"Better than Nothing" Privacy with Bloom Filters: to what extent?

Giuseppe Bianchi, Lorenzo Bracciale, and Pierpaolo Loreti

DIE, Universitá di Roma "Tor Vergata", Rome, Italy
{name.surname}@uniroma2.it

Abstract. Bloom filters are probabilistic data structures which permit to conveniently represent set membership. Their performance/memory efficiency makes them appealing in a huge variety of scenarios. Their probabilistic operation, along with the implicit data representation, yields some ambiguity on the actual data stored, which, in scenarios where cryptographic protection is unviable or unpractical, may be somewhat considered as a *better than nothing* privacy asset. Oddly enough, even if frequently mentioned, to the best of our knowledge the (soft) privacy properties of Bloom filters have never been explicitly quantified. This work aims to fill this gap. Starting from the adaptation of probabilistic anonymity metrics to the Bloom filter setting, we derive exact and (tightly) approximate formulae which permit to readily relate privacy properties with filter (and universe set) parameters. Using such relations, we quantitatively investigate the emerging privacy/utility trade-offs. We finally preliminary assess the advantages that a *tailored* insertion of a few extra (covert) bits achieves over the commonly employed strategy of increasing ambiguity via addition of random bits.

1 Introduction

A query to scholar.google.com reveals that (almost literally!) a myriad of papers have employed Bloom filters or their extensions in a variety of largely diverse scenarios. This is all but a surprise. Bloom filters are compact and computationally efficient - O(1) - probabilistic data structures devised to conveniently permit membership queries, and provide straightforward support for set operations such as union, intersection, inclusion, etc. Invented in 1970 [1] for spell checking, they have found several applications in database systems since the beginning of the eighties [2–5], and, more recently, they have been applied to a variety of networking scenarios [6], collaborative P2P and distributed computing systems [7, 8], genomics [9], and so on.

A Bloom filter is an array $B[.]$ of m bits. Data, in most generality treated as a string (e.g. a name, an email, an IP address, etc), is encoded in the filter by taking k hash functions having digest within the filter size m, and by setting the relevant positions in the bit array $B[.]$. In order to query if a data item is stored in the filter, it suffices to check whether all the k hash functions, taken over the considered data, point to bits set to 1 in the filter. In Bloom filters, false positive may occur when a query for a data *not originally stored* in the filter nevertheless "*hits*" all bits already set to 1. The false positive probability is the price to pay for space/time efficiency, and in any case can be easily controlled by suitably designing the filter parameters m and k.

Privacy with Bloom filters. Actually, there are several scenarios where the implicit representation of the stored data, and the false positive's ambiguity of a Bloom filter's response, may be considered an asset rather than an annoyance. For instance, Alice is a social network subscriber whose list of friends is made public through a Bloom filter[1]. Alice may feel her privacy to be somewhat protected (taking aside, for the moment, whether this is true or not) by two apparent "facts". First, Alice's list of friends is not provided in clear text. Hence, the only way to ascertain whether a specific person, say Ernest, is in her set of friends is to explicitly query for him in the Bloom filter. Second, even if a query for Ernest returns a positive answer, Alice can *deny* that this is the case, blaming a false positive.

In practice, the protection offered by the *first* "fact", namely the implicit data representation due to one-way hashing, is questionable in the sheer majority of real-world scenarios, and more precisely when the universe set is easily *enumerable*. An attacker armed with sufficient computational capability to enumerate the entire universe set may perform a check on each possible element, and thus reconstruct the filter's content, besides the ambiguity due to false positives. Indeed, the cases in which enumeration is *not* feasible are the exception, rather than the norm. When used to store IP addresses, a Bloom filter enumeration would take 2^{32} checks, a far from prohibitive number; Facebook, the world's largest social network, totalizes as of today about 900 million accounts, less than 2^{30}, and the same magnitude applies to US 9-digit social security numbers. And, at last, an attacker may exploit side knowledge to dramatically narrow down the "Candidate Universe set" to explore within[2].

Whenever a Bloom filter must come along with *strong* privacy protection, cryptographic extensions are available [10–14]. However, these solutions bring about the burden of distributing and/or managing relevant cryptographic material (e.g., keys), which hence restrict their usability. So, when a cryptographic scheme is unviable or inconvenient, and enumeration attacks are feasible, we are left with just the *second* above mentioned "fact", namely the ambiguity given by the false positives as a soft, *better than nothing*, form of privacy protection.

Quite surprisingly, at least to the best of our knowledge, no prior work appears to have addressed the very natural emerging question: *can we quantify the privacy preservation capabilities of a standard Bloom filter?*

False positives or *hiding set* size? Actually, as shown in the remainder of the paper, such a quantification is not difficult, but requires some minimal attention in avoiding pitfalls, some obvious, some perhaps more subtle. Assume that a Bloom filter contains n elements, and its parameters are set so as to accomplish a false positive probability

[1] As indeed done in a former social network called LOAF, today not anymore active.

[2] For instance, despite the 900 million Facebook total accounts, an attacker can restrict enumeration to a target country (e.g. countries such as Austria or Ireland have slightly more than 2 million accounts each) or city. Similarly, before the "Social Security Number randomization" introduced on June 25, 2011, the first three out of the 9 SSN digits had geographical significance, and thus queries for citizens in a given geographical region would have trivially reduced to an universe set smaller than one million.

ψ. It is straightforward to see that, by itself, ψ is *not*, alone, an appropriate metric to measure privacy.

Indeed, roughly speaking, privacy stems from the inability of an attacker to distinguish an element out of the n stored in the filter, from elements that *appear* inserted, but which are not. But the number v of such elements, which we descriptively refer to as *hiding set*, not only depends on the false positive probability, but also on the size N_u of the (Candidate) Universe set through the obvious (average) relation $v = (N_u - n) \cdot \psi$. For instance, compare two filters A and B including $n = 100$ elements each, and having false positive probabilities set to $\psi_A = 10\%$ and $\psi_B = 1\%$, respectively. If the universe sets have different size, say $N_{u,A} = 1600$ and $N_{u,B} = 20100$, then filter B may exhibit a larger *hiding set* than A despite the much smaller false positive (in the example, $v_A = 150$ versus $v_B = 200$). In essence, we can conclude that, as in the case of utility [15], also in the case of privacy the cardinality of the universe set plays a crucial role *in combination* with the filter's false positive.

