Probabilistic Properties of the Spatial Bloom Filters and Their Relevance to Cryptographic Protocols

Luca Calderoni, Paolo Palmieri, and Dario Maio, *Member, IEEE*

***Abstract*—The classical Bloom filter data structure is a crucial component of hundreds of cryptographic protocols. It has been used in privacy preservation and secure computation settings, often in conjunction with the (somewhat) homomorphic properties of ciphers such as Paillier's. In 2014, a new data structure extending and surpassing the capabilities of the classical Bloom filter has been proposed. The new primitive, called spatial Bloom filter (SBF) retains the hash-based membership-query design of the Bloom filter, but applies it to elements from multiple sets. Since its introduction, the SBF has been used in the design of cryptographic protocols for a number of domains, including location privacy and network security. However, due to the complex nature of this probabilistic data structure, its properties had not been fully understood. In this paper we address this gap in knowledge and we fully explore the probabilistic properties of the SBF. In doing so, we define a number of metrics (such as emersion and safeness) useful in determining the parameters needed to achieve certain characteristics in a filter, including the false positive probability and inter-set error rate. This will in turn enable the design of more efficient cryptographic protocols based on the SBF, opening the way to their practical application in a number of security and privacy settings.**

***Index Terms*—Spatial Bloom filters, cryptographic protocols**

I. Introduction

BLOOM filters are a probabilistic data structure allowing membership queries on the elements of a set, proposed by Burton Howard Bloom in 1970 [1]. A Bloom filter represents a set in a space-efficient way through the use of hash functions. The filter built on a set can be queried to test whether an element is a member of that set: false positive matches are possible (where an element outside the set is recognized as being a member of it), but false negatives are not. While introducing false positives, Bloom filters have a strong space advantage over other data structures for representing sets. Moreover, the construction parameters of the filter allow the false positive probability to be tuned according to the user's needs, making Bloom filters suitable for most application scenarios. In fact, Bloom's data structure has proved to be extremely popular in a number of domains, and in particular in networking and security settings [2]. Another crucial characteristic of Bloom filters is that the data structure can be naturally represented as a set of binary values. This, in turn, allows for further compressibility, but more crucially, it also makes a number of encryption operations possible. In particular, the use of (somewhat) homomorphic encryption schemes such as Paillier [3] allows membership queries to be performed over an encrypted filter. This property has been used in a number of set-based cryptographic protocols, including privacy preserving search [4], private record linage and intersection [5], and private similarity testing [6] among others.

Despite its flexibility and applicability, the main limitation of Bloom filters is their ability to represent only a single set: whenever computation has to be performed over multiple sets, each set requires a separate filter. This limitation was recently overcome by the introduction of the spatial Bloom filter (SBF), a data structure extending the classical Bloom filter to enable representation of an unlimited number of sets [7], [8]. The spatial Bloom filter was initially proposed as part of a location privacy protocol, and used to represent multiple geographical areas to be compared privately with the location of a user. This is the main reason behind the term *spatial*. However, it is important to point out that this data structure is not limited to the location privacy domain, but it is indeed suitable for any kind of set-based problem. To this end, it could be conveniently referred to as 'Multi-set Bloom Filter' or 'Overlapping Bloom Filter', although it is preferable to maintain its name unaltered for reason of continuity with respect to the literature and for ease of reference by the scientific community. The SBF original protocol, based on Paillier's cryptosystem, compares positively with existing location-privacy solutions, as evidenced by the comparative study by Solomon et al [9]: in particular, the spatial Bloom filter allows for a significant reduction in memory usage and communication overhead. The space and communication efficiency of the SBF opened the way for its application in other domains, including anonymous routing protocols and the Internet of Things [10]. However, despite the potential for the spatial Bloom filter to be used in a number of scenarios, its probabilistic properties have not been fully studied to date. In particular, the storage of multiple sets introduces elements of complexity in the determination of the false positive probability. While the original paper [8] correctly estimated the probability for an element external to the sets represented in the filter to be recognized as a member (the typical false positive case), it did not fully explore the case of inter-set errors, where elements belonging to a set are recognized as member of a different one, where both sets are

Manuscript received July xx, 2017; revised January xx, 2018; accepted January xx, 2018. Date of publication Month xx, 2018; date of current version Month xx, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. xxxxxx xxxxxx. (Corresponding author: Luca Calderoni.)

L. Calderoni and D. Maio are with the Department of Computer Science and Engineering, Università di Bologna, Italy (e-mail: luca.calderoni@unibo.it; dario.maio@unibo.it).

P. Palmieri is with the Department of Computer Science, University College Cork, Ireland (e-mail: p.palmieri@cs.ucc.ie). Part of this research work was carried out while he was with Cranfield University, UK.

This paper has supplementary downloadable material at http://ieeexplore.ieee.org, provided by the authors. Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier

part of the filter.

In this paper we study in depth the probabilistic properties of the spatial Bloom filter, and we introduce a number of novel metrics such as *emersion* and *safeness*, that are useful in determining the characteristics of a filter, both before (*a priori*) and after (*a posteriori*) its construction. The new metrics allow the construction of filters precisely calibrated to respect user defined parameters, including the acceptable false positive probability and error rate. As such, the new metrics will enable the efficient use of the SBF in security and privacy cryptographic protocols, reducing space and communication overhead to the optimal minimum. As part of this work, we further facilitate SBF adoption by providing a reference implementation of the data structure (released under LGPL). The implementation provides experimental validation for the proposed metrics and the accuracy of the probabilistic model. Finally, we discuss three different security settings where SBF can be adopted, and we show how the metrics can be used to identify the filter characteristics that best fit the requirements of each case.

