Challenging differential privacy: the case of non-interactive mechanisms

Raghavendran Balu, Teddy Furon, Sébastien Gambs

Challenging differential privacy: the case of non-interactive mechanisms

Raghavendran Balu[1], Teddy Furon[1], and Sébastien Gambs[1,2]

[1] Inria Rennes Bretagne-Atlantique, France
[2] Université de Rennes 1 / IRISA, France
rbalu@inria.fr, teddy.furon@inria.fr, sgambs@irisa.fr

Abstract. In this paper, we consider personalized recommendation systems in which before publication, the profile of a user is sanitized by a non-interactive mechanism compliant with the concept of differential privacy. We consider two existing schemes offering a differentially private representation of profiles: BLIP (BLoom-and-flIP) and JLT (Johnson-Lindenstrauss Transform). For assessing their security levels, we play the role of an adversary aiming at reconstructing a user profile. We compare two inference attacks, namely *single* and *joint* decoding. The first one decides of the presence of a single item in the profile, and sequentially explores all the item set. The latter strategy decides whether a subset of items is likely to be the user profile, and considers all the possible subsets. Our contributions are a theoretical analysis as well as a practical implementation of both attacks, which were evaluated on datasets of real user profiles. The results obtained clearly demonstrates that joint decoding is the most powerful attack, while also giving useful insights on how to set the differential privacy parameter ϵ.

Keywords: Differential privacy, Joint decoding.

1 Introduction

Most of the social applications, like recommender systems or private matching, require computing some kind of pairwise similarity between the profiles of different users. Some of the challenges that such systems 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 if they have no guarantees on the privacy of their profiles.

To address these concerns, the concept of differential privacy [3] has been introduced by Dwork in the context of private analysis on statistical databases and has known a widespread adoption in the privacy community. In a nutshell, the main privacy guarantee provided by differential privacy is that for any computation that will be performed on the database, adding or removing a single row from the database will not significantly change the probability of a particular output. Usually, the database is composed of the collection of the individuals' data, and differential privacy protects the privacy of a particular individual,

which corresponds to a row of the database. In contrast in our context, the "database" is actually the profile of an individual (*e.g.*, composed of the items he has liked) and therefore the guarantees provided by differential privacy applies to the protection of the items contained in the profile.

One of the usual limits of differential privacy is that each time a differentially private computation takes place, the user loses a little bit of privacy (as measured by the value of the 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. The adversary is then able to reconstruct almost entirely the user's profile. One possible approach to solve this problem is to sanitize the profile of a user with a non-interactive mechanism compliant with the concept of differential privacy before his publication. In particular, this paper investigates the privacy guarantees offered by two non-interactive mechanisms offering a differentially private representation of profiles: BLIP (BLoom-and-flIP) [1] and JLT (Johnson-Lindenstrauss Transform) [2].

In this paper, we propose two inference attacks that help to assess the privacy guarantee provided by the BLIP and JLT mechanisms. We provide an analysis of the utility and the protection offered by BLIP and JLT against these attacks, by deriving theoretical bounds on the resulting approximation error generated by a specific value of the privacy parameter. Furthermore, we evaluate experimentally the trade-off between privacy and utility achieved by these mechanisms. These attacks helps to better understand the privacy guarantees offered by a differentially-private mechanism, while also enabling the privacy practitioner to tune ϵ experimentally.

A detailed survey on inference attacks on sanitized data can be found in [4] and [5]. Common attacks include eigen-analysis [6, 7], MAP estimation [7], Independent Component Analysis (ICA) [8] and distribution analysis [9]. MAP estimation and ICA make direct assumptions on the distribution of the original data, whereas distribution analysis and our approach estimate it from publicly available information. In addition, eigen-analysis makes even stronger assumptions on the representation of data and thus is not generic enough to apply to representations studied in this paper. Furthermore, the possibility of using probabilistic inference techniques to attack sanitized histogram data has been illustrated in [10] and [11]. In these works, bounds of records count are estimated from histogram of attributes coming from a Markov Chain Monte Carlo (MCMC) simulation. This line of work is different from our approach aiming at reconstructing a user profile from perturbed data. Application of probabilistic inference techniques for parameter estimation on differentially private data is illustrated in [12]. In this work, the authors have also experimentally validated their approach using MCMC on parameter estimation of logistic regression and probabilistic inference of principal components. Although their objective was not directly the reconstruction of data, their approach demonstrates that probabilistic inference is possible on differentially private data.

The outline of the paper is the following. First in Section 2, we give an overview of the concept of differential privacy. Then, we describe two non-

interactive differentially private mechanisms in Sections 3 and 4 that have been recently proposed: BLIP (*BLoom-and-flIP*) [1] and one based on the *Johnson-Lindenstrauss Transform* (JLT in short) [2]. These mechanisms transform the profile of a user into a compact representation that estimate the similarity between profiles while hiding the presence or absence of a particular item in the profile (in the sense of differential privacy). Afterwards in Section 5, we provide a theoretical analysis showing that the joint decoding strategy is more powerful than the single decoding strategy in reconstructing the profile of a user. Finally in Section 6, we propose a tractable implementation of this strategy based on the MCMC algorithm before reporting in Section 7 the results on two real datasets.

2 Differential privacy

In this paper, we are interested in a strong privacy notion called *differential privacy* [3]. 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 might have gathered. In our setting, the input of the computation is the profile of a user and the randomized output will be a perturbed version of a compact representation of this profile (*e.g.*, a Bloom filter or a random projection).

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 [13]). *A randomized function $\mathcal{F} : \mathcal{D}^n \to \mathcal{D}^n$ is ϵ-differentially private, if for all neighboring profiles $\mathbf{x}, \mathbf{x}' \in \mathcal{D}^n$ and for all $\mathbf{t} \in \mathcal{D}^n$:*

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

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 0.1, 0.25, 1.5 or even 10). The smaller the value of ϵ, the higher the privacy but also as a consequence the higher the impact might be on the utility of the resulting output. A relaxed notion differential privacy called (ϵ,δ)-differential privacy [14], can be seen as a probabilistic variant in which the guarantees of differential privacy hold with probability of $1 - \delta$.

Originally, differential privacy was developed within the context of private data analysis and the main guarantee is that if a differentially private mechanism is applied on a dataset composed of the personal data of individuals, no output would become significantly more (or less) probable whether or not a *single* participant contributes to the dataset. This means that observing the output of the mechanism only gains negligible information about the presence (or absence) of a particular individual in the database. This statement is a statistical property about the behavior of the mechanism (*i.e.*, function) and holds independently of the auxiliary knowledge that the adversary might have gathered. More specifically, even if the adversary knows the whole database but one individual row, a

mechanism satisfying differential privacy still protects the privacy of this individual row. In our setting, the database that we want to protect is the profile of a user and the objective of a differentially private mechanism is to hide the presence or absence of a particular item in the profile.

