to analyze only the local perturbation at the level of a single bit of a Bloom filter, while still guaranteeing differential privacy for the entire Bloom filter without requiring to analyze the BLIP mechanism as a whole.

3.2 The Flipping Probability

The bit-flipping function (flip) plays the role of the local function f in Definition 2, while the BLIP mechanism corresponds to $\mathcal{F}$ in this definition. The local perturbation

function flip takes a bit x and outputs $1-x$ with probability $p < 1/2$ and x otherwise. The main challenge is to find the optimal probability p that flip should use in order to maximize utility as a function of the privacy parameter ϵ. With respect to the utility, the smaller p, the more accurate the output and therefore the best utility can be obtained by minimizing p.

It is important to realize that analyzing the flipping of a bit of the Bloom filter *in isolation* is not sufficient to guarantee the privacy of individual *items* of the profile that are encoded in the Bloom filter. Indeed, recall that an item can potentially impact 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 $\mathcal{F}$ if for each item $i \in \mathcal{I}$ and for each pair of neighboring profiles P_1, P_2 such that $i \notin P_2$ and $P_1 = P_2 \cup \{i\}$, and for all bit strings $t \in \{0,1\}^m$, the following condition holds:

$$\left| \ln \frac{\Pr(\mathcal{F}(\mathcal{B}_1) = t)}{\Pr(\mathcal{F}(\mathcal{B}_2) = t)} \right| \leqslant \epsilon \ , \tag{2}$$

where $\mathcal{B}_1$ is the Bloom filter of P_1 and $\mathcal{B}_2$ the Bloom filter of P_2.

Theorem 1 (Privacy guarantees for items). *Setting the bit-flipping probability p to $1/(1+e^{\epsilon/k})$ satisfies condition* (2) *and thus provides ϵ-differential privacy for items. In other words, 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 $T = \bigcup_{h \in \mathcal{H}} \{h(i)\}$ be the set of positions whose corresponding bits in $\mathcal{B}$ are equal to one if i is in $\mathcal{B}$, and $k = |T|$ be the number of those positions (not counting duplicates). We divide the Bloom filter $\mathcal{B}$ into two partitions: $\mathcal{B}^T$ and $\mathcal{B}^{-T}$, where the former is the restriction to the bits whose positions are in T as defined earlier, and the latter is the restriction to all other bits. A particular partition t^T (respectively t^{-T}) is defined in a similar manner. We denote by $q = 1-p$, the inverse of the flipping probability (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 properties of $\mathcal{F}$, which is itself independent for each bit by definition. The proof proceeds as follows:

$$\begin{aligned}
\left| \ln \frac{\Pr(\mathcal{F}(\mathcal{B}_1) = t)}{\Pr(\mathcal{F}(\mathcal{B}_2) = t)} \right| &= \left| \ln \frac{\Pr(\mathcal{F}(\mathcal{B}_1^T) = t^T) \Pr(\mathcal{F}(\mathcal{B}_1^{-T}) = t^{-T})}{\Pr(\mathcal{F}(\mathcal{B}_2^T) = t^T) \Pr(\mathcal{F}(\mathcal{B}_2^{-T}) = t^{-T})} \right| && \text{by independence} \\
&= \left| \ln \frac{\Pr(\mathcal{F}(\mathcal{B}_1^T) = t^T)}{\Pr(\mathcal{F}(\mathcal{B}_2^T) = t^T)} \right| && \text{because } \mathcal{B}_1^{-T} = \mathcal{B}_2^{-T} \\
&= \left| \ln \frac{\Pr(\mathcal{F}(\overbrace{1 \ldots 1}^{k}) = t^T)}{\Pr(\mathcal{F}(\mathcal{B}_2^T) = t^T)} \right| && \text{as } i \text{ is in } \mathcal{B}_1 \\
&= \left| \ln \frac{p^z q^{k-z}}{\delta} \right| = \left| \ln \left[\frac{q^k}{\delta} \left(\frac{p}{q} \right)^z \right] \right| \ ,
\end{aligned}$$

where $p^k \leqslant \delta \leqslant q^k$ (as $\mathcal{B}_2^T$ is just a fixed bit string) and $0 \leqslant z \leqslant k$ is the number of zero bits in t^T. If $p = 1/(1+e^{\epsilon/k})$ then $q^x = (1-p)^x = \exp(\epsilon x/k) p^x$ and