II. Preliminaries

Definition 1. *We define a* Bloom filter $B(S)$ *representing the originating set* $S = \{a_1, \ldots, a_n\}$ *as the set*

$$B(S) = \bigcup_{a \in S, h \in H} h(a) \ , \tag{1}$$

where $H = \{h_1, \ldots, h_k\}$ *is a set of* k *hash functions such that each* $h_i \in H : \{0,1\}^* \to \{1, \ldots, m\}$*, that is, the hash functions take binary strings as input and output a random number uniformly chosen in* $\{1, \ldots, m\}$*.*

The spatial Bloom filter (SBF) is defined as follows [8]:

Definition 2. *Given the originating sets* $\Delta_1, \Delta_2, \ldots, \Delta_s$ *to be represented in the filter, let* $\bar{S}$ *be the union set* $\bar{S} = \bigcup_{\Delta_i \in S} \Delta_i$ *and let* S *be the set of sets* $S = \{\Delta_1, \Delta_2, \ldots, \Delta_s\}$*. Let* O *be the strict total order over* S *for which* $\Delta_i < \Delta_j$ *for* $i < j$*. Let also* $H = \{h_1, \ldots, h_k\}$ *be a set of* k *hash functions such that each* $h_i \in H : \{0,1\}^* \to \{1, \ldots, m\}$*, that is, each hash function in* H *takes binary strings as input and outputs a random number uniformly chosen in* $\{1, \ldots, m\}$*. We define the* spatial Bloom filter *over* (S, O) *as the set of couples*

$$B^{\#}(S, O) = \bigcup_{i \in I} \langle i, \max L_i \rangle \tag{2}$$

where I *is the set of all values output by hash functions in* H *for elements of* $\bar{S}$

$$I = \bigcup_{\delta \in \bar{S}, h \in H} h(\delta) \tag{3}$$

and L_i *is the set of labels* l *such that:*

$$L_i = \{l \mid \exists \delta \in \Delta_l, \exists h \in H : h(\delta) = i\} \ . \tag{4}$$

A spatial Bloom filter $B^{\#}(S, O)$ can be represented as a vector $b^{\#}$ composed of m values, where the i-th value

$$b^{\#}[i] = \begin{cases} l & \text{if } \langle i, l \rangle \in B^{\#}(S, O) \\ 0 & \text{if } \langle i, l \rangle \notin B^{\#}(S, O) \end{cases} \quad \text{[8].} \tag{5}$$

TABLE I
The notation commonly used in SBF literature and the new symbols introduced within this paper.

Known symbol	Description
$\mathcal{E}$	A domain for elements to be mapped inside a SBF
$B^{\#}(S, O)$	A spatial Bloom filter
$b^{\#}$	The vector representation of the SBF
k	The number of hash runs
m	The number of cells of the SBF
n	The total number of elements to be inserted into the SBF
s	The number of originating sets
Δ_i	The i-th originating set
S	The set of originating sets $\{\Delta_1, \ldots, \Delta_s\}$, $\lvert S \rvert = s$
$\bar{S}$	The union set $\bigcup_{\Delta_i \in S} \Delta_i$, where $\lvert \bar{S} \rvert = n$
L	The set of set labels $\{1, \ldots, s\}$
New symbol	**Description**
$v^{\#}$	The verification function $v^{\#} : \mathcal{E} \to \{0, \ldots, s\}$
$b^{\#}_{\dashv i}$	The vector representation of an SBF when it reaches the state i
n_i	The number of members of the i-th set, $\lvert \Delta_i \rvert = n_i$
c_i	The number of cells in $b^{\#}$ containing the value i
$c_{i \dashv i}$	The number of cells in $b^{\#}_{\dashv i}$ containing the value i ($c_{i \dashv s} = c_i$)
μ_i	The number of self-collisions observed for the i-th set
FPP_i	The *a priori* false positive probability for the i-th set
FPP'_i	The *a posteriori* false positive probability for the i-th set
FPR_i	The false positive rate observed for the i-th set
ISEP_i	The *a priori* inter-set error probability for the i-th set
ISEP'_i	The *a posteriori* inter-set error probability for the i-th set
ISER_i	The inter-set errors rate observed for the i-th set
ISE_i	The number of inter-set errors observed for the i-th set
SAFEP	The *a priori* probability to have $\text{ISER}_i = 0$, $\forall i \in L$
SAFEP_i	The *a priori* probability for the i-th set to have $\text{ISER}_i = 0$
Ω_i	The emersion of the i-th set

In the following, when referring to a spatial Bloom filter, we refer to its vector representation $b^{\#}$. Each position of this one-dimensional array is referred to as a filter *cell*.

Table I presents the notation commonly used in the spatial Bloom filter literature. Please note that some of the new symbols introduced in this paper are included as well, for easier consultation.

A. SBF construction

A spatial Bloom filter is built as follows [8]. Initially, all values in $b^{\#}$ are set to 0. Then, for each element $\delta \in \Delta_1$ and for each hash function $h \in H$ we compute $h(\delta) = i$ and we write 1 (the label of Δ_1) at $b^{\#}[i]$. We call this process the *insertion* of element δ into the filter. We do the same for elements belonging to Δ_2 (writing the value 2), and we proceed incrementally until all s sets in S have been encoded in $b^{\#}$. We observe that the ordered construction process implies that in case of a collision (see below), sets with lower labels are more likely to be overwritten than sets with higher label values. No overwriting can happen for elements of Δ_s. We discuss how this affects the filter probabilistic properties in Section III.