It could then be argued that the cardinality of the *hiding set* might be considered as a suitable privacy metric. This is closer to the truth, but in a quite subtle way. Indeed, consider for instance a filter having an *hiding set* whose cardinality is twice the number of inserted elements. This could roughly suggest that the elements in the set are 3-anonymous (two "covert" false positive elements for each true filter's element, following K-anonymity definition [16]). However, we will show that, in such setting, up to one third of the elements *won't be anonymous at all*!

Our contribution. The contribution of this paper is threefold. i) We introduce ***privacy metrics suitable for Bloom filters***. Specifically, we cast the K-anonymity model [16] in its probabilistic acceptation [17], to the Bloom filter setting. For the case K=2, we give a dedicated name, γ-deniability (equivalent to our more general definition of γ-2-anonymity), and treatment, as we believe it is of particular interest in several practical applications. ii) We ***quantify, with both exact and tightly approximated formulae***, γ-deniability and γ-K-anonymity. We leverage such metrics to determine to what extent a standard (optimized) Bloom filter configuration yields privacy performance, and which (very limited) room a filter designer has in exploiting suboptimal configurations (for a same false positive target) for improving privacy. iii) As it ultimately appears that substantial privacy improvements may be accomplished only by increasing the *hiding set* size, we investigate, using a preliminary heuristic, the advantages that ***a tailored insertion of supplementary filter bits*** may accomplish with respect to the customary practice of adding random bits.

1.1 Related Works

Bloom filters and their several modifications have been deeply studied and applied in many scenarios. Broad surveys are available [6, 8], tackling both their basic design aspects as well as their widespread application.

Cryptographic extensions of Bloom filters have emerged in Private Information Retrieval and Searchable Encryption. Designs target *strong* privacy requirements, at the price of key management for querying and/or constructing the Bloom filter index. Cryptographic techniques employed are many: trapdoors in the Bloom filter's hash functions

[12]; composability of the Pohlig-Hellman cipher [10, 11]; blind signatures and oblivious pseudorandom functions [13]; Boneh-Goh-Nissim public key encryption [14]; etc.

Several scenarios employ **plain Bloom filters to exchange private information**. In several cases, privacy is not quantified, but is generically mandated to the Bloom filter one-way hashing with limited further argumentation [18–20] (although in this latter work a large universe set is mentioned). In other cases, such as [21], privacy is accomplished by not admitting repeated queries (i.e., enumeration) and by setting the Bloom filter parameters so as to achieve enough false positives. Still with reference to plain Bloom filters, the work [22], dealing with the sharing of payload information across Distributed Intrusion Detection systems, is among the few which somewhat quantify privacy, although this is done indirectly, through the quantification of the large universe set involved in the specifically considered application, and the large number of possible n-grams per filter bit.

Some works use Bloom filters but introduce **supplementary non cryptographic procedures to improve their privacy level**. This is the case of [23], which proposes to index documents from multiple providers organized into disjoint privacy groups, and devises an iterative procedure which uses randomized Bloom filters to produce a privacy preserving index. Another approach, proposed in [24], consists in splitting a Bloom filter into segments distributed across multiple participants; it is suggested that the higher false probability in a segment improves privacy.

An **attack devised to revert privacy** has been described in [25]. This work analyzes an earlier proposed system [26] based on Bloom filters for string comparison in private record linkage, and shows how to extract significant amount of private information through a Constraint Satisfaction Cryptanalysis. Note that in some scenarios, the ability to invert a Bloom filter comes as a functional advantage: indeed [27] proposes a Bloom filter enhancement where extra information permits to list the complete filter's content with high probability.

Concerning **privacy metrics specifically devised for Bloom filters** we are not aware of any prior specific work. Of course, a huge amount of papers have tackled privacy and anonymity definitions in the much more general database setting, such as K-anonymity [16] and probabilistic K-anonymity [17], L-diversity [28], T-closeness [29], Differential Privacy [30], etc. In this work we cast K-anonymity, in its probabilistic version, to the Bloom filter setting (with special attention to the practically more interesting case $K = 2$ which we conveniently define as deniability), and we provide explicit formulae to measure it.

2 Preliminaries

This section reviews background information on Bloom filters, formally introduces the notion of *hiding set*, and derives a combinatorial *balls and bins* result used in the remainder of the work. For the reader's convenience, Table 1 summarizes notation.

2.1 Bloom Filters

A Bloom filter [1, 6] is a probabilistic data structure used to represent set membership. A Bloom filter is implemented as an array $B[i], i \in (1, m)$, of m bits accessed via k

Notation	Meaning
$BF(\mathcal{S})$	Bloom filter storing a set $\mathcal{S}$
$\mathcal{S}$	Set of elements inserted in the filter
$n = \lvert\mathcal{S}\rvert$	number of elements inserted in the Bloom filter
m	Size in bits of the Bloom filter
k	Number of used hash functions
$\psi(m, k, n)$	False positive probability for given filter parameters
$\mathcal{U}$	Set of elements in the universe, $\lvert\mathcal{U}\rvert = N_u$
$\mathcal{V}$	Hiding set (set of false positive elements), $\lvert\mathcal{V}\rvert = N_v$

Table 1. Used notation

independent hash functions $H_1(x)...H_k(x)$, each of which maps a string $x \in \{0,1\}^*$ to one of the m bits within the bit array.

Consider a set $\mathcal{S} = \{x_1, \cdots x_n\}$ of n elements. We denote with $BF(\mathcal{S})$ a Bloom filter whose initially empty array is filled with all the elements $x \in \mathcal{S}$, by repeating for every x the following **insert** procedure: $\forall j \in \{1..k\}, B[H_j(x)] \leftarrow 1$.

Querying the presence of an element $x \in \{0,1\}^*$ within a Bloom filter consists of computing $\wedge_{j=1}^{k} B[H_j(x)]$ (i.e., returning 1 only if *all* corresponding bits are 1).