$(p/q)^x = \exp(-\epsilon x/k)$. Therefore $p^k \leqslant \delta \leqslant \exp(\epsilon)p^k$ or $1 \leqslant \delta/p^k \leqslant \exp(\epsilon)$ implying that $1 \geqslant p^k/\delta \geqslant \exp(-\epsilon)$, and hence $0 \geqslant \ln(p^k/\delta) \geqslant -\epsilon$.

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

We have for the second term $0 \leqslant \epsilon(k-z)/k \leqslant \epsilon$, from which it follows immediately that the maximum value of $\left|\ln(p^k/\delta) + \epsilon(k-z)/k\right|$ is at most ϵ, thus proving the theorem. Moreover, this maximum is tight showing that this choice of p is tight as well. □

Remark 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 Definition 2), it is easy to verify that $p = \frac{1}{1+e^\epsilon}$ is the minimum value for which

$$\left|\ln\frac{\Pr(\mathsf{flip}(y) = b)}{\Pr(\mathsf{flip}(y') = b)}\right| \leqslant \epsilon \quad ,$$

holds for all b, y, y' in $\{0, 1\}$. This value coincides with the value obtained by substituting 1 for k in $p = \frac{1}{1+e^{\epsilon/k}}$, thus showing its optimality.

We show how it is possible to address and correct the approximation error resulting from the bit flipping mechanism in the full version of this paper due to space constraints.

4 Utility Evaluation

In this section, we evaluate experimentally how the utility is impacted by the BLIP mechanism by applying it for the computation of a semantic clustering algorithm. More precisely, we consider a semantic clustering protocol whose objective is to provide each node of the system with its c closest nodes, as measured by the similarity based on the scalar product on their Bloom filters. In this setting, we measure the utility in terms of *recall*, which we formally define later [6]. We have used the three following datasets for our experiments from the Delicious and Digg collaborative platforms traces and a news survey conducted in our lab. The survey has been conducted on 120 users who gave their (binary) opinion on 100 news items. The profile of a user in the Digg trace are the items forwarded; in the Delicious trace, the items tagged; and in the survey trace, the news items liked.

	# of users	# of items	average profile size	Bloom filter size (m)	# of hash func. (k)
Delicious	500	51453	135	5000	18
Digg	500	1237	317	5000	18
Survey	120	196	68	5000	18

The experiment is conducted by simulating a clustering protocol, similarly to [6]. In this experiment, each node corresponds to one node from one of the three datasets described previously. In a random manner, each node i divides 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 $|s_i|/|t_i| \approx 0.9$ (the value 0.9 has been chosen to match the setting of

[6]). The experiment is split into two steps: the clustering (*i.e.*, training) phase and the search phase.

The clustering phase is divided into 20 rounds. At each round, every node i exchanges its Bloom filter with its current neighbors and some other nodes picked at random [15]. Based on the information acquired, the node keeps as new neighbors, the nodes displaying a high similarity (if any). At the end of the clustering phase, each node will have chosen the c most similar nodes met so far as its *neighbors*. Usually after a sufficient number of rounds, the neighbors of a node should converge towards the optimal ones in terms of the recall (as defined in the next paragraph) [6].