It is important to clarify how collisions (overwrites) happen. In particular, we note that they occur when a single hash function produces the same output for two distinct elements (what is commonly known as a hash collision), but also when two different hash functions produce the same output for the

same element or for two different elements. In all these cases, the same filter cell is written twice. Throughout this paper we refer to this phenomenon as *cell overwrite*, or simply as *collision*, indifferently.

In the following, we provide a definition for some concepts and events related to the filter construction (including two specific kind of collisions) that had never been formalized in the SBF literature, but are useful for the study of its probabilistic properties. First, we define a *state* in the construction:

Definition 3. *Let us consider the spatial Bloom filter $b^{\#}$. Given $i \in L$, we say the filter is in* state i *(and we refer to its vector representation as $b^{\#}_{\dashv i}$) if and only if all the elements of set Δ_i have been inserted into the filter.*

Therefore, state 0 ($b^{\#}_{\dashv 0}$) represents the empty filter (with all of its cells set to zero). At the end of the construction process, the filter is in state s ($b^{\#}_{\dashv s}$).

During the filter construction, the insertion of the elements of set Δ_i into the filter causes a number of cells to be set to i. The exact number depends on the outputs of the hash functions in the insertion process: if all outputs are different, we say that no collision has occurred and $(k \cdot n_i)$ cells are written. However, if two (or more) hash outputs are the same, one (or more) collision has occurred, and the total number of cells set to i is lower. Collisions can occur during the insertion of different elements, or even for the same element over different hash functions. We call *self-collisions* the overwrite events occurring over elements of the same set, and we define them as follows:

Definition 4. *Given a filter $b^{\#}$, a* self-collision *for set Δ_i occurs each time after the first the value i is written on a same cell during the insertion of the elements of Δ_i. We call μ_i the total number of self-collisions for set Δ_i, occurring during the transition from the state $i-1$ to the state i (that is, while elements of Δ_i are being inserted).*

Similarly, overwrite events occurring during the insertion of a single element (i.e. when two or more different hash functions address the same cell) are referred to as *intra-element collisions*.

B. SBF verification

Let us consider a SBF $B^{\#}(S, O)$ built over the sets $\Delta_1, \ldots, \Delta_s$ such that $\bar{S} = \bigcup_{i=1}^{s} \Delta_i$. Each set Δ_i contains elements belonging to a generic domain $\mathcal{E}$.

The verification procedure which classifies an element $\delta \in \mathcal{E}$ as belonging to one of the originating sets (see [7], [8]), may be formalized using a functional notation as follows:

$$\begin{aligned} v^{\#}: \quad & \mathcal{E} \to L_0 \\ & \delta \mapsto v^{\#}(\delta) \end{aligned} \tag{6}$$

where $L_0 = \{0, \ldots, s\}$. This function takes as input a generic element of $\mathcal{E}$ and outputs an integer included between 0 and s. This integer indicates the set to which the element is supposed to belong to (if the output is > 0), or indicates that the element does not belong to any of the originating sets (when the output is 0).

III. Probabilistic Properties of the Spatial Bloom Filter

As both Bloom filters and Spatial Bloom Filters are probabilistic data structures by definition, it is crucial to understand the probabilistic characteristics they possess, and how these determine their ability to correctly answer set membership queries. In reality, both data structures display a behavior that can be deterministic or probabilistic depending on the element for which membership is queried. By deterministic behavior, we refer to a membership query that always returns the same result, based only on the set of elements over which the filter is built (called the *originating set*). A probabilistic behavior is instead displayed when the membership query result depends on factors other than the originating set, such as the chosen hash functions and the resulting filter construction. In the case of a classic Bloom filter, which is built on a single set, it is possible for an element outside the filter's originating set to be wrongly recognized as a member of it (producing a *false positive*). This behavior is due to the hash functions colliding (that is, producing the same output) for different elements, and therefore its occurrence depends on the chosen set of hashes. Hence, the filter is non-deterministic (and hence probabilistic) in replying to queries regarding elements that are not part of the originating set. An error in the opposite direction (*false negative*) is instead impossible: membership queries are always positively replied for members of the originating set. For those elements, the filter displays a deterministic behavior.

Bloom filters are well suited for a false positive/false negative discussion, as they provide binary classification: the verification function (which computes the result of a membership query) just determines whether or not the element provided as input belongs to the filter originating set. Spatial Bloom filters, on the contrary, are constructed over a number of (disjoint) originating sets. Thus, the verification process is to be considered a multinomial classification problem.

When the Spatial Bloom Filter was first introduced [7], [8], the authors discussed the SBF probabilistic properties concerning false positives. However, there was no focus on the difference between the wrong classification of an element outside the originating sets that is wrongly recognized as belonging to one; and the wrong classification of a member of an originating set that is recognized as belonging to a different one. The *a priori* (i.e. before construction) false positive probability that was discussed in the original papers is in fact based only on the probability for each hash function to collide, and therefore to point to a position of the filter which was previously written onto.

Similarly to the classic Bloom filter, the SBF does not allow for false negatives: a membership query over an element that is part of the originating sets will always result in a positive response. However, the element may be classified as a member of the wrong set. This phenomenon, which is specific to the SBF, deserves special attention, and a proper understanding of the probability with which these exchanges may occur is crucial to enable application of the spatial Bloom filters. In the following, we therefore study this *inter-set error* probability, something that was left unexplored in the existing

TABLE II
THE BF AND SBF IN RELATION TO THE PROPOSED ERROR TAXONOMY.

	False positives	False negatives	Inter-set errors
BF	Yes	No	No
SBF	Yes	No	Yes

SBF literature.