Dwork, McSherry, Nissim and Smith have designed a generic technique, called the *Laplacian mechanism* [13], 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 the *exponential mechanism* [15] which 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). Both previous mechanisms (*i.e.*, Laplacian and Exponential 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, 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 be 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 computed from the output released by the non-interactive mechanism, thus after publishing this output the curator can go offline. One particular type of non-interactive mechanism is the *generation of a synthetic dataset* that allows the answer to certain class of queries (but not necessarily all) to be approximated. Examples of non-interactive mechanisms for differential privacy include [16, 17].

In the next sections, we describe two non-interactive mechanisms that have recently been proposed. The first mechanism is based on randomizing a Bloom filter representation of the profile [1] while the second relies on the application of the Johnson-Lindenstrauss transform and the addition of noise [2]. Both mechanisms preserve some global properties such as the ability to compute a distance between two profiles while hiding the details of the profiles themselves.

3 BLIP

The main objective of BLIP [1] is to prevent the adversary from learning the presence (or absence) of an item in the profile of a user by observing the Bloom filter representation of this profile. Our theoretical analysis provided in Section 5 is based on the model of profiles and the BLIP sanitization described thereafter.

3.1 Setup of BLIP

The setup that we consider for the theoretical analysis is the following. We assume that a profile P is a list of c items randomly picked from a set of $N \in \mathbb{N}^\star$ possible items: $P = \{j_1, \ldots, j_c\}$. We denote the set of items by $[N]$, with

$[N] \triangleq \{1, \ldots, N\}$ and the set of all possible profiles by $\mathcal{P}$. This set is a subset of the power set of $[N]$ and we have $|\mathcal{P}| = \binom{N}{c}$. For the moment, we make the assumption that c is publicly known, but this hypothesis will be lifted later by inferring this value directly from the Bloom filter.

The profile is first encoded in the form of a Bloom filter, which is a binary string of L bits. Each item $j \in P$ is hashed through K different hash functions $(h_1, \ldots, h_K)$. Each hash function yields a position $h_k(j)$ in the Bloom filter, pseudo-randomly selected based on the identifier of the item j. One simple technique to implement this is to rely on K cryptographic hash functions modulo L. We call the codeword $\mathbf{X}_j$ associated to item j the following string of L bits:

$$X_j(\ell) = \begin{cases} 1 \text{ if } \exists k \in [K] \text{ such that } h_k(j) = \ell, \\ 0 \text{ otherwise.} \end{cases} \tag{1}$$

The Bloom filter associated to the profile $P = \{j_1, \ldots, j_c\}$ is denoted by $\mathbf{B}_P$ and computed as the aggregation of the codewords:

$$\mathbf{B}_P = \mathbf{X}_{j_1} \vee \ldots \vee \mathbf{X}_{j_c}, \tag{2}$$

in which $\vee$ denotes the bit-wise (inclusive) OR operator. Our presentation of Bloom filters is different than usual to stress the link with our general problem.

The BLIP mechanism adds noise to the Bloom filter representation of a profile before publishing it. We denote the output of BLIP by $\tilde{\mathbf{B}}_P$:

$$\tilde{\mathbf{B}}_P = \mathbf{B}_P \oplus \mathbf{N}, \tag{3}$$

in which $\oplus$ corresponds to the bit-wise logical (exclusive) XOR operator and $\mathbf{N} \in \{0,1\}^L$ is a random binary string of size L, whose symbols are *i.i.d.* (independent and identically distributed) as a Bernoulli distribution $\mathcal{B}(p_\epsilon)$ (*i.e.* , $N(\ell) \in \{0,1\}$ and $\mathbb{P}[N(\ell) = 1] = p_\epsilon$, $\forall \ell \in [L]$). Alaggan, Gambs and Kermarrec [1] proved that the BLIP mechanism ensures ϵ-differential privacy for the items of the profile if

$$p_\epsilon = 1/(1 + e^{\epsilon/K}). \tag{4}$$

3.2 The simple model

We assume that the hash functions produce independently random outputs, which means that the probability that $h_k(j)$ "points" to a given index is $1/L$. This assumption implies that the bits of the codewords can be modeled as independent Bernoulli random variables: $X_j(\ell) \sim \mathcal{B}(p)$, $\forall (j, \ell) \in [N] \times [L]$ with

$$p \triangleq \mathbb{P}[X_j(\ell) = 1] = 1 - \left(1 - \frac{1}{L}\right)^K. \tag{5}$$

For a random P composed of c items, we have $B_P(\ell) \sim \mathcal{B}(\pi_c)$, $\forall \ell \in [L]$, with

$$\pi_c \triangleq \mathbb{P}[B_P(\ell) = 1] = 1 - (1-p)^c = 1 - \left(1 - \frac{1}{L}\right)^{cK}. \tag{6}$$

As for the BLIP, $\tilde{\mathbf{B}}_P$ contains i.i.d. random symbols $\tilde{B}_P(\ell) \sim \mathcal{B}(\tilde{\pi}_c)$ with

$$\tilde{\pi}_c \triangleq \mathbb{P}[\tilde{B}_P(\ell) = 1] = (1 - p_\epsilon)\pi_c + p_\epsilon(1 - \pi_c). \tag{7}$$

3.3 More complex models

This subsection presents two possible extensions of the simple model, in which we no longer assume that c is fixed in advance and publicly known.

To account for this, we introduce the probability $\mathbb{P}[|P| = c]$, in which $|P|$ denotes the number of items in P. Then, we have to replace π_c by:

$$\pi_c \to \pi = \sum_{c>0} \pi_c \mathbb{P}[|P| = c]. \tag{8}$$

This new expression leads to $\tilde{\pi} = (1 - p_\epsilon)\pi + p_\epsilon(1 - \pi)$. Not knowing c may not be a big challenge for the adversary because he can easily infer the number of items in a profile. The quantity $\omega(\tilde{\mathbf{B}}_P)/L$, in which $\omega(.)$ is the Hamming weight of a binary string (the number of bits set to one), is an unbiased estimator of $\tilde{\pi}_c$. Inverting (7) is possible when $p_\epsilon \neq 1/2$ (*i.e.* , $\epsilon > 0$) since p_ϵ is public:

$$\hat{\pi}_c = \frac{\omega(\tilde{\mathbf{B}}_P)/L - p_\epsilon}{1 - 2p_\epsilon}, \tag{9}$$

which in turn gives an estimator $\hat{c}$ by inverting (6). In the same way, a confidence interval for $\tilde{\pi}_c$ based on $\omega(\tilde{\mathbf{B}}_P)/L$ yields a confidence interval $[c_{\min}, c_{\max}]$ on c.