Afterwards, the utility is evaluated by checking if each node can find the items contained in its search set s_i in the profiles of its neighbors. The number of the items found is divided by $|s_i|$ (for normalization) and forms the *recall* metric r (for $0 \leqslant r_i \leqslant 1$). Considering a particular clustering, the higher r, the more useful the clustering is. More formally, during the search phase, each node i computes its recall value r_i defined as:

$$r_i = \frac{|s_i \cap (\cup_{j \in n_i} t_j)|}{|s_i|} \quad ,$$

where n_i is the set of neighbors of the node i once the clustering phase is terminated. The overall recall $r = \mathsf{mean}(r_1, \ldots, r_n)$ simply corresponds to the average of the recalls of all the $n = 500$ (or $n = 120$ for the survey dataset) nodes (*cf.* Figure 1).

We provide in Figure 1 the results of the experiments for the recall versus privacy parameter ϵ (average over 100 runs). On this plot, we use $p = 1/(1 + \exp(\epsilon/k))$ for the flipping probability ($k = 18$). The main plot displays the recall obtained when using the cosine similarity based on the bias-corrected scalar product (which is developed in the full version of the paper). 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

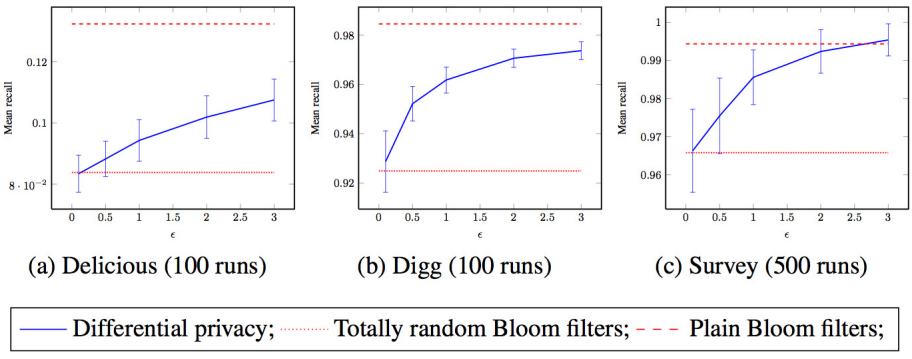

Fig. 1. Recall obtained with BLIP. The bars correspond to the standard deviation.

baseline (1) and baseline (2) are directly dependent of the characteristics of the dataset considered.

From these plots, we can observe that the utility remains non-trivial even for values of ϵ that are small, in comparison to the utility obtained with totally randomized Bloom filters (which directly lead to a random neighborhood for each node). In general the utility obtained is far from the non-private solution (for Delicious the value is about 0.261) in which nodes exchange directly their profiles but 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 between privacy (*i.e.*, resilience against the profile reconstruction attack) and utility (*i.e.*, recall in the case of the semantic clustering task).

5 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. In a setting in which each node has its profile represented in the form of a Bloom filter, the main objective of the adversary is to infer the description of the profile of a node from its Bloom filter representation. We assume that in the same manner as other nodes in the network, the adversary (which in fact could simply be one of these nodes) can have easily 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.

5.1 Description of the Attack

Remember that we consider a computationally-unbounded adversary. In the *profile reconstruction 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 to ensure ϵ-differential privacy (we assume the value of p to be a public parameter and to be known by the adversary to avoid some kind of security by obscurity). 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, hence the name "profile reconstruction attack". We measure the success of this attack by measuring how close the reconstructed profile is to the original one in terms of the (squared) cosine similarity. Note that due to the probability of false positives inherent to Bloom filters (and this even in the non-perturbed case), the exact reconstruction of the original profile with 100% confidence may be impossible.

In a nutshell, the implementation of the profile reconstruction attack that we propose works as follows. For each item in the application domain, the adversary checks its corresponding bits in the Bloom filter it observed and sets k_0 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 $p^{k_1}(1-p)^{k_0}\binom{k_1+k_0}{k_0} > c$, 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 exactly the probability that k_0 bits were flipped while k_1 bits were not flipped. Note that k_0 (respectively k_1) is the number of bits in the subset of the Bloom filter corresponding to the item currently considered that were set to 0 (respectively 1). We also tried other possible predicates such as the predicate $k_1 > c' \cdot k_0$ for some other appropriate constant c', but the other predicates were always less efficient than the one considered, therefore we disregarded them.