In the final part of this work we validate the presented probabilistic framework with extensive experimentation, by using the reference implementation of the data structure we developed as part of this research. This allows us to discuss not only the expected error probabilities, but also the effective error rates occurring over actual filters. Specifically, we will discuss false positives and inter-set errors from both the *a priori* and the *a posteriori* perspective. This concept allow us to confirm the correctness of our probabilistic model and the implementation of the data structure. Knowledge of the *a posteriori* error rates is crucial for real-world adoption of the SBF, as these indicate the suitability of a specific filter to a chosen context.

A final remark is needed with respect to the probabilistic approach we use in the following. The first probabilistic model proposed by Bloom [1] and refined by Mullin [11] was proved to be a lower bound of a more precise probabilistic model introduced by Christensen [12]. However, the relative error between the latest proposed model and the classic one is irrelevant for increasing values of m (specifically, this error is negligible when $m > 1024$). As SBFs are designed to store multiple sets, their application with very low values of m is of little relevance. Hence, our discussion on SBF's probabilistic properties relies on the well known classic model.

A. Error taxonomy

A membership query over a spatial Bloom filter can result in the following outcomes:

True positive: the element belongs to one of the originating sets, and is classified as such.

True negative: the element does not belong to any of the originating sets, and is classified as such.

False positive: the element does not belong to any of the originating sets, but is classified as belonging to one of the originating sets.

False negative: the element belongs to one of the originating sets, but is classified as not belonging to any of the sets.

Inter-set error: the element belongs to one of the originating sets, but is wrongly classified as belonging to another set.

Table II sums up the general properties of BF and SBF concerning errors.

We formalize this taxonomy in accordance with the verification function introduced in Section II-B. Let us consider the filter $b^{\#}$ and the associated verification function $v^{\#}$. Given $\Delta_i \in S$, $\delta \in \mathcal{E}$ and $i, j, k \in L$, we can assert that:

True positive: $v^{\#}(\delta) = i, \quad \delta \in \Delta_i$.
True negative: $v^{\#}(\delta) = 0, \quad \delta \notin \bar{S}$.
False positive: $v^{\#}(\delta) = k, \quad \delta \notin \bar{S}$.
False negative: $v^{\#}(\delta) = 0, \quad \delta \in \Delta_i$.
Inter-set error: $v^{\#}(\delta) = j, \quad \delta \in \Delta_i,\ i \neq j$.

Definition 5. *Let us consider a SBF $b^{\#}$ and a set $\bar{N}$ such that $\bar{N} \subset \mathcal{E}$, $\bar{S} \cap \bar{N} = \emptyset$. Assuming we test for membership each element of $\bar{N}$ against $b^{\#}$, let* FP_i *be the overall number of false positives reported for the i-th set. We define the* false positive rate *for the set Δ_i over the verification set $\bar{N}$ as:*

$$\text{FPR}_i = \frac{\text{FP}_i}{|\bar{N}|} \ . \tag{7}$$

Definition 6. *Let us consider a SBF $b^{\#}$. Assuming we test for membership each element of $\bar{S}$ against $b^{\#}$, let* ISE_i *be the overall number of inter-set errors reported for the i-th set (i.e. the number of occurrences for the event $v^{\#}(\delta) = j,\ \delta \in \Delta_i,\ i \neq j$). We define the* inter-set error rate *for the set Δ_i over the construction set $\bar{S}$ as:*

$$\text{ISER}_i = \frac{\text{ISE}_i}{|\Delta_i|} \ . \tag{8}$$

B. False positives in Bloom filters

Before proceeding to discuss the SBF's probabilistic properties, it is useful to recall, for comparison, some key points concerning false positives in the classic Bloom filter [1]. Let us consider a Bloom filter composed of m bits (here referred to as $cells$ for ease of comparison with SBFs) and built using k hash functions, each returning as output a digest uniformly chosen within the range $[1, m]$. Finally, let us consider n members of a generic set to be inserted into the filter. During the insertion procedure, the probability for a specific cell to be hit by a hash digest is $1/m$. Hence, the probability for that cell not to be written is $(1 - 1/m)$. Since to complete the insertion procedure for a single element we need to compute k digests, the probability for a specific cell not to be hit (and thus written into) at the end of the whole construction process of the filter (i.e. when all n elements have been inserted) is:

$$\left(1 - \frac{1}{m}\right)^{kn} \ . \tag{9}$$

Let us now consider an element which does not belong to the originating set and let us check for its membership against the filter. The probability for a non-empty cell to be hit by a hash digest is:

$$1 - \left(1 - \frac{1}{m}\right)^{kn} \ . \tag{10}$$

However, in order for the element to be supposed to belong to the originating set, all of the k digests have to address non-empty cells. The probability for such an event to occur is:

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \ . \tag{11}$$

This equation thus represents the *a priori false positive probability* for a Bloom filter.

It is important to point out that, after the construction procedure is completed, the exact number of non-empty cells is known. We can then infer a more precise probability concerning false positive events. Specifically, let us suppose the filter has c cells set to 1 (i.e. non-empty). When we check

for the membership of an element, the probability for all of the k hashes to output digests addressing non-empty cells is:

$$\left(\frac{c}{m}\right)^k \tag{12}$$

which we may refer to as the *a posteriori false positive probability* for a specific Bloom filter.

C. False positives in spatial Bloom filters

Spatial Bloom filters feature several originating sets. Given $\left|\bar{S}\right| = n$, the probability to experience a false positive when we classify an element against a SBF equals the one which can be derived for a classic BF built over $\bar{S}$, as long as the values m and k are the same for the two filters.