An even more refined model consists in taking into account the popularity of the items. Indeed, popular items impact the Bloom filter by ensuring that some of its bits are more likely to be set to one. To tackle this issue, we still pretend that the bits are independent but distributed according their own Bernoulli law: $B_P(\ell) \sim \mathcal{B}(\pi(\ell))$, $\forall \ell \in [L]$. The same model holds for the BLIP symbols: $\tilde{B}_P(\ell) \sim \mathcal{B}(\tilde{\pi}(\ell))$, with $\tilde{\pi}(\ell) = (1 - p_\epsilon)\pi(\ell) + p_\epsilon(1 - \pi(\ell))$.

4 JLT

Kenthapadi and co-authors [2] proposed another mechanism to prevent the adversary from learning the presence (or absence) of an item in the profile, although their scheme tackles a different data type (*i.e.*, real vector). In the sequel, we denote this proposal by JLT because it is based on the Johnson-Lindenstrauss Transform.

4.1 Description

The profile is encoded in the form of a real vector of length L as follows. A codeword $\mathbf{X}_j$ associated to item j is a real vector. Its L components have been independently and identically drawn such that $X_j(i) \overset{i.i.d.}{\sim} \mathcal{N}(0, 1/L)$, $\forall (i, j) \in [L] \times N$. The codebook $(\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_N)$ is generated once for all and is public. Profile P is encoded into vector $\mathbf{Y}_P = \sum_{j \in P} \mathbf{X}_j$, then the user adds a noise $\mathbf{N}$ (private data) before publishing $\tilde{\mathbf{Y}}_P = \mathbf{Y}_P + \mathbf{N}$. The authors of [2] recommend a white Gaussian noise: $N(i) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$. According to [2, Lemma 2], if

$$L \geq 2(\log(N) + \log(2/\delta)), \quad \sigma \geq \frac{4}{\epsilon}\sqrt{\log(1/\delta)} \text{ and } \epsilon < \log(1/\delta) \tag{10}$$

then this mechanism complies with (ϵ, δ)-differential privacy (for $0 < \delta < 1$).

4.2 A simple probabilistic model

The adversary does not know the profile P and therefore he models the observation $\tilde{\mathbf{Y}}_P$ as a white Gaussian noise since $\tilde{\mathbf{Y}}_P$ is the sum of $c+1$ white Gaussian noises. As these patterns are statistically independent, their powers sum up so that $\tilde{Y}_P(i) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2 + c/L)$. We assume now that σ^2 is a recommended noise power, and thus that it is a public parameter. This allows the adversary to estimate the number of items in profile P in the following manner:

$$\hat{c} = \frac{L}{L-1} \sum_{i=1}^{L} \tilde{Y}_P(i)^2 - L\sigma^2. \tag{11}$$

Consider now the case in which the adversary knows that the item j is in the profile. This knowledge stems into a refined statistical model of the observation: $\tilde{Y}_P(i) \overset{i.i.d.}{\sim} \mathcal{N}(X_j(i), \sigma^2 + (c-1)/L)$. In the same way, knowing the profile P ends up with $\tilde{Y}_P(i) \overset{i.i.d.}{\sim} \mathcal{N}\left(\sum_{j \in P} X_j(i), \sigma^2\right)$.

5 Theoretical analysis

In this section, we propose two decoders that can be used by an adversary to reconstruct the profile of a given user out of his public representation. This analysis is detailed for the BLIP mechanism, but similar concepts hold for the JLT scheme. The expressions of the information theoretical quantities are given in Appendix 9 for BLIP and Appendix 10 for JLT.

5.1 Single decoder

From the observation of one BLIPed representation $\tilde{\mathbf{b}}$, the adversary would like to infer which item belongs to the original profile. The adversary can conduct this inference by analyzing the L symbols of $\tilde{\mathbf{b}}$ and making an hypothesis test about the presence of item j in the underlying profile.

- $\mathcal{H}_0$: Item j is not in the profile, which means that the observed BLIP symbols are statistically independent from the symbols of codeword $\mathbf{X}_j$: $\mathbb{P}[\tilde{B}_P(\ell), X_j(\ell)] = \mathbb{P}[\tilde{B}_P(\ell)]\mathbb{P}[X_j(\ell)]$, $\forall \ell \in [L]$.
- $\mathcal{H}_1$: Item j belongs to P, and thus there is a slight dependency between the symbols of the observed BLIP and that of codeword $\mathbf{X}_j$: $\mathbb{P}[\tilde{B}_P(\ell), X_i(\ell)] = \mathbb{P}[\tilde{B}_P(\ell)|X_i(\ell)]\mathbb{P}[X_i(\ell)]$, $\forall \ell \in [L]$.

For a given item, this test may make two types of error: 1) False positive rate α_1: The probability of detecting the presence of an item that does not belong to the profile; 2) False negative rate α_2: The probability of missing the presence of an item that belongs to the profile. Information theory gives an upper bound on the performance of the test thanks to the Stein's lemma. More precisely, for a given α_2, the probability of false positive cannot be lower than

$$\alpha_1 \geq e^{-(I(\tilde{\mathbf{B}}_P; \mathbf{X}) + 1)/(1-\alpha_2)}, \tag{12}$$

in which $I(\tilde{\mathbf{B}}_P; \mathbf{X})$ is the mutual information between a BLIPed filter and the codeword of an item of the profile.

This test concerns a particular item, but an adversary that wants to reconstruct the whole profile needs to repeat it for the whole ensemble of size N. This repetition increases the global probability of false positive η_1:

$$\eta_1 = 1 - (1 - \alpha_1)^{N-c} \lesssim N\alpha_1, \tag{13}$$

in which we assume that $N\alpha_1 \ll 1$ and $c \ll N$. η_1 is the probability that at least one item not in the profile is detected as belonging to the profile. At the end, for targeted error probabilities (α_2, η_1), inequality (12) constraints the size of the item ensemble the adversary can deal with:

$$\log(N) \leq \frac{I(\tilde{\mathbf{B}}_P; \mathbf{X})}{1 - \alpha_2} + \log \eta_1. \tag{14}$$

The last inequality stresses the important role of $I(\tilde{\mathbf{B}}_P; \mathbf{X})$. Appendices 9 and 10 provide expressions of this quantity for the BLIP and JLT mechanisms.

5.2 Joint decoder

Let us consider another strategy. From the observation $\tilde{\mathbf{b}}$, the adversary would like to test whether P was the original profile that gave birth to this BLIPed representation. The difference with the previous approach is that the presence of items are not tested independently but jointly, hence the name "joint decoder".