In Bloom filters, false positives are possible but false negatives are not. A false positive $\psi(m, k, n)$ is the probability that a query performed for an element x *not* stored in $BF(\mathcal{S})$ returns 1, where the parameters m and k specify the Bloom filter (size of the bit array m, and number of hash functions k), and n is the cardinality of the stored set $\mathcal{S}$. Even if an exact expression for $\psi(m, k, n)$ is available [31], virtually all works in the field rely on a simple, but tight, approximation (see e.g., [6] for its derivation):

$$\psi(m, k, n) \approx \left(1 - [1 - 1/m]^{nk}\right)^k \approx \left(1 - e^{-nk/m}\right)^k. \tag{1}$$

With the exception of the derivation in Appendix A, when computing false positive probabilities we will thus conveniently resort to (1). Finally, in practical applications, the Bloom filter parameters m, k are frequently optimized for a given stored set size n, so as to minimize the false positive probability. We will briefly review the relevant relations in Section 4, while assessing the privacy properties of optimized Bloom filters.

2.2 Hiding set

As very clearly explained in a recent paper [15], and perhaps contrarily to some practitioners' belief, a Bloom filter's false positive probability is *not* the only metric which affects performance. In several (but not all) applications, performance is fundamentally affected by the *absolute number* v (versus the false positive's *fraction* ψ) of elements which *appear* included in the filter due to false positives, but which are not. For instance, [15] discusses the specific case of Bloom filters used as cache summary, and shows that when v gets significantly larger that the actual size of the cached set (which may be the case when the universe set is large), the usage of a Bloom filter may even make performance *worse* than when the Bloom filter is *not used at all* [15].

It is intuitive to expect that also the privacy properties of a Bloom filter primarily depend on the cardinality v of the set of elements which are apparently included in

the filter, but which are not. (But we will show later on that such dependence is not as obvious as it might seem). We thus give to such set the descriptive name *hiding set*.

More formally, let $\mathcal{U}$, with $|\mathcal{U}| = N_u$ be the "Candidate Universe set", i.e. the Universe set at the net of the elements that an attacker may *a-priori* get rid by using external information or other inference means (quantification being application specific, hence out of the scopes of this paper). Let $\mathcal{S} = \{x_1, \cdots x_n\}$ be a set of n elements, and $BF(\mathcal{S})$ be a Bloom filter with size m and k hash functions, filled with such n elements.

Definition 1. *A set $\mathcal{V}$ is called* Hiding Set *for a Bloom filter $BF(\mathcal{S})$ if $\mathcal{V}$ contains all the elements $v_i \in \mathcal{U}$ s.t. $v_i \notin \mathcal{S}$ and a query for v_i in $BF(\mathcal{S})$ returns 1 (i.e. v_i is a false positive).*

Remark 1. The number of elements in $\mathcal{V}$ is a random variable N_v, with (binomial) probability distribution

$$P\{N_v = v\} = \binom{N_u - n}{v} \psi(m,k,n)^v \left(1 - \psi(m,k,n)\right)^{N_u - n - v}$$

and mean value $E[N_v] = (N_u - n)\psi(m,k,n)$. This trivially follows from the fact that every element in the universe set not belonging to the set $\mathcal{S}$ experiences an independent false positive probability $\psi(m,k,n)$.

2.3 Probability of non empty bins

Throughout the paper, we will make frequent use of the following combinatoric *balls and bins* result.

Lemma 1. *Consider $u \geq 1$ bins and $z \geq 1$ balls. Each ball is independently placed in a randomly chosen bin. Let U be the random variable representing the resulting number of non empty bins. Then U has the following probability distribution:*

$$U_u(z;x) = \frac{\left\{ {z \atop x} \right\} \binom{u}{x} x!}{u^z}, \quad \forall x \in (1,u) \tag{2}$$

and mean value

$$E[U] = u\left(1 - \left(1 - \frac{1}{u}\right)^z\right). \tag{3}$$

The lemma is readily proven via a counting exercise[3]. We recall that the Stirling number of the second kind $\left\{ {z \atop x} \right\}$ expresses the number of ways to partition a set of z elements (labelled balls) into x non empty subsets (bins). $\binom{u}{x}$ yields the number of ways in which exactly x bins are chosen out of u total bins, and $x!$ is the number of ways we can label the chosen bins. Hence, the numerator in (2) provides the number of ways in which z

[3] Exercise which requires a bit of care: the underlying trap is to focus on multisets counts, i.e. *unlabeled* balls, and neglect the fact that multisets are *not* equiprobable; for instance, with two unlabeled balls and two urns, we have three possible multisets $\{\{*,*\},\{\}\}$, $\{\{*\},\{*\}\}$, $\{\{\},\{*,*\}\}$, whereas the number of equiprobable combinations is 4 (as a count on labelled balls indeed yields).

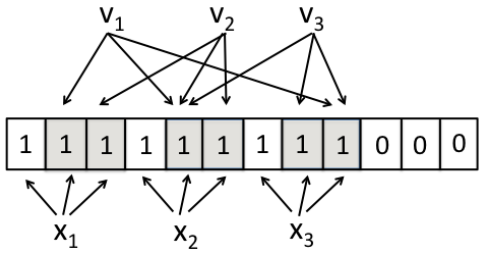

Fig. 1. A toy example where an hiding set of same cardinality as the set truly included in the filter does not provide any anonymity

labelled balls fall into exactly x bins out of u available ones. The probability distribution is finally derived by dividing for the total number u^z of ways to distribute z labelled balls across u labelled bins.

The mean value (3) might be eventually derived by direct computation, recalling that, using standard inclusion/exclusion arguments, Stirling numbers of the second kind are expressed as

$$\left\{ {z \atop x} \right\} = \frac{1}{x!} \sum_{i=0}^{x-1} (-1)^i \binom{x}{i} (x-i)^z.$$

But of course a much more convenient direct derivation consists in exploiting the basic fact that, also for *non independent* random variables X_i (indeed our case below), $E\left[\sum X_i\right] = \sum E\left[X_i\right]$. Hence it suffices to describe a single bin via the random variable $X_i \in \{0, 1\}$ which assumes value 1 when the bin is non empty; trivially note that $E[X_i] = \left(1 - \left(1 - \frac{1}{u}\right)^z\right)$, and multiply by the total number u of bins to obtain (3).