However, as shown in [7], [8], the false positive probability may be split into several set-specific probabilities. As each set Δ_i contains $n_i = |\Delta_i|$ members, it follows that:

$$\text{FPP}_s = \left(1 - \left(1 - \frac{1}{m}\right)^{kn_s}\right)^k \tag{13}$$

$$\text{FPP}_{s-1} = \left(1 - \left(1 - \frac{1}{m}\right)^{k(n_s+n_{s-1})}\right)^k - \text{FPP}_s \tag{14}$$

$$\cdots$$

$$\text{FPP}_1 = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k - \text{FPP}_s - \cdots - \text{FPP}_2 \ . \tag{15}$$

In general, the *a priori false positive probability* for the i-th set is:

$$\text{FPP}_i = \left(1 - \left(1 - \frac{1}{m}\right)^{k\sum_{j=i}^{s} n_j}\right)^k - \sum_{j=i+1}^{s} \text{FPP}_j \ . \tag{16}$$

Similarly to the Bloom filter, when the SBF construction is completed, we may count, for each set, the number of cells containing the set label. These values allow us to perform a more precise probabilistic estimate of the set-specific false positives for the given filter. Specifically, let c be the total number of non-zero cells and let c_i be the number of cells which hold the value i. The *a posteriori false positive probability* for each set can be formalized as follows:

$$\text{FPP}'_s = \left(\frac{c_s}{m}\right)^k \tag{17}$$

$$\text{FPP}'_{s-1} = \left(\frac{c_s + c_{s-1}}{m}\right)^k - \text{FPP}'_s \tag{18}$$

$$\cdots$$

$$\text{FPP}'_1 = \left(\frac{c}{m}\right)^k - \text{FPP}'_s - \cdots - \text{FPP}'_2 \ . \tag{19}$$

In general, for the i-th set:

$$\text{FPP}'_i = \left(\frac{\sum_{j=i}^{s} c_j}{m}\right)^k - \sum_{j=i+1}^{s} \text{FPP}'_j \ . \tag{20}$$

D. False negatives

Both Bloom filters and spatial Bloom filters do not admit false negatives. This property can be intuitively derived from their construction process.

Proposition 1. *Let us consider the filter $b^{\#}$ and the associated verification function $v^{\#}$. Given $i \in L$, $\forall \delta \in \Delta_i$, $v^{\#}(\delta) \neq 0$.*

Proof. Since $\delta \in \Delta_i$, this element was processed by the k hash functions during the construction process of $b^{\#}$. Depending on collisions, this process resulted in a number of cells ≥ 1 and $\leq k$ set to i. As the set of hashes to be used during the construction process coincides with the one to be used during the verification process, checking for the element δ will head to the same cells which were written during the construction phase. Hence, the only way to realize $v^{\#}(\delta) = 0$ would be if one of the elements inserted into the filter after δ would produce an overwrite of one of the aforementioned cells with the value 0. However, per the SBF definition, elements belonging to sets $\Delta_i, \ldots, \Delta_s$ may only produce overwrites of value $j \in \{i, \ldots, s\}$, from which the assertion follows. □

E. Inter-set errors

One of the main features of SBFs consists in the overwriting rule used for collision handling during the construction process. Following this principle, it is possible for an originating set to have the cells representing it (that is, the cells containing the set label) overwritten with greater label values. This event occurs each time an element belonging to a set Δ_j given as input to a hash function results in an output digest pointing to a cell containing a lower value $i < j$.

This is why, during the verification phase, it is possible for an element belonging to the set Δ_i to be wrongly classified as belonging to a different set Δ_j, only if $j > i$. In order for this exchange to occur, the element needs to be particularly "unlucky": specifically, each one of the cells representing it in the filter should be overwritten.

The expected frequency of these events, here referred to as *inter-set errors* (ISE), may be formalized as an *a priori* probability before the filter construction takes place. After the filter has been constructed, instead, we are able to compute the exact rate of inter-set errors by means of a self-check, i.e. testing all of the members of $\bar{S}$ against the filter and collecting the inter-set errors for each set Δ_i on occurrence (see Definition 6 for reference).

In the following, we formalize the overwriting process from a probabilistic point of view and we discuss the *a priori inter-set error probability* (ISEP) and the *a posteriori inter-set error probability* (ISEP′) for each set of a SBF.

1) Cells overwriting: Let us suppose we want to construct a SBF starting from s originating sets $\Delta_1, \ldots, \Delta_s$. We want to predict, for each set, how many cells will represent it when the filter reaches its final state s. In other words, we want to compute, for each set Δ_i, an estimate of the number $c_{i\dashv s}$ of cells set to i in $b^{\#}_{\dashv s}$ (by convention, $c_{i\dashv s} = c_i$).

For ease of understandability, let us recall that the filter is built following the strict total order O: starting from the state 0, each set is processed in ascending order until the filter reaches

the state s. Specifically, when the filter reaches the i-th state, $b^{\#}_{\dashv i}$ will contain the maximum number of cells set to i. As each element is processed by k hash functions, members of Δ_i may produce at most kn_i distinct cells set to i, depending on the number of *self-collisions* (see Definition 4 for reference).