Basically, the analysis is the same as previously except that the information theoretic quantity is now $I(\tilde{\mathbf{B}}_P; P) = I(\tilde{\mathbf{B}}_P; (\mathbf{X}_{j_1}, \ldots, \mathbf{X}_{j_c}))$ and that the ensemble of profiles is much bigger. Roughly, $\log(|\mathcal{P}|) \approx c \log N$, thus we have:

$$\log(N) \leq \frac{I(\tilde{\mathbf{B}}_P; P)}{c(1 - \alpha_2)} + \log \eta_1. \tag{15}$$

Stated differently, the performance of this approach is driven by the quantity $I(\tilde{\mathbf{B}}_P; P)/c$. Theorem [18, Eq. (3.4)] states that $I(\tilde{\mathbf{B}}_P; (\mathbf{X}_{j_1}, \ldots, \mathbf{X}_{j_c}))/c \geq I(\tilde{\mathbf{B}}_P; \mathbf{X}_j)$, which means that considering the items jointly yields better performances. Appendices 9 and 10 provide expressions of this quantity for respectively the BLIP and JLT mechanisms. For this first scheme, subsection 9.2 shows that the difference $I(\tilde{\mathbf{B}}_P; (\mathbf{X}_{j_1}, \ldots, \mathbf{X}_{j_c}))/c - I(\tilde{\mathbf{B}}_P; \mathbf{X}_j)$ can be be substantial for practical setups. We also provide upper bounds simply depending on ϵ.

6 Practical decoders

The previous section can be summarized as follows: joint decoding is theoretically more powerful than single decoding. However, no complexity argument has been so far taken into account. This section deals with this issue by proposing practical implementations of a single and a joint decoder. Again, we take the example of BLIP but our approach is more generic as it works also with JLT.

6.1 Single decoders

In practice, a single decoder computes from the observed BLIPed profile a score s_j for any item $j \in [N]$, which reflects the likelihood of belonging to the profile (*i.e.*, the most likely item has the highest score). The score is compared to a threshold to decide whether or not the item should be included in the reconstructed profile. The complexity of this single decoder is $O(N)$ since it is exhaustive and goes through all the possible items.

As a practical implementation, we propose the Maximum Likelihood decoder in which the score $s_j = \log \frac{\mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}} | j \in P]}{\mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}}]}$ equals, by independence of the symbols:

$$s_j = n_{11} \log \frac{1 - p_\epsilon}{\tilde{\pi}} + n_{01} \log \frac{p_\epsilon}{1 - \tilde{\pi}}, \quad \text{with:} \tag{16}$$

$$n_{11} = |\{\ell \in [L] | \tilde{b}(\ell) = 1 \text{ AND } X_j(\ell) = 1\}|, \tag{17}$$

$$n_{01} = |\{\ell \in [L] | \tilde{b}(\ell) = 0 \text{ AND } X_j(\ell) = 1\}|. \tag{18}$$

This decoder is derived from models that are more realistic in which $\pi_c \approx \pi_{c-1} \approx \pi$, so that the score of item j only takes into account the $(n_{11} + n_{01})$ symbols in which $X_j(\ell) = 1$ (*i.e.*, at most K symbols over L).

6.2 Joint decoder

In practice, a joint decoder computes from the observed BLIPed filter a score for any profile $P' \in \mathcal{P}$, which reflects the likelihood that P' is the true profile. This score is computed by taking into account L symbols but the complexity of a joint decoder is proportional to $|\mathcal{P}|$ (*i.e.*, $O(N^c)$), which is computationally expensive. Yet, there exists at least three possible approaches that approximate joint decoding with a reasonable complexity: 1) Monte Carlo Markov Chain (MCMC) [19, 20], 2) Belief Propagation Decoder [21] and 3) Joint Iterative Decoder [22].

In this paper, we investigate the first approach. The MCMC decoder is based on two key ideas. First, it receives as input an observed BLIPed filter $\tilde{\mathbf{b}}$ and then creates a Markov Chain that will be used to sample profiles according to the posterior distribution $\mathbb{P}[P|\tilde{\mathbf{b}}]$. This sampling requires a burn-in period after which the Markov Chain has converged. Once this convergence has occurred, it samples profiles with the targeted posterior distribution. During a second phase, some profiles are sampled and statistics are computed such as the marginal a posteriori distribution $\hat{\mathbb{P}}[j \in P|\tilde{\mathbf{b}}]$ that item j belongs to the true profile.

Posterior distribution. The objective is to sample profiles according to the posterior distribution $\mathbb{P}[P|\tilde{\mathbf{b}}]$, which can be written as:

$$\mathbb{P}[P|\tilde{\mathbf{b}}] = \frac{\mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}}|P]\mathbb{P}[P]}{\mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}}]}. \tag{19}$$

In this equation, $\mathbb{P}[P]$ is the a priori probability of P. To simplify our presentation, we consider only the simple model exposed in Section 3.2. We denote by $|P|$ the size of profile P (*i.e.*, the number of his items), and we set by

$\mathbb{P}[P] = 0$ if $|P| \neq c$, and $1/|\mathcal{P}|$ otherwise. Any profile is equally likely provided it has exactly c items. When we use more realistic models in our experimental work, the prior will be substantially different. We denote by $\omega(\mathbf{B})$ the Hamming weight of a binary vector $\mathbf{B}$ (*i.e.*, the number of bits set to 1). The probability $\mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}}|P] = \mathbb{P}[\mathbf{N} = \mathbf{B}_P \oplus \tilde{\mathbf{b}}]$ has the following expression

$$\mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}}|P] = p_\epsilon^{\omega(\mathbf{B}_P \oplus \tilde{\mathbf{b}})}(1-p_\epsilon)^{L-\omega(\mathbf{B}_P \oplus \tilde{\mathbf{b}})}. \tag{20}$$

The evaluation of the last quantity $\mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}}]$ in (19) is more involved:

$$\mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}}] = \sum_{P \in \mathcal{P}} \mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}}|P]\mathbb{P}[P]. \tag{21}$$

It requires a screening of $\mathcal{P}$, which is intractable for large c and N, which is why we will rely on the Markov chain.

Markov Chain. A Markov Chain is an iterative process with an internal state (*i.e.*, a profile in our case) taking value $P^{(t)}$ at iteration t. The next iteration draws a new state $P^{(t+1)}$ according to a transition probability distribution $\mathbb{P}[P^{(t+1)}|P^{(t)}]$. The Markov Chain is initialized randomly at state $P^{(0)}$. The probability distribution of transitions is crafted with care to enforce a convergence of the distribution of sampled profile $P^{(t)}$ to the posterior $\mathbb{P}[P|\tilde{\mathbf{b}}]$ of (19) as $t \to \infty$ (see Section 6.3). In practice, the convergence occurs after the first T iterations, the so-called burn-in period. Once this period has passed, it means that the Markov Chain has forgotten its starting point (*i.e.*, the samples are now independent of $P^{(0)}$) and that the distribution of the sample profiles has converged.