3 Privacy metrics

In this work, unlike for instance [10–14], we are *not* concerned with cryptographic extensions of Bloom filters. Hence, security/privacy requirements usually assumed in cryptography are not applicable to *standard* Bloom filters. Rather, suitable privacy metrics should ideally *cast* well established non-cryptographic privacy/anonymity metrics to the Bloom filter setting specificities.

A fairly natural starting point appears to be the notion of K-anonymity. However, the very first idea of defining K-anonymity through the mere cardinality of the actual stored set and the hiding set resulting from false positives is inappropriate. This is best shown by the toy example illustrated in Figure 1: here, 3 elements x_1, x_2, and x_3, are actually stored in a Bloom filter with $k = 3$ hash functions, whereas other 3 elements, v_1, v_2, v_3, are false positives. Since, for each true element, there is a false positive one, this could suggest a 2-anonymity level. But neither of the three elements x_1, x_2, x_3 are actually anonymous at all! Indeed, each of them is trivially identified as being the only candidate hashing over some filter bits (the 1st, 4th and 7th, respectively).

The above example rather suggests that there seems to be a closer than expected analogy between the Bloom filter scenario and that considered in [16], despite the fact that, here, we are focusing on atomic data (the set of elements), whereas [16] deals with

structured data organized into tuples of *attributes*. Indeed, we can intuitively interpret any element x in the universe set as being described by a tuple of $i \in (1, m)$ *boolean "attributes"* associated to each filter's bit $B[i]$. An element x has "attribute" $B[i]$ if and only if $H_j(x) = i$ for at least one $j \in (1, k)$ - i.e. if an only if one of the hash functions, if applied to the considered element x, would "hit" the Bloom filter's bit $B[i]$. It readily follows that [16]'s K-anonymity definition can be cast to the Bloom filter case. Informally, an element x actually included in the filter is K-anonymous if, for each of the bits $B[i]$ "hit" by the considered element, there are at least other $K - 1$ elements which *appear* included in the filter (because of false positives), and *which map to the same filter's bit*.

This said, it is immediate to see that a *deterministic* K-anonymity requirement may not always be attained. With a relatively small universe set, and a potentially large filter size (usually designed on the basis of the false probability target mandated by an application), the probability that a filter's bit is "hit" by a *unique* element in the whole universe set (and hence that such element may not be anonymized by *any* other possible element) may be far from negligible[4]. Rather, *probabilistic* extensions [17] of the original K-anonymity model appear more suited to the Bloom filter setting.

3.1 γ-deniability

Before taking further generalizations, we first focus on what we believe is the most compelling question: *to what extent an element inserted in a Bloom filter can be disclosed via enumeration*? Informally, for an element inserted in the filter, we use the descriptive attribute "deniable" whenever the owner of the filter can deny that the element is actually stored, blaming a false positive. Since only a *fraction* of inserted elements may be deniable, we resort to the following probabilistic definition.

Definition 2. *An element $x \in \mathcal{S}$ inserted in a Bloom filter $BF(\mathcal{S})$ is said to be* deniable *if $\forall i \in \{1..k\}$, there exists at least one hiding set element $v \in \mathcal{V}$, such that $\exists j \in \{1..k\}$ s.t. $H_i(x) = H_j(v)$. A Bloom filter configuration $BF(\mathcal{S})$ is γ-deniable (or, alternatively, we refer to such property as γ-deniability), whenever a randomly chosen element $x \in \mathcal{S}$ is deniable with probability γ.*

Note that our γ-deniability definition is on purpose restrictive: it imposes that "covert" elements must *not* belong to the original set $\mathcal{S}$, but must be drawn only from the hiding set $\mathcal{V}$. In other words, an element is deniable when it can be replaced with elements not originally stored in the filter, without changing the filter bitmap. Otherwise, it would be possible to have all set elements 1-deniable, but the set *as a whole* would not be deniable (e.g. think to a set of just two elements, hashing to the same filter bits, and no false positives).

[4] Indeed, the probability that a Bloom filter bit is used by a unique element in the universe set is (Poisson) approximated by $N_u k/m \cdot e^{-N_u k/m}$, being N_u the size of the universe set, k the number of hash functions and m the Bloom filter size. Using $k = 4$ and $m = 628$, namely the minimum filter size able to store 100 elements with a false probability target of 5%, even an universe set of 1000 elements would leave about 11 impossible to anonymize elements, irrespective of how the filter is filled.

	B[1]	B[2]	B[3]	B[4]	B[5]	B[6]	B[7]	B[8]	B[9]
x_1	1	0	1	0	0	0	0	1	0
x_2	0	0	1	1	0	0	0	1	0
x_3	0	0	0	1	0	1	0	0	1
BF(S)	1	0	1	1	0	1	0	1	1
v_1	1	0	1	1	0	0	0	0	0
v_2	0	0	1	0	0	1	0	1	0
v_3	1	0	0	0	0	1	0	1	0

Fig. 2. Bloom filter composed of elements $\langle x_1, x_2, x_3\rangle$ that admits 3 false positives: v_1, v_2, v_3

Figure 2 depicts an illustrative example. The set of elements inserted in the Bloom filter is $\mathcal{S} = \langle x_1, x_2, x_3\rangle$; the hiding set comprises the (false positive) elements $\mathcal{V} = \langle v_1, v_2, v_3\rangle$. The set element x_1 is deniable because its relevant filter bits $B[1], B[3], B[8]$ are covered by v_1 and v_2. Element x_3 is not deniable because the relevant bit $B[9]$ is not covered by any hiding set element. Overall, the entire filter is 0.66-deniable. Note that if element v_1 were not in the hidden set, x_2 would be *not* deniable, as its bit $B[4]$ would be covered only by an element in the actual set, rather than by an hiding set element.