During the insertion phase of the elements of Δ_i, each cell has a probability of $1/m$ to be hit (and thus set to i) and, consequently, a probability of $(1 - 1/m)$ not to be hit. Hence, the probability for a cell not to be hit when the filter reaches the i-th state is:

$$\left(1 - \frac{1}{m}\right)^{kn_i} \quad . \tag{21}$$

On the contrary, the probability to contain the value i is

$$1 - \left(1 - \frac{1}{m}\right)^{kn_i} \quad . \tag{22}$$

Therefore, since the filter is composed of m cells, the expected value of cells set to i in $b^{\#}_{\dashv i}$ is:

$$E[c_{i\dashv i}] = m\left(1 - \left(1 - \frac{1}{m}\right)^{kn_i}\right) \quad . \tag{23}$$

After the filter reaches the state i, the construction phase goes forth until all the members of the remaining sets $\Delta_{i+1}, \ldots, \Delta_s$ are processed. These elements are the only ones which may produce overwrites upon the cells set to i. We call these overwrite events collisions. Let us define the total number of such elements as

$$n_i^{\text{FILL}} = \sum_{j=i+1}^{s} n_j \quad . \tag{24}$$

Hence, following the same principle discussed above, the probability for a cell in $b^{\#}_{\dashv s}$ not to be set to a value $j > i$ is:

$$\left(1 - \frac{1}{m}\right)^{kn_i^{\text{FILL}}} \quad . \tag{25}$$

We conclude that the expected number of cells set to i when the construction process is completed (i.e. in $b^{\#}_{\dashv s}$) is:

$$E[c_{i\dashv s}] = m\left(1 - \left(1 - \frac{1}{m}\right)^{kn_i}\right)\left(1 - \frac{1}{m}\right)^{kn_i^{\text{FILL}}} \quad . \tag{26}$$

2) Error probability: Having analyzed the issue of overwrites (collisions), we can proceed to discuss the *inter-set error probability*. It is important to clarify that, in order for an element $\delta \in \Delta_i$ to be wrongly classified as a member of a different set, all of its corresponding cells (which originally contained the value i) need to be overwritten by a value j, where $i < j \leq s$.

Hence, the probability for such an event to occur may be discussed as the probability for a member of the set Δ_i to "disappear" from the filter, that is, to have all of its representative cells overwritten when the filter reaches state s. An example of such an event is provided in Figure 1.

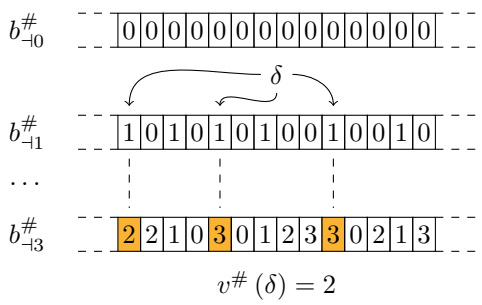


Fig. 1. A simple example of an inter-set error for a generic element δ (here we assume $k = 3$ and $s = 3$). At the end of the construction process, the three cells representing the element were overwritten. Thus, the verification function outputs a wrong label when it is queried for δ's membership.

For simplicity, let us suppose that each member of Δ_i is represented by exactly k cells in $b^{\#}_{\dashv i}$, that is, the k hash functions used to map each element have not been affected by *intra-element collisions*. This assumption is reasonable when $k \ll m$, as in this case it is relatively rare to experience a collision among k hash digests over m cells. The probability for a cell not to be set to a value j ($i < j \leq s$) when the filter reaches the state s is proposed in (25). Hence, the probability for a cell in $b^{\#}_{\dashv s}$ to contain such a value is

$$1 - \left(1 - \frac{1}{m}\right)^{kn_i^{\text{FILL}}} \quad . \tag{27}$$

Therefore, the probability for an element of Δ_i to have all of its k corresponding cells overwritten in $b^{\#}_{\dashv s}$ is

$$\text{ISEP}_i = \left(1 - \left(1 - \frac{1}{m}\right)^{kn_i^{\text{FILL}}}\right)^k \quad . \tag{28}$$

It follows that the expected value of *inter-set errors* for the set Δ_i is

$$E[\text{ISE}_i] = n_i\left(1 - \left(1 - \frac{1}{m}\right)^{kn_i^{\text{FILL}}}\right)^k \quad . \tag{29}$$

Although at the end of the construction process we may compute an exact *inter-set error rate* via a self-check upon the elements of each set, an *a posteriori inter-set error probability* may be derived as well. Specifically, when the filter reaches the state i, there are $c_{i\dashv i} = kn_i - \mu_i$ cells set to i inside the filter. Hence, when a member of Δ_i is checked for membership, each hash function will hit one among these cells. When the filter reaches the state s some of the aforementioned cells may be overwritten. In order for an element of Δ_i to be misrecognized, we must never hit a cell set to i within k trials, i.e. we must hit overwritten cells which were previously set to i for each of the k hash outputs. As when the filter reaches the state s it contains $c_i \leq kn_i - \mu_i$ cells set to i, the probability for this event to occur is:

$$\text{ISEP}'_i = \left(\frac{(kn_i - \mu_i) - c_i}{kn_i - \mu_i}\right)^k \quad . \tag{30}$$

The denominator represents the overall number of cells a hash may point to when it is applied to an element of Δ_i (as these are the cells which were written at the state i); while the numerator expresses the number of cells among the aforementioned ones which were overwritten (as c_i is the number of cells set to i at the end of the construction process).