5.2 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 the three datasets introduced earlier. The objective of this experiment is to derive empirically 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 squared cosine similarity is computed between the reconstructed profile outputted by the adversary 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 mean similarity value over all values of c, where the mean (and standard deviation) is taken over the users.

Figure 2 shows significant diversity between the three considered datasets in the least possible adversarial success. For instance, the smallest squared cosine similarity attained in the Survey dataset is about 0.3 whereas it is 0.002 in the Delicious dataset. This may possibly stem from the difference in the size of item domain among the datasets. Nonetheless, we can conclude that in the worst case BLIP successfully prevents the adversary from producing a reconstructed profile having a cosine similarity with the original one much higher than the baseline (which is different for each dataset) when ϵ is less than 10, for all the considered datasets. As a consequence, we can derived an empirical upper bound on ϵ.

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 taking also into account the error bounds discussed in the full version of the paper due to space constraints.

6 Related Work

Non-interactive differential privacy. Most of the previous works studying non-interactive mechanisms in the context of differential privacy [8,18] have considered mechanisms that release a synthetic database that can be used to perform queries instead of the original one. The first work [8] is very inefficient due to its high computational complexity while [18] considers only input databases that have a sparse representation.

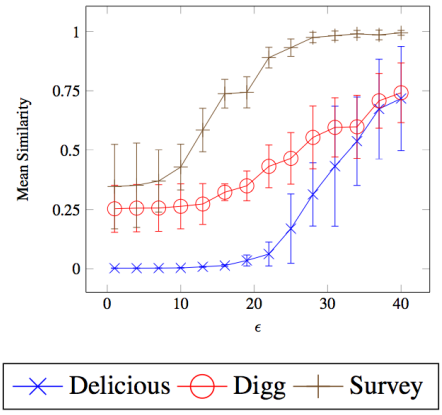

Fig. 2. Profile reconstruction attack. The vertical bars represent the standard deviation. The values on the y-axis are squared cosine similarity between the original profile and the reconstructed profile.

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 on 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 [13,3,5], private information retrieval [21], private matching [22], private publication of search logs [14] and anti-counterfeiting in supply chains [16].

Due to lack of space we do not detailed these methods but for the closest to our work is [14], 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 to apply it on normal sets such as user profiles.

7 Conclusion

In this paper, we have 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 scalar 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 an generic inference attack against the flipped Bloom filter, called the profile reconstruction attack, which enables reconstruction of the full original profile (almost) if applied on a plain (*i.e.*, non-randomized) Bloom filter but does not work anymore on the perturbed Bloom filter if the value of the parameter ϵ is chosen wisely (we provide a technique for choosing an upper bound to this parameter).

In the future, we plan to broaden the scope of inference attacks considered in order to evaluate their rate of success on a Bloom filter released by the BLIP mechanism. In particular, we want to design more sophisticated inference attacks in which the adversary has some partial knowledge about the profile of a user or that exploit the correlations between items and assess empirically if the ϵ-differential privacy ensures an efficient protection against these attacks. We also plan to study how the relaxation of the notion of full differential privacy to computational differential privacy [20] and the combination with the compressive mechanism [18] can be used to provide a lower error bound for the same privacy guarantees. Another avenue of research that we will explore is how the difference of expectations in terms of privacy among different users (which could lead to different values of ϵ) affect the privacy guarantees offered globally for different types of users and items.