Theorem 1. *Let $\mathcal{S}$ be a set of size n, inserted in a Bloom filter $BF(\mathcal{S})$ with size m and k hash. Let $\mathcal{U}$ be the universe set, with size N_u. An* exact *expression for the filter's γ-deniability is*

$$\gamma\left(BF(\mathcal{S})\right) = \sum_{b=1}^{m} U_m(nk; b) \sum_{v=0}^{N_u-n} \binom{N_u-n}{v} \left(\frac{b}{m}\right)^{kv} \left(1-(b/m)^k\right)^{N_u-n-v} \cdot \cdot \sum_{r=0}^{b} U_b(vk; r) \left(\frac{r}{b}\right)^k \tag{4}$$

which can be approximated in closed form by

$$\gamma\left(BF(\mathcal{S})\right) \approx \left(1 - \exp\left(-\frac{vk}{m(1-e^{-kn/m})}\right)\right)^k \tag{5}$$

being $v = (N_u - n) \cdot \psi(m, k, n) = (N_u - n)(1-e^{-nk/m})^k$ the average hiding set cardinality.

Proof. See Appendix A.

3.2 γ-K-anonymity

The previous γ-deniability is a special case (K=2) of the following probabilistic K-anonymity notion adapted to the Bloom filter setting.

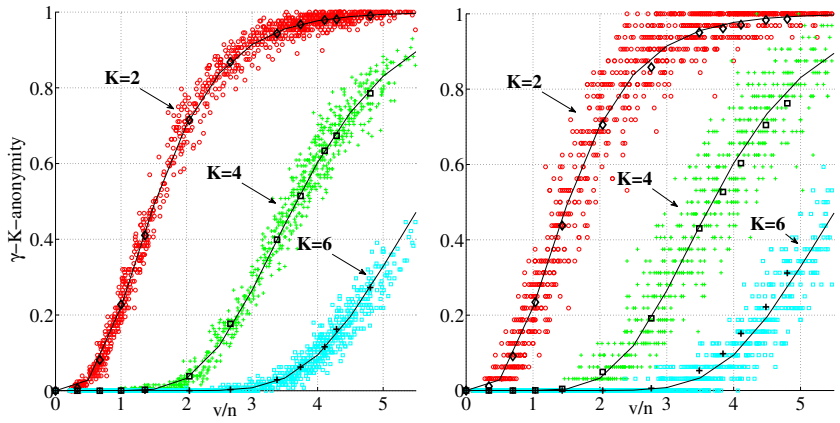

Fig. 3. γ-K-anonymity, for K=2,4,6, and two filter sizes (left: $m = 1024$, $n = 128$, $k = 5$; right: $m = 256$, $n = 32$, $k = 5$; both: $\psi = 0.0217$). Scatter plots: 100 simulations per each of 12 universe sizes; markers: simulation averaged per each universe size; lines: approximation (6).

Definition 3. *An element $x \in \mathcal{S}$ inserted in a Bloom filter $BF(\mathcal{S})$ is* K-anonymous *if $\forall i \in \{1..k\}$, there exist at least $K-1$ hiding set elements $\langle v_1 ... v_{K-1} \rangle \in \mathcal{V}$, such that $\exists \langle j_1 ... j_{K-1} \rangle \in \{1..k\}$ s.t. $H_i(x) = H_{j_1}(v_1) = ... = H_{j_{K-1}}(v_{K-1})$. A Bloom filter configuration $BF(\mathcal{S})$ is γ-K-anonymous (or, alternatively, we refer to such property as γ-K-anonymity), whenever a randomly chosen element $x \in \mathcal{S}$ is K-anonymous with probability γ.*

With reference to the example of Figure 2, element x_1 is 3-anonymous because each of its bit are covered by two or more bits of the hiding set ($B[1] \leftarrow (v_1, v_3)$, $B[3] \leftarrow (v_1, v_2)$, $B[8] \leftarrow (v_2, v_3)$) while x_2 is only 2-anonymous because its bit $B[4]$ is covered only by v_1. Overall, the entire filter can be considered as 0.33-3-anonymous.

We sketch in Appendix B how to derive an exact, albeit cumbersome and unpractical, γ-K-anonymity formula (not reported for reasons of space), as well as the following much more convenient approximation (being $v = (N_u - n) \cdot \psi(m, k, n)$):

$$\gamma\left(K, BF(\mathcal{S})\right) \approx \left(1 - \exp\left(-\frac{vk}{m(1-e^{-kn/m})}\right) \sum_{i=0}^{K-2} \frac{\left(vk/[m(1-e^{-kn/m})]\right)^i}{i!}\right)^k \tag{6}$$

Comparison between analytical approximations and simulation results are shown in Figure 3. In all cases, the γ-K-anonymity approximation (6) is very accurate, and accuracy improves as the filter size grows. Note that the scatter plot shows that dispersion with respect to average values, obviously found with different realizations of a same universe set size, do reduce with larger (more realistic) filter sizes.

4 Privacy/Utility trade-offs

In most practical applications, Bloom filters are *optimized* so as to minimize false positives for a same memory usage. We recall from well known Bloom filter results [6, 8]

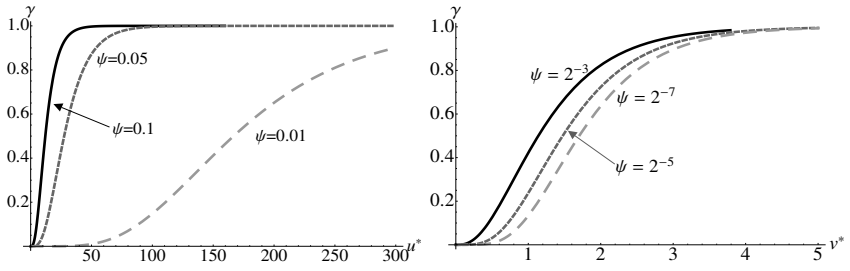

Fig. 4. (a) Left: γ-deniability versus relative universe size u^*, for $\psi = 10\%, 5\%, 1\%$; (b) Right: γ-deniability versus relative hiding set size v^*, for $\psi = 2^{-3}, 2^{-5}, 2^{-7}$

that the minimum false positive probability is attained when *half* of the bits are set to 1. In this case, $\psi = 2^{-k}$ (or, conversely, $k = -\log_2 \psi$), and, for a given stored set size n, the filter size m and the number of hash functions k are related by $kn = m \ln 2$.