Monte Carlo method. After the burn-in period, the Markov Chain keeps on sampling for M more iterations. The marginal a posteriori probabilities are then estimated by a Monte Carlo method, which computes the empirical frequency that item j is present in sample $P^{(t)}$:

$$\hat{\mathbb{P}}[j \in P|\tilde{\mathbf{b}}] = |\{t \in [T+1, T+M]|j \in P^{(t)}\}|/M. \tag{22}$$

From these estimations, several post-processing are possible such as:

- inferring the most likely items of the true profile by ranking them in decreasing marginal probabilities,
- reconstructing the profile as the set of items whose marginal probability is above a given threshold,
- reconstructing the profile as the set of items with highest marginal.

6.3 Transition probabilities

Algorithmic coding of a profile. Section 3.3 describes how to infer from the observed BLIP a maximum number $c_{\max}$ of items of the corresponding profile. In

this algorithm, we code a profile as a vector of $c_{\max}$ components taking values in $[N] \cup \{0\}$. Some of these components may take the value "0" meaning an "empty item", while the others have different values (*i.e.*, there is no pair of non-zero components with the same value). For instance, for $c_{\max} = 5$, $P = (0, 3, 2, 0, 4)$ represents the profile of 3 items: #2, #3 and #4.

We define $\mathcal{V}(P_0, i)$ as the neighborhood of profile P_0 in the following manner:

$$\mathcal{V}(P_0, i) = \{P \in \mathcal{P} | P(k) = P_0(k) \quad \forall k \neq i\}. \tag{23}$$

This neighborhood profile is the set of all profiles whose coding differs at most from the i-th component. Note that $P_0 \in \mathcal{V}(P_0, i)$. If $P_0(i) = 0$, this neighborhood comprises profiles having at most one more item. Otherwise if $P_0(i) > 0$, this neighborhood contains profiles having at most one different item (*i.e.*, $P_0(i)$ is substituted by another item) and one profile having one less item (*i.e.*, item $P_0(i)$ is substituted by 0, the "empty item").

Multi-stage Gibbs sampling. Instead of computing the transition probabilities for all the possible profiles, we restrict the transitions to the neighborhood of the actual state. At the iteration $t+1$, an integer i is first uniformly drawn in $[c_{\max}]$ that indicates the subset $\mathcal{V}(P^{(t)}, i)$. Then, the following transition probability distribution is computed: $\forall P \in \mathcal{V}(P^{(t)}, i)$

$$\mathbb{P}[P^{(t+1)} = P | P^{(t)}] = \frac{\mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}} | P] \mathbb{P}[P]}{\sum_{P' \in \mathcal{V}(P^{(t)}, i)} \mathbb{P}[\tilde{\mathbf{B}}_{P'} = \tilde{\mathbf{b}} | P'] \mathbb{P}[P']} \tag{24}$$

Iteration $t+1$ ends by randomly drawing state $P^{(t+1)}$ from this distribution.

This choice of probabilistic transitions is called a multi-stage Gibbs sampler with random scan [23, Alg. A.42]. It guarantees that the law of sampled profiles converges to the stationary distribution $\mathbb{P}[P|\tilde{\mathbf{b}}]$, which legitimates our approach [23, Sect. 10.2.1]. The unknown multiplicative constant $\mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}}]$ in (19) has disappeared in the ratio. This transition probability distribution only depends on the priors $\mathbb{P}[P]$ (which depends on the mathematical model of a profile), and the conditional probabilities $\mathbb{P}[\tilde{\mathbf{B}}_P = \tilde{\mathbf{b}} | P]$ (which depends on the privacy-preserving mechanism). For instance, for the JLT mechanism, $\mathbb{P}[\tilde{\mathbf{Y}}_P = \tilde{\mathbf{y}} | P] \propto \exp(-\|\tilde{\mathbf{y}} - \sum_{j \in P} \mathbf{X}_j\|^2 / 2\sigma^2)$.

7 Experiments

7.1 Setup

In this section, we test the inference attacks designed on two real datasets: Digg and MovieLens. The Digg dataset has been collected on a social news aggregator and the profile of a user is composed of the news he has read. The MovieLens dataset is a snapshot from a movie recommendation site and in this dataset the profile of a user is composed of the movies he likes. For the experiments, we split both datasets into two parts : the training set and the testing set. The

Table 1. Datasets characteristics

	Nb of users	Training set size	Testing set size	N	c_{avg}	Sparsity %
Digg	531	331	200	1237	317	25.63%
MovieLens	943	600	343	1682	106	6.30%

characteristics of these datasets are summarized in Table 1, in which c_{avg} is the average number of items per profile and sparsity is the average occupancy of items among the user profiles.

During the experiments, we assume that the adversary has access to some raw profiles of users to estimate the item priors (*i.e.*, popularities of items). This is similar to assuming that the adversary has access to some global information about the general distribution of items in the population. We rely on the training dataset for computing the frequencies of items while the testing dataset is used solely for evaluating the performance of the attacks. In terms of parameters, for BLIP we set the number of hash functions $K = 20$ and the number of bits of the representation to $L = 5,000$. The values of ϵ are from the set $\{59, 28, 17, 8, 6, 5, 3, 2, 0\}$, which equivalently translate to the corresponding flipping p_ϵ from the range $\{0.05, 0.2, 0.3, 0.4, 0.42, 0.44, 0.46, 0.48, 0.5\}$. For the JLT scheme, we set the size of the representation L to $1,000$. L is set to a lower value as the representation, a dense real vector, is richer than the binary version of BLIP. The privacy parameter ϵ takes value in the set $\{600, 6, 3, 2, 1, 0.75, 0.5, 0.25, 0.1\}$, which translates into a noise level σ in $\{0, 1, 2, 3, 6, 8, 12, 24, 61\}$.

For MCMC, we used a burn-in period of $T = 1,000$ samples and estimation sample size of $M = 19,000$ for all the experiments. In practice, we observed that the performance is not very sensitive to the burn-in period length. As with other MCMC based approaches proper initialization for sampling is highly desirable for a faster convergence to the stationary distribution. We used the input public representation of the profile to estimate $\hat{c}$ and started with $\hat{c}$ random items. A poor estimation of $\hat{c}$ has to be traded-off with a longer burn-in period. We also prefilter items that are to be tested against the public profile for joint decoder, to reduce the search space. To realize this, we first predict the $f \times \hat{c}$ most probable items for a given profile ($f \in [2, 6]$) using single decoder and then run the joint decoder on the filtered items to return $\hat{c}$ items. This prefiltering decreases significantly the running time of the algorithm without impacting the prediction as only unlikely items will not be considered by the joint decoder.

7.2 Reconstruction attacks

We benchmark four attacks that produce a score per item:

- The single decoder described in [1].
- The popularity-based attack in which the score of an item is its prior estimated from the training data, independent of the given public representation.