IV. Emersion

We define, for each originating set of a SBF, a value quantifying the degree of representation for that set: the resulting metric is a synthetic indicator expressing how much of the set "emerges" when the filter reaches its final state s. As pointed out in Section III, each set Δ_i reaches its maximum degree of representation when the filter reaches the state i. At this stage, the number of cells representing Δ_i (i.e. set to i) is $kn_i - \mu_i$. We achieve the metric by comparing this value to the number of cells representing the set when the filter reaches the state s (c_i):

Definition 7. *Given a filter $b^{\#}_{\dashv s}$, we define the* emersion *of each originating set Δ_i (and we refer to it as Ω_i) as:*

$$\Omega_i = \frac{c_{i\dashv s}}{c_{i\dashv i}} = \frac{c_i}{kn_i - \mu_i} \ . \tag{31}$$

The *emersion* of a set is included in the range $[0, 1]$. Clearly, if no cell among the $kn_i - \mu_i$ is overwritten, the set Δ_i would emerge completely. In this case $c_i = kn_i - \mu_i$ and thus $\Omega_i = 1$. The opposite condition is reached when all these cells are overwritten. In this case the set is completely "submerged": $c_i = 0$ and $\Omega_i = 0$.

Before the filter construction takes place, it is possible to calculate an expected value for the *emersion* of the i-th set considering the expected value of cells set to i in $b^{\#}_{\dashv s}$ and the one of those set to i in $b^{\#}_{\dashv i}$. Following (23) and (26) it is easy to derive:

$$E[\Omega_i] = \frac{E[c_{i\dashv s}]}{E[c_{i\dashv i}]} = \left(1 - \frac{1}{m}\right)^{kn_i^{\text{FILL}}} \ . \tag{32}$$

Having defined the emersion of a set, it is possible to express some of the above metrics as a function of the set's emersion. For instance, it is easy to note in relation to (28) and (29) that:

$$\text{ISEP}_i = (1 - E[\Omega_i])^k \ , \tag{33}$$

$$\text{and } E[\text{ISE}_i] = n_i \left(1 - E[\Omega_i]\right)^k \ . \tag{34}$$

Again, let us consider the *a posteriori* probability provided by (30). If we express it as $(1 - c_i/(kn_i - \mu_i))^k$, it is immediate to prove that

$$\text{ISEP}'_i = (1 - \Omega_i)^k \ . \tag{35}$$

Proposition 2. *Let us consider the filter $b^{\#}_{\dashv s}$ and the associated verification function $v^{\#}$. Given $\Omega_i = 1$, $\forall \delta \in \Delta_i$, $v^{\#}(\delta) = i$.*

Proof. We already proved that $\forall \delta \in \Delta_i$, $v^{\#}(\delta) \neq 0$. Hence, $v^{\#}(\delta) \neq i$ can only occur if the element δ is subject to an inter-set error. However, as $\Omega_i = 1$, $c_i = kn_i - \mu_i$, i.e., none of the cells related to Δ_i was overwritten. Hence, when we test δ for membership, the hash functions will address only cells set to i. □

Proposition 3. *Let us consider the filter $b^{\#}_{\dashv s}$ and the associated verification function $v^{\#}$. Given $j \in L$ and $\Omega_i = 0$, $\forall \delta \in \Delta_i$, $v^{\#}(\delta) = j$, $j > i$.*

Proof. The statement indicates that when a set has emersion equal to 0, all of its elements are wrongly recognized as belonging to a different set. The only way $v^{\#}(\delta) = i$ can occur is if the hash functions address at least one cell set to i. However, as $\Omega_i = 0$, $c_i = 0$, i.e., no cell in $b^{\#}_{\dashv s}$ is set to i. Therefore, as we already proved that $\forall \delta \in \Delta_i$ it is not possible to address cells set to 0, when we test δ for membership, the hash functions will address only cells set to j, where $j > i$ by SBF definition. □

From (35), it follows that, when $\Omega_i = 1$, $\text{ISEP}'_i = (1-1)^k = 0$ and, on the contrary, if $\Omega_i = 0$, $\text{ISEP}'_i = (1-0)^k = 1$. Again, following (20), we can state that when $\Omega_i = 0$, $\text{FPP}'_i = 0$. In fact, as $c_i = 0$:

$$\text{FPP}'_i = \left(\tfrac{0 + c_{i+1} + \cdots + c_s}{m}\right)^k - \text{FPP}'_{i+1} - \cdots - \text{FPP}'_s \ , \tag{36}$$

while

$$\text{FPP}'_{i+1} = \left(\tfrac{c_{i+1} + \cdots + c_s}{m}\right)^k - \text{FPP}'_{i+2} - \cdots - \text{FPP}'_s \ , \tag{37}$$

hence

$$\begin{aligned} \text{FPP}'_i = \left(\tfrac{c_{i+1} + \cdots + c_s}{m}\right)^k - \left(\tfrac{c_{i+1} + \cdots + c_s}{m}\right)^k + \text{FPP}'_{i+2} + \dots \\ + \text{FPP}'_s - \text{FPP}'_{i+2} - \cdots - \text{FPP}'_s = 0 \ . \end{aligned} \tag{38}$$

We can conclude noting that, when a set is completely emerged, there is no chance for its elements to be wrongly assigned to a different set:

$$\Omega_i = 1 \implies \text{ISEP}'_i = 0, \text{ISE}_i = 0, \text{ISER}_i = 0 \ . \tag{39}$$

Conversely, when a set is completely submerged, its elements will always be subject to inter-set errors:

$$\Omega_i = 0 \implies \text{ISEP}'_i = 1, \text{ISE}_i = n_i, \text{ISER}_i = 1 \ . \tag{40}$$

Moreover, it is easy to derive from Proposition 3 that, when a set is completely submerged, it is impossible to report false positives on that specific set as well:

$$\Omega_i = 0 \implies \text{FPP}'_i = 0, \text{FPR}_i = 0 \ . \tag{41}$$

Again, it is important to note that the last set to be processed during the filter construction (Δ_s) features some interesting properties.