It is very enlightening to explicitly devise a special expression for the γ-deniability of an *optimized* Bloom filter (a similar derivation can be carried out for the more general γ-K-anonymity case: details are omitted for reasons of space). Noting that, in such case, $(1-e^{-kn/m}) = 1/2$, and exploiting the relations $kn = m \ln 2$ and $k = -\log_2 \psi$, the γ-deniability approximation (5) can be expressed as function of (just) the false probability target ψ and the *relative hiding set* size $v^* = v/n$ as follows:

$$\gamma\left(BF(\mathcal{S})\right) = \left(1 - e^{-2vk/m}\right)^k = \left(1 - e^{-2v/n \ln 2}\right)^k = \left(1 - 4^{-v^*}\right)^{-\log_2 \psi} \quad (7)$$

Similarly, by defining with $u^* = (N_u - n)/n$ the *remaining relative Universe size*, and by recalling that $v = (N_u - n)\psi$, we obtain the following direct relation between γ-deniability and universe size (and false probability):

$$\gamma\left(BF(\mathcal{S})\right) = \left(1 - 4^{-\frac{N_u - n}{n}\psi}\right)^{-\log_2 \psi} = \left(1 - 4^{-u^*\psi}\right)^{-\log_2 \psi} \quad (8)$$

Formulae (8) and (7) are shown in Figures 4a and 4b, respectively, for different false positive targets. As expected, Figure 4a shows that a same target γ-deniability target requires large universe sets as the false positive target gets tighter. Figure 4b yields more interesting insights. On one side, it confirms that the Bloom filter's privacy properties are mainly characterized by the hiding set size, namely the *product* between (remaining) Universe set and false positive probability, and that the privacy properties of a Bloom filter are fairly inefficient: an hiding set twice the size of the original set accomplished a 0.8-deniability or less. On the other side, this plot shows that, by itself, the false positive *alone* (and, consequently, the number of hash functions employed) has a fairly limited impact when *not* associated to an increase in the hiding set size.

5 Improving privacy with targeted bit filling

These last considerations yield the suspect (confirmed by numerical results for various m, k pairs and same ψ, not reported for space reasons) that, given a *same* false posi-

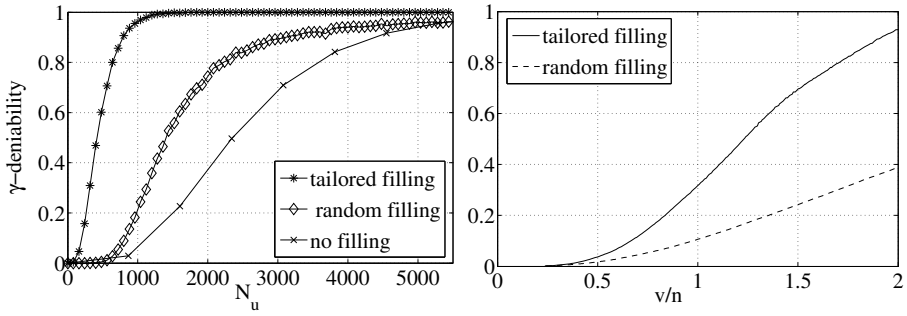

Fig. 5. (a) Left: Fraction of covered elements vs universe size N_u for a 1024 bit filter: no filling vs tailored filling vs random filling (same number of supplementary inserted bits); (b) Right: γ-deniability on the v/n ratio in the case of a fixed universe of 1408 elements.

tive target ψ, there is little margin for improving the filter's privacy using *suboptimal* parameter settings, indeed paid with a larger filter size. Thus, the only way to *sensibly* increase the filter's privacy properties is to increase the hiding set cardinality.

Frequently, Bloom filter parameters are a priori given. In these conditions, a straightforward action is to reduce statistical disclosure by adding a few random bits. At the price of increasing the false positive probability, these extra bits increase the number of elements which appear included in the filter, and hence better protect the actual content.

The discussion carried out in this paper suggests that a better approach consists in adding *tailored* bits instead of random ones. The idea is to set bits which ultimately include false positives which specifically cover filter elements otherwise deniable. To get some preliminary insights on the possible quantitative advantages of such *tailored filling*, we have implemented a simple (and preliminary) greedy heuristic based on the minimum weighted set covering problem, whose details are presented in Appendix C.

Figure 5a compares the γ-deniability achieved by such *tailored filling* strategy versus the γ-deniability obtained by setting a same random number of bits across the filter (*random filling* strategy). Despite the simplicity of this preliminary approach, results appear already very promising: a same γ-deniability is obtained with *tailored filling* with a significantly reduced universe size. Figure 5b compares the risk-utility trade-off for the two cases of random and tailored filling, fixing the universe size to 1408 elements and varying the number of supplementary inserted bits that in turn results in a variation of the v/n ratio. In this use case we have 128 elements inserted in a Bloom filter of 1024 bits.

6 Conclusions

In this paper we have defined and quantified privacy metrics tailored to Bloom filters. Using such relations, we have investigated the dependency of privacy from the absolute number of false positives (hiding set) as well as the relative fraction (false positive probability). Finally, we have preliminary investigated the advantages that a *tailored* insertion of extra covert bits yields over a random insertion strategy.

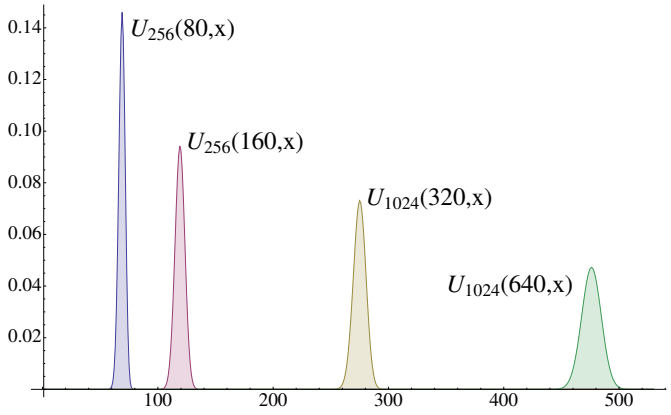

Fig. 6. Plots for $U_{256}(80, x)$, $U_{256}(160, x)$, $U_{1024}(320, x)$, $U_{1024}(640, x)$

Appendix A - Proof of Theorem 1 (sketch)

Let us start by defining the following quantities (random variables):

- $b \in (1, m)$: number of Bloom filter bits set to 1 after the insertion of the n elements of $\mathcal{S}$, each inserted through k hash draws. Using the balls and bins result (2) of Lemma 1, b has probability distribution $U_m(nk; b)$.
- $v \in (0, N_u - n)$: number of false positive elements in the hiding set $\mathcal{V}$. The probability distribution of v, *conditioned to the knowledge of* b, is a binomial distribution with probability parameter $(b/m)^k$. Note that we *do not need to approximate the false positive probability*, as we condition on b, for which we have an exact distribution.
- $r \in (0, b)$: number of bits which would still be 1 if we were removing all the "true" filter elements $\mathcal{S}$, i.e. number of distinct bits "hit" by the false positive elements $\mathcal{V}$ whose vk hash functions are, by definition of false positive, drawn among the b filter's bits set to 1. Conditioning to b and v, the distribution of r is given by Lemma 1: $U_b(vk; r)$.