– Our MCMC joint decoder with and without priors (*i.e.*, with flat priors) in which the scores are the estimated marginal a posteriori probabilities.

Reconstruction $\hat{P}$ is then the list of the top $\hat{c}$ items ranked based on their scores.

We measure the performance of a reconstruction attack by computing the cosine similarity between the reconstruction $\hat{P}$ and the true profile P as expressed in (25) for all the profiles of the testing set.

$$\cos(P, \hat{P}) = \frac{|P.\hat{P}|}{|P||\hat{P}|} \tag{25}$$

Afterwards, we compute the following statistics: average, the 10% and the 90% quantiles of the cosine similarities.

The plots in Figure 1 show that the performance of the reconstruction attack is better for high values of ϵ while it degrades as $\epsilon \to 0$. In this case, $p_\epsilon \to 0.5$ and every profile becomes equiprobable so that inferring the original profile becomes impossible. In addition, $\hat{c}$ depends on ϵ and low value results in a poor estimation of $\hat{c}$, which impacts the similarity measure as only top $\hat{c}$ items of the prediction is considered in the reconstructed profile. As the estimation of $\hat{c}$ is performed similarly for all the four attacks, the performance drop is common to all of them. Overall the performance of our MCMC attack is better than the single decoder of [1] for almost all ϵ values over the two datasets. Another way to see this is to find the range of ϵ in which a given attack performs worse than the baseline (*i.e.*, the popularity-based attack). For instance, by setting $\epsilon = 8$, the designer is sure that the single attack is no longer a threat. However, a skilled adversary can reconstruct almost 50% of the profile thanks to our MCMC attack.

Taking into account the prior of items improves the efficiency in the reconstruction significantly, provided that the estimation is reliable. This improvement is clearly observed on the MovieLens dataset. As for the Digg setup, priors of the training set do not generalized to the test set, hence they do not help much. We conducted the same experiment with the JLT scheme. The figure is not included in the paper due to a lack of space, but the results that we obtained are very close from the one of BLIP and thus we can draw the same conclusions.

7.3 Identifying the presence of an item

When ϵ is very small, Figure 1 clearly shows that the adversary cannot hope to reconstruct the full profile. In this situation, we evaluate the prediction of top R items, with $R \ll c$, as another assessment of the privacy guarantees. The success is measured in terms of the mean Average Precision at R (mAP@R) given in (26), which is the mean over the Q profiles in the test dataset of the average of the precisions at rank $1 \leq r \leq R$. The precision(r) refers to the fraction of correct items out of the top r predicted items. The mAP is sensitive to the order of the correct results and is a better gauge of the quality of a ranking.

$$\text{mAP@}K = \frac{1}{Q} \sum_{q=1}^{Q} \left(\frac{1}{R} \sum_{r=1}^{R} \text{precision}_q(r) \right). \tag{26}$$

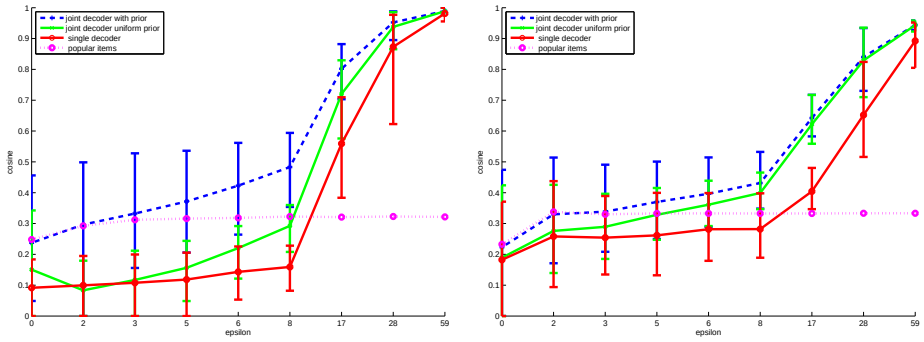

Fig. 1. Values of the cosine similarity (average, 10% quantile and 90% quantile) of BLIP for MCMC with prior, with no prior and single decoding for various ϵ on Movielens (left) and Digg (right) dataset.

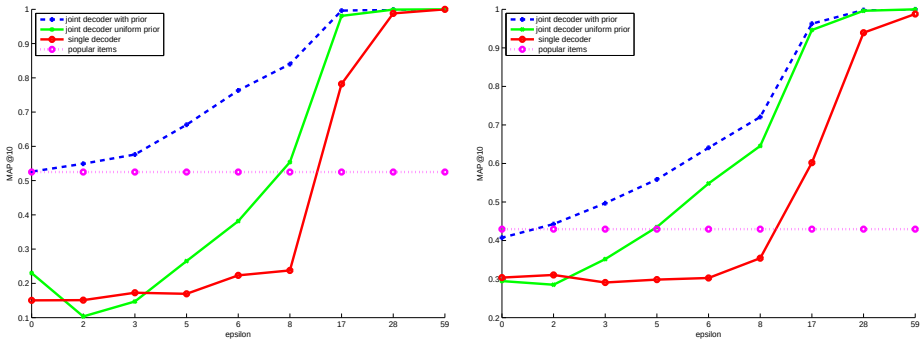

Fig. 2. Mean Average Precision for $R = 10$ for BLIP for MCMC with prior, with no prior and single decoding for various ϵ on Movielens (left) and Digg (right) dataset.

The characteristics of mAP@R depicted in Figure 2 are almost similar to the exact reconstruction measurement. Even if the exact reconstruction of profile is hardly possible for a given ϵ, predicting the top R items work. For instance, the maximum reconstruction for $\epsilon = 0$ for Movielens is 0.23 whereas the mean average precision is close to 0.5. The same conclusion holds for the Digg dataset.

7.4 Utility-privacy trade-off

Finally, we also studied the achievable trade-off between privacy and utility. Since BLIP and JLT are used for similarity estimation, we quantify the utility in terms of the recall, which is defined as the probability of identifying the k-nearest neighbors (we set $k = 10$ in our experiments). In this experiment, we measure privacy as $1 - \cos(P, \hat{P})$ (see (25)) based on the joint decoder. Figure 3 illustrates the utility-privacy trade-off obtained for various ϵ. The trade-off is almost similar on the two datasets. The privacy preserving properties of JLT transform is slightly better than BLIP, at least for the parameters we used in our