References

1. Alvim, M.S., Andrés, M.E., Chatzikokolakis, K., Palamidessi, C.: On the Relation between Differential Privacy and Quantitative Information Flow. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part II. LNCS, vol. 6756, pp. 60–76. Springer, Heidelberg (2011)
2. Amer-Yahia, S., Benedikt, M., Lakshmanan, L.V.S., Stoyanovich, J.: Efficient network aware search in collaborative tagging sites. PVLDB 2008, 1(1) (August 2008)
3. Bawa, M., Bayardo, R.J., Agrawal, R., Vaidya, J.: Privacy-preserving indexing of documents on the network. The VLDB Journal 18(4), 837–856 (2009)
4. Beimel, A., Nissim, K., Omri, E.: Distributed Private Data Analysis: Simultaneously Solving How and What. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 451–468. Springer, Heidelberg (2008)
5. Bellovin, S.M., Cheswick, W.R.: Privacy-enhanced searches using encrypted Bloom filters. Tech. rep., Columbia University CUCS-034-07 (2007)
6. Bertier, M., Frey, D., Guerraoui, R., Kermarrec, A.M., Leroy, V.: The Gossple anonymous social network. In: Proceedings of the 11th International Middleware Conference (Middleware 2010), ACM/IFIP/USENIX, Bangalore, India, November 29 - December 3, pp. 191–211 (2010)

7. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13(7), 422–426 (1970)
8. Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: Dwork, C. (ed.) Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC 2008), pp. 609–618. ACM, Victoria (2008)
9. Bose, P., Guo, H., Kranakis, E., Maheshwari, A., Morin, P., Morrison, J., Smid, M., Tang, Y.: On the false-positive rate of Bloom filters. Information Processing Letters 108(4), 210–213 (2008)
10. Dwork, C.: Differential Privacy: A Survey of Results. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)
11. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
12. Dwork, C., Naor, M.: On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. Journal of Privacy and Confidentiality 2(1), 93–107 (2010)
13. Goh, E.J.: Secure indexes. Tech. rep., Cryptology ePrint Archive 2003/216 (March 16, 2004)
14. Götz, M., Machanavajjhala, A., Wang, G., Xiao, X., Gehrke, J.: Privacy in search logs. CoRR abs/0904.0682 (2009)
15. Jelasity, M., Guerraoui, R., Kermarrec, A.-M., van Steen, M.: The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations. In: Jacobsen, H.-A. (ed.) Middleware 2004. LNCS, vol. 3231, pp. 79–98. Springer, Heidelberg (2004)
16. Kerschbaum, F.: Public-Key Encrypted Bloom Filters with Applications to Supply Chain Integrity. In: Li, Y. (ed.) DBSec. LNCS, vol. 6818, pp. 60–75. Springer, Heidelberg (2011)
17. Lee, J., Clifton, C.: How Much Is Enough? Choosing ϵ for Differential Privacy. In: Lai, X., Zhou, J., Li, H. (eds.) ISC 2011. LNCS, vol. 7001, pp. 325–340. Springer, Heidelberg (2011)
18. Li, Y.D., Zhang, Z., Winslett, M., Yang, Y.: Compressive mechanism: utilizing sparse representation in differential privacy. CoRR abs/1107.3350 (2011)
19. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), Providence, RI, USA, October 20-23, pp. 94–103 (2007)
20. Mironov, I., Pandey, O., Reingold, O., Vadhan, S.P.: Computational Differential Privacy. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 126–142. Springer, Heidelberg (2009)
21. Pon, R.K., Critchlow, T.: Performance-Oriented Privacy-Preserving Data Integration. In: Ludäscher, B., Raschid, L. (eds.) DILS 2005. LNCS (LNBI), vol. 3615, pp. 240–256. Springer, Heidelberg (2005)
22. Shikfa, A., Önen, M., Molva, R.: Broker-Based Private Matching. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 264–284. Springer, Heidelberg (2011)
23. Tarkoma, S., Rothenberg, C.E., Lagerspetz, E.: Theory and practice of Bloom filters for distributed systems. IEEE Communications Surveys & Tutorials (99), 1–25 (2011)
24. Warner, S.L.: Randomized response: a survey technique for eliminating evasive answer bias. Journal of the American Statistical Association 60(309), 63–69 (1965)