A randomly chosen set element is *deniable* if all its k hash functions fall over the r (out of b) bits further covered by the hiding set. This occurs with probability $(r/b)^k$. The exact expression (4) is now a straightforward application of the law of total probability.

Computation of the exact formula (4) is cumbersome and time consuming. However, for practical filter parameters, all the above involved distributions are tightly concentrated around their mean (this is well known for Binomial distributions; see e.g. figure 6 for what concerns the *balls and bins* distribution $U_u(z; x)$ from Lemma 1). So the closed form approximation (5) is readily obtained by replacing, in $(r/b)^k$,

- $r \leftarrow b\left(1 - (1 - 1/b)^{vk}\right) \approx b\left(1 - e^{-vk/b}\right)$;
- $b \leftarrow m\left(1 - (1 - 1/m)^{nk}\right) \approx m\left(1 - e^{-nk/m}\right)$;
- $v \leftarrow (N_u - n)\psi$, using for ψ the false positive approximation $\psi = (1 - e^{-nk/m})^k$.

Appendix B - Derivation of γ-K-anonymity formulae (sketch)

Let us define the quantities b and v as in Appendix A. Let $r_K \in (1, b)$ be the number of filter bits which are set to 1 (i.e. they are "hit" by at least one of the nk hash functions applied to "true" filter elements in $\mathcal{S}$) *and* which are further "hit" by at least $K-1$ hash functions among the vk applied to elements of the hiding set $\mathcal{V}$. Note that $r_2 = r$, with r defined as per Appendix A, and that $b \geq r_2 \geq r_3 \geq r_4 \cdots$.

A random element drawn from $\mathcal{S}$ is γ-K-anonymous if all its k hash functions fall in the subset of r_{K-1} bits defined above. Assuming r_{K-1} and b known, this would occur with probability $(r_{K-1}/b)^k$. An exact derivation of a γ-K-anonymity expression can now proceed similar to Appendix A. It suffices to repeatedly apply Lemma 1 and note that (as shown in Appendix A) $r_2 \in (1, b)$ is a r.v. with distribution $U_b(vk; r_2)$, $r_3 \in (1, r_2)$ is a r.v. with distribution $U_{r_2}(vk - r_2; r_3)$, and so on; final application of the law of total probability would yield the result (but a cumbersome and far from practical one!).

Hence, it is much preferable to directly derive an approximation by using *mean values* in the γ-K-anonymity expression $(r_{K-1}/b)^k$. The mean value for the r.v. r_{K-1} can be easily derived as follows. Let us focus on a single bit among the b covered by element from $\mathcal{S}$. Define the random variable $X_i \in \{0, 1\}$ which assumes value 1 when the considered bin is "hit" by $K-1$ or more hash functions among the vk applied to elements of the hiding set $\mathcal{V}$. A Poisson approximation readily yields

$$P\{X_i = 1\} = 1 - \sum_{j=0}^{K-2} e^{-vk/b} \frac{(vk/b)^j}{j!}$$

Hence,

$$bE[X_i] = b \left(1 - e^{-vk/b} \sum_{j=0}^{K-2} \frac{(vk/b)^j}{j!} \right)$$

Approximation (6) is now readily obtained by substituting, in $(r_{K-1}/b)^k$, the above for r_{K-1}, and (as per Appendix A), $b \leftarrow m\left(1 - e^{-nk/m}\right)$.

Appendix C - tailored filling heuristic

For simplicity, we here describe an approach to improve γ-deniability only. Let us define the following three sets:

- $\mathcal{E}$: filter bits initially *not* covered by any false positive;
- $\mathcal{D}$: filter bits initially set to zero;
- $\mathcal{R} = \mathcal{U} \setminus \mathcal{S} \cup \mathcal{V}$: subset of universe elements initially neither in the filter nor false positives.

We want to find a collection of elements $\mathcal{C} \subseteq \mathcal{R}$ that cover *all* the bits in $\mathcal{E}$ while minimizing the number of filling bits added to $\mathcal{D}$. This problem resembles the minimum set cover problem, with the difference that we want to choose a coverage of $\mathcal{E}$ from a

Algorithm 1 tailored-filling($\mathcal{E}, \mathcal{D}, \mathcal{R}$)

$U \leftarrow \mathcal{E}$
$C \leftarrow 0$
$D \leftarrow \mathcal{D}$
while $U \neq 0$ **do**
  select $c \in \mathcal{R}$ that maximizes the ratio: $c \bigcap U/(c \bigcap D)$
  $U \leftarrow U \diagdown (c \bigcap U)$
  $C \leftarrow C \cup c$
  $D \leftarrow D \diagdown (c \bigcap D)$
end while
return C

subset of $\mathcal{R}$ that does not need to be of minimal cardinality, but rather needs to minimize the number of bits set in $\mathcal{D}$. We preliminary address this problem adapting a very popular *minimum weighted set covering* greedy heuristic, shown in Algorithm 1.

In the worst case, when $\nexists r_i, r_j \in \mathcal{R}$ s.t. $r_j \bigcap r_i \bigcap \mathcal{D} \neq \emptyset$ (i.e. all the subset of $\mathcal{R}$ do not share any element of $\mathcal{D}$), the collection C returned by the algorithm covers at most a number of bits in $\mathcal{D}$ that are $\log(|\mathcal{E}|)$ times the one covered by the optimal choice. This directly inherits from the properties of the greedy algorithm for the minimum weight set cover, since we use as weights the number of additional bit we need to set in the Bloom filter.