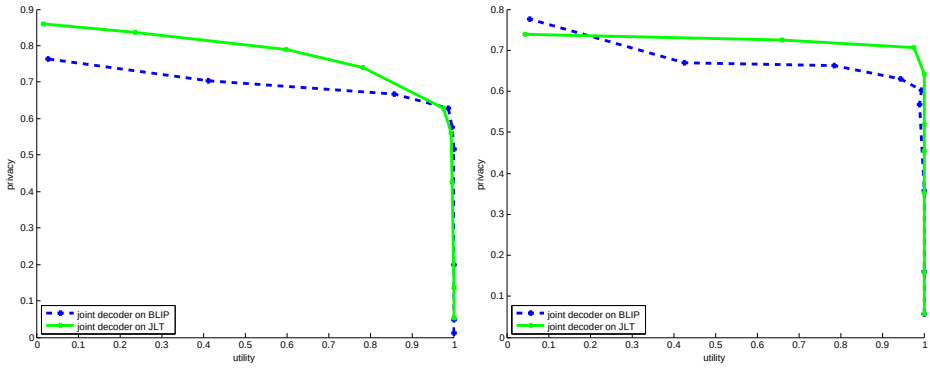

Fig. 3. Utility against privacy for BLIP and JLT for various ϵ on Movielens (left) and Digg (right) datasets.

simulation. This difference in performance is due partially to the representation superiority of dense real vector over binary vector. However, BLIP offers a more compact representation of the profile ($5,000$ bits versus $1,000$ scalars). The plot is helpful in fixing ϵ giving good utility without compromising much on privacy.

8 Conclusion

In differential privacy, the trade-off between utility and privacy is set by the parameter ϵ. However, being able to choose an appropriate value for this parameter is still an open research question, which has not been deeply investigated, with a few exceptions [24, 25]. In this paper, we have made a step forward to answer this question by proposing two generic inference attacks, namely single and joint decoding, whose objective is to reconstruct the profile of a user out of a differentially-private representation produced through a non-interactive mechanism. The first inference attack decides of the presence of a single item and sequentially explores all the item set, while the latter strategy decides whether a subset of items is likely to be the user profile and considers all possible subsets.

We have evaluated the effectiveness of the attack on two schemes producing differentially private representations: BLIP (BLoom-and-flIP) [1] and JLT (Johnson-Lindenstrauss Transform) [2]. Our theoretical analysis as well as the experimental results clearly shows that joint decoding is more powerful than single decoding. Overall, we believe that this attack helps better understanding the privacy guarantees offered by a wide class of differentially-private mechanisms (interactive or not) as well as for the privacy practitioner to tune experimentally ϵ to ensure the maximum utility without compromising much on privacy.

References

1. Alaggan, M., Gambs, S., Kermarrec, A.M.: BLIP: Non-interactive Differentially-Private Similarity Computation on Bloom Filters. In: 14th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems, Toronto, Canada (2012)

2. Kenthapadi, K., Korolova, A., Mironov, I., Mishra, N.: Privacy via the johnson-lindenstrauss transform. arXiv preprint arXiv:1204.2606 (2012)
3. Dwork, C.: Differential privacy. In: Int. Conf. on Automata, Languages and Programming. Volume 4052 of LNCS., Springer (2006) 1–12
4. Liu, K., Giannella, C., Kargupta, H.: A survey of attack techniques on privacy-preserving data perturbation methods. In: Privacy-Preserving Data Mining. Volume 34 of Advances in Database Systems. Springer (2008) 359–381
5. Chen, K., Liu, L.: A survey of multiplicative perturbation for privacy-preserving data mining. In: Privacy-Preserving Data Mining. Springer (2008) 157–181
6. Guo, S., Wu, X.: On the use of spectral filtering for privacy preserving data mining. In: ACM Symp. on Applied Computing. (2006) 622–626
7. Huang, Z., Du, W., Chen, B.: Deriving private information from randomized data. In: ACM SIGMOD Int. Conf. on Management of data, ACM (2005) 37–48
8. Guo, S., Wu, X.: Deriving private information from arbitrarily projected data. In: Advances in Knowledge Discovery and Data Mining. Volume 4426 of LNCS., Springer (2007) 84–95
9. Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms. In: 20th ACM SIGMOD-SIGACT-SIGART Symp. on Principles of database systems. (2001) 247–255
10. Diaconis, P., Sturmfels, B.: Algebraic algorithms for sampling from conditional distributions. The Annals of Statistics **26**(1) (Feb. 1998) 363–397
11. Dobra, A.: Measuring the disclosure risk for multi-way tables with fixed marginals corresponding to decomposable log-linear models. Technical report (2000)
12. Williams, O., McSherry, F.: Probabilistic inference and differential privacy. In: Advances in Neural Information Processing Systems. (2010) 2451–2459
13. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Theory of Cryptography. Volume 3876 of LNCS., Springer (2006) 265–284
14. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: Privacy via distributed noise generation. In: EUROCRYPT. (2006) 486–503
15. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: IEEE Symposium on Foundations of Computer Science. (2007) 94–103
16. Beimel, A., Nissim, K., Omri, E.: Distributed private data analysis: on simultaneously solving *how* and *what*. In: Proc. of Advances in Cryptology. (2008) 451–468
17. Li, Y.D., Zhang, Z., Winslett, M., Yang, Y.: Compressive mechanism: utilizing sparse representation in differential privacy. CoRR **abs/1107.3350** (2011)
18. Moulin, P.: Universal fingerprinting: capacity and random-coding exponents. **arXiv:0801.3837** (January 2008)
19. Knill, E., Schliep, A., Torney, D.C.: Interpretation of pooling experiments using the Markov chain Monte Carlo method. J. Comput. Biol. **3**(3) (1996) 395–406
20. Furon, T., Guyader, A., Cerou, F.: Decoding fingerprints using the Markov Chain Monte Carlo method. In: IEEE Int. Work. on Information Forensics and Security (WIFS). (2012) 187–192
21. Sejdinovic, D., Johnson, O.: Note on noisy group testing: asymptotic bounds and belief propagation reconstruction. In: Proc. 48th Allerton Conf. on Commun., Control and Computing, Monticello, IL, USA (October 2010) arXiv:1010.2441v1.
22. Meerwald, P., Furon, T.: Toward practical joint decoding of binary Tardos fingerprinting codes. IEEE Trans. on Inf. Forensics and Security, **7**(4) (2012) 1168–1180
23. Robert, C., Casella, G.: Monte Carlo statistical methods. Springer Verlag (2004)
24. Lee, J., Clifton, C.: How much is enough? Choosing ϵ for differential privacy. In: Information Security. Volume 7001 of LNCS., Springer (2011) 325–340

25. Alvim, M.S., Andrés, M.E., Chatzikokolakis, K., Palamidessi, C.: On the relation between differential privacy and quantitative information flow. In: Int. Conf. on Automata, Languages and Programming. (2011) 60–76

9 Appendix A: BLIP mechanism

9.1 Single and joint decoding

We have $I(\tilde{\mathbf{B}}_P; \mathbf{X}) = H(\tilde{\mathbf{B}}_P) - H(\tilde{\mathbf{B}}_P|\mathbf{X})$, in which H is the (Shannon) entropy of a random variable. With the simple model detailed in Section 3.2, we get that

$$I(\tilde{\mathbf{B}}_P; \mathbf{X}) = L(h_b(\tilde{\pi}_c) - (1-p)h_b(\tilde{\pi}_{c-1}) - ph_b(p_\epsilon)), \tag{27}$$

with $h_b(p)$ the entropy of a Bernoulli distribution $\mathcal{B}(p)$ (in hats):

$$h_b(p) \triangleq -p\log(p) - (1-p)\log(1-p) = h_b(1-p). \tag{28}$$

The probabilities $\tilde{\pi}_c$ and $\tilde{\pi}_{c-1}$ appear in (27) because we assume that the profiles are of identical size c. When considering more complex but also more practical models, this difference vanishes as $\tilde{\pi}_c$ and $\tilde{\pi}_{c-1}$ are replaced by $\tilde{\pi}$:

$$I(\tilde{\mathbf{B}}_P; \mathbf{X}) \approx Lp(h_b(\tilde{\pi}) - h_b(p_\epsilon)). \tag{29}$$

As for the joint decoding, Bloom filter being a deterministic process, we write:

$$\begin{aligned} I(\tilde{\mathbf{B}}_P; P) &= I(\tilde{\mathbf{B}}_P; \mathbf{B}_P) = H(\tilde{\mathbf{B}}_P) - H(\tilde{\mathbf{B}}_P|\mathbf{B}_P) \\ &= H(\tilde{\mathbf{B}}_P) - H(\mathbf{N}) = L(h_b(\tilde{\pi}_c) - h_b(p_\epsilon)). \end{aligned} \tag{30}$$

9.2 Comments

Example. Figure 4 (left) shows $I(\tilde{\mathbf{B}}_P; P)/c$ as a function of c and L. From a particular (c, L), we set

$$K = \left\lfloor \log(2)\frac{L}{c} \right\rfloor, \tag{31}$$

which is the recommended number of hash functions in Bloom filter design, and we apply the model of Section 3.2 with $\epsilon = 20$. For a given c, too small L means too few observed symbols for reliably estimating the profile. Too large L implies a big K and therefore, p_ϵ tends to $1/2$ according to (4). Figure 4 (right) shows that $I(\tilde{\mathbf{B}}_P; P)/c - I(\tilde{\mathbf{B}}_P; \mathbf{X})$ can be substantial: a joint decoding allows the adversary to tackle up to 3.5 (*i.e.* $e^{1.25}$) times more items.

Upper bounds. As $\epsilon \to 0$, $p_\epsilon \to 1/2$ as well as $\tilde{\pi}$, so that $I(\tilde{\mathbf{B}}_P; \mathbf{X}) \to 0$ and also $I(\tilde{\mathbf{B}}_P; P) \to 0$. When $\epsilon = 0$, observing the BLIP is useless since it brings no information. In this situation, neither the single nor the joint decoding can do anything. We can bound the quantity in common in both expressions as follows:

$$\begin{aligned} h_b(\tilde{\pi}_c) - h_b(p_\epsilon) &\leq \log(2) - h_b(p_\epsilon) \leq \log(2) - \log\left(1 + e^{\epsilon/K}\right) + \frac{\epsilon}{K}\frac{e^{\epsilon/K}}{1+e^{\epsilon/K}} \\ &\leq \frac{\epsilon}{K}\frac{e^{\epsilon/K}}{1+e^{\epsilon/K}} \leq \frac{\epsilon}{K}. \end{aligned} \tag{32}$$

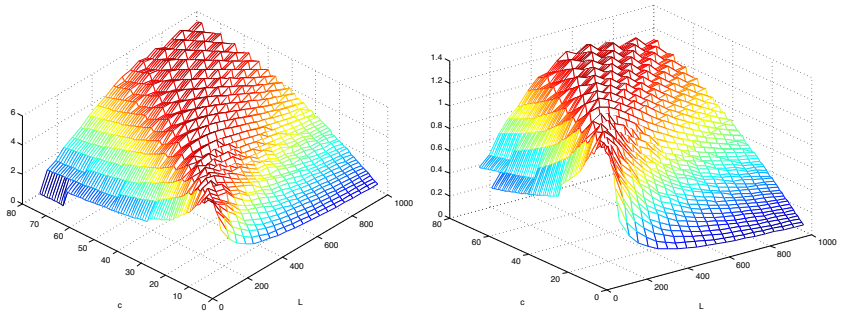

Fig. 4. (Left): Mutual information of the joint decoder $I(\tilde{\mathbf{B}}_P; P)/c$ in nats as a function of (c, L). (Right): Difference $I(\tilde{\mathbf{B}}_P; P)/c - I(\tilde{\mathbf{B}}_P; X)$ in nats as a function of (c, L).

Typical Bloom filter setup. Figure 4 shows that estimating an important number of items is possible provided that L grows linearly with c. Indeed, it is also common practice in the design of Bloom filter to set:

$$L = \left\lceil -c \frac{\log(P_{fp})}{(\log 2)^2} \right\rceil, \tag{33}$$

in which P_{fp} is the probability of false positive of the Bloom filter (*i.e.*, to detect the presence of an item not belonging to P). Inserting (31) and (33) in the expression of the mutual informations, we get quantities independent of c:

$$\frac{1}{c} I(\tilde{\mathbf{B}}_P; P) \sim -\frac{\log(P_{fp})}{\log(2)} \left(1 - \frac{1}{\log(2)} h_b \left((1 + 2^{\frac{\epsilon}{-\log(P_{fp})}})^{-1} \right) \right), \tag{34}$$

$$I(\tilde{\mathbf{B}}_P; \mathbf{X}) \sim \log(2). \frac{1}{c} I(\tilde{\mathbf{B}}_P; P). \tag{35}$$

This shows that if the Bloom filter is properly designed, the power of the attack does not depend on c but solely on the values of $-\log(P_{fp})$ and ϵ. Moreover, the joint decoder is $1/\log(2) \sim 1.44$ more "powerful" than the single decoder.

10 Appendix B: JLT mechanism

The same analysis holds for the JLT representation described in Section 4. The main difference lies in the fact that we manipulate differential entropies because the JLT representation is a real vector. The quantities at stake respectively for the single and joint decoders are upper bounded, thanks to conditions (10)

$$I(\tilde{\mathbf{Y}}_P; \mathbf{X}) = \frac{L}{2} \log \left(1 + \frac{1}{(c-1) + L\sigma^2} \right) \leq \frac{\epsilon}{32 + 2\epsilon(c-1)L}, \tag{36}$$

$$\frac{I(\tilde{\mathbf{Y}}_P; P)}{c} = \frac{L}{2c} \log \left(1 + \frac{c}{L\sigma^2} \right) \leq \frac{\epsilon}{32}, \tag{37}$$