Acknowledgment: This work has been partially supported by the Italian PRIN 2009 PeopleNet Project

References

1. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7) (1970) 422–426
2. Stonebraker, M., Keller, K.: Embedding expert knowledge and hypothetical data bases into a data base system. In: Proc. of the 1980 ACM SIGMOD Int. Conf. on Management of data. (1980) 58–66
3. Maryanski, F.J.: An architecture for fault tolerance in database systems. In: Proceedings of the ACM 1980 annual conference. ACM '80 (1980) 389–398
4. Gremillion, L.L.: Designing a bloom filter for differential file access. Commun. ACM **25**(9) (September 1982) 600–604
5. Mullin, J.K.: Accessing textual documents using compressed indexes of arrays of small bloom filters. Comput. J. **30**(4) (1987) 343–348
6. Broder, A., Mitzenmacher, M.: Network applications of bloom filters: A survey. In: Internet Mathematics. (2002) 636–646
7. Cai, H., Ge, P., Wang, J.: Applications of bloom filters in peer-to-peer systems: Issues and questions. In: Proceedings of the 2008 Int. Conf. on Networking, Architecture, and Storage. NAS '08 (2008) 97–103
8. Tarkoma, S., Rothenberg, C., Lagerspetz, E.: Theory and practice of bloom filters for distributed systems. Communications Surveys Tutorials, IEEE **14**(1) (quarter 2012) 131 –155
9. Stranneheim, H., Kaller, M., Allander, T., Andersson, B., Arvestad, L., Lundeberg, J.: Classification of dna sequences using bloom filters. Bioinformatics **26**(13) (2010) 1595–1600
10. Bellovin, S.M., Cheswick, W.R.: Privacy-enhanced searches using encrypted bloom filters. IACR Cryptology ePrint Archive **2004** (2004) 22

11. Raykova, M., Vo, B., Bellovin, S.M., Malkin, T.: Secure anonymous database search. In: Proc. of the 2009 ACM workshop on Cloud computing security. CCSW '09 (2009) 115–126
12. Goh, E.J.: Secure indexes. Cryptology ePrint Archive, Report 2003/216 (2003) http://eprint.iacr.org/2003/216/.
13. Nojima, R., Kadobayashi, Y.: Cryptographically secure bloom-filters. Trans. Data Privacy **2**(2) (2009) 131–139
14. Boneh, D., Kushilevitz, E., Ostrovsky, R., Skeith, III., W.E.: Public key encryption that allows pir queries. In: Proceedings of the 27th annual international cryptology conference on Advances in cryptology. CRYPTO'07, Berlin, Heidelberg, Springer-Verlag (2007) 50–67
15. Rottenstreich, O., Keslassy, I.: The bloom paradox: When not to use a bloom filter? In: Proc. 31th IEEE Int. Conf. on Computer Communications. INFOCOM, Orlando, Fl, USA (2012)
16. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **10**(5) (2002) 557–570
17. Lodha, S., Thomas, D.: Probabilistic anonymity. In: Proc. of the 1st ACM SIGKDD int. conf. on Privacy, security, and trust in KDD. PinKDD'07 (2008) 56–79
18. Gross, P., Parekh, J., Kaiser, G.: Secure selecticast for collaborative intrusion detection systems. In: 3rd International Workshop on Distributed Event-Based Systems (DEBS'04). (2004)
19. Shanmugasundaram, K., Brönnimann, H., Memon, N.: Payload attribution via hierarchical bloom filters. In: Proceedings of the 11th ACM conference on Computer and communications security. CCS '04, New York, NY, USA, ACM (2004) 31–41
20. Gorai, M., Sridharan, K., Aditya, T., Mukkamala, R., Nukavarapu, S.: Employing bloom filters for privacy preserving distributed collaborative knn classification. In: Information and Communication Technologies (WICT), 2011 World Congress on. (dec. 2011) 495 –500
21. Siegenthaler, M., Birman, K.: Sharing private information across distributed databases. Network Computing and Applications, IEEE International Symposium on (2009) 82–89
22. Parekh, J.J., Wang, K., Stolfo, S.J.: Privacy-preserving payload-based correlation for accurate malicious traffic detection. In: Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense. LSAD '06 (2006) 99–106
23. Bawa, M., Bayardo, Jr, R.J., Agrawal, R., Vaidya, J.: Privacy-preserving indexing of documents on the network. The VLDB Journal **18**(4) (2009) 837–856
24. Lai, P.K.Y., Yiu, S.M., Chow, K.P., Chong, C.F., Hui, L.C.K.: An efficient bloom filter based solution for multiparty private matching. In: Proc. of the 2006 Int. Conf. on Security & Management, SAM 2006, Las Vegas, Nevada, USA, June 26-29, 2006. (2006) 286–292
25. Kuzu, M., Kantarcioglu, M., Durham, E., Malin, B.: A constraint satisfaction cryptanalysis of bloom filters in private record linkage. In: Proceedings of the 11th international conference on Privacy enhancing technologies. PETS'11 (2011) 226–245
26. Schnell, R., Bachteler, T., Reiher, J.: Private record linkage with bloom filters. In: Proc. of Statistics Canada Symposium 2010: Social Statistics: The Interplay among Censuses, Surveys and Administrative Data. (2010) 304–309
27. Goodrich, M.T., Mitzenmacher, M.: Invertible bloom lookup tables. CoRR **abs/1101.2245** (2011)
28. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: L-diversity: Privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data **1**(1) (2007)
29. Li, N., Li, T.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: In Proc. of IEEE 23rd Intl Conf. on Data Engineering (ICDE07). (2007)
30. Dwork, C.: Differential privacy: a survey of results. In: Proceedings of the 5th international conference on Theory and applications of models of computation. TAMC'08 (2008) 1–19
31. Bose, P., Guo, H., Kranakis, E., Maheshwari, A., Morin, P., Morrison, J., Smid, M., Tang, Y.: On the false-positive rate of bloom filters. Inf. Process. Lett. **108**(4) (2008) 210–213