Proposition 4. *Let us consider the filter $b^{\#}_{\dashv s}$ and the associated verification function $v^{\#}$. Then, $\forall \delta \in \Delta_s$, $v^{\#}(\delta) = s$.*

Proof. Following Definition 7, it is evident that $\Omega_s = c_{s\dashv s}/c_{s\dashv s} = 1$. Hence, the assertion may be derived directly from Proposition 2. □

Again, we may notice that, as $n_s^{\text{FILL}} = 0$,

$$\text{ISEP}_s = (1 - (1 - 1/m)^0)^k = 0 \ . \tag{42}$$

It follows that the set Δ_s is never affected by inter-set errors.

As a final consideration, we briefly discuss the emersion of single elements. Each element of a set Δ_i reaches its maximum degree of representation when the filter reaches the state i. At this stage, it is represented by k cells, minus the number of *intra-element collisions* (see Section II-A). For the j-th element of Δ_i, this number of cells may be represented as $c_{ij\dashv i}$. Hence, the emersion of a single element may be defined as:

$$\omega_{ij} = \frac{c_{ij\dashv s}}{c_{ij\dashv i}}. \tag{43}$$

In order to recognize a specific element as belonging to the proper set, it is sufficient that one of the cells related to that element is not overwritten. The only remarkable case occurs when all these cells are overwritten, i.e. when $\omega_{ij} = 0$. This condition leads to an *inter-set error*, which is discussed in the paper in Section III-E. We also note that when an element is checked for membership, it is not possible to discern a false positive in a element-specific fashion. False positives are in fact related to sets, and their probability depends on the overall number of labels (representing the set) stored in the filter at the end of its construction. This is why emersion is discussed on a set basis in this context.

V. Safeness

In certain application settings (as discussed in Section VIII) the presence of inter-set errors may discourage usage of spatial Bloom filters. It is thus important to understand how it is possible to build filters that are not subject to these errors.

Definition 8. *Given a filter $b^{\#}_{\dashv s}$, we say that the originating set Δ_i is* safe *when* $\text{ISER}_i = 0$.

Thus, a safe set is never affected by inter-set errors for the specific filter $b^{\#}_{\dashv s}$. We note that, following Proposition 2, when $\Omega_i = 1$ the set Δ_i is *safe* and, similarly, following Proposition 4, the set Δ_s is always *safe*. Within the context of the probabilistic model proposed in Section III-E, we can calculate the *a priori* probability for a specific set to be safe at the end of the construction process. Let us consider (28) which represents the probability for a specific member of Δ_i to be affected by an inter-set error. Hence, $1 - \text{ISEP}_i$ is the probability for that element not to be affected by this kind of error. It follows that, the probability for all of the Δ_i's members not to be affected by inter-set errors is

$$\text{SAFEP}_i = (1 - \text{ISEP}_i)^{n_i} \tag{44}$$

which represents in turn the probability for that set to be safe. Now that we know this set-specific probability, we can discuss the *safeness* property concerning the entire filter.

Definition 9. *A filter $b^{\#}_{\dashv s}$ is* safe *when $\forall i \in L$, $\text{ISER}_i = 0$.*

Thus, we consider a SBF to be safe when it is not affected by inter-set errors for any of the originating sets. We conclude by calculating the *a priori* probability for a SBF to be safe. In order to reach the safeness condition, each of the filter's originating sets needs to be safe. Hence, as we can consider these events as independent, we say that:

$$\text{SAFEP} = \prod_{i=1}^{s} (1 - \text{ISEP}_i)^{n_i} \ . \tag{45}$$

This is an important result as this probability enables us to estimate how many times, on average, we need to build a filter using different (random) hash functions before achieving a version of it that is safe. In practice, as we discuss in Section VI, hash functions are combined with a random hash salt during the construction process in the proposed implementation. This makes it straightforward to perform several trials upon the same construction set $\bar{S}$ until a safe version of the filter is produced.

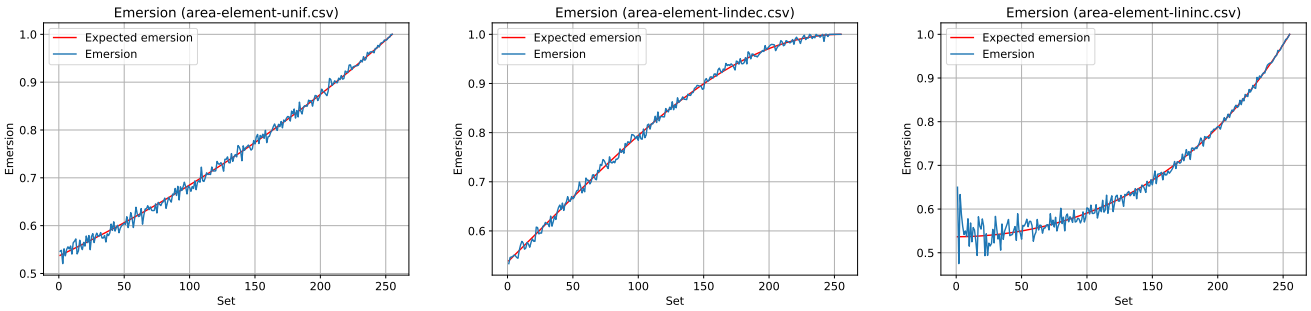
VI. Implementation and experiments

We provide a reference implementation of the spatial Bloom filter data structure, including the original building and querying algorithms as well as functions for the calculation of the novel probabilistic metrics proposed in this paper. The implementation is available both in C++ and Python, as a library providing the `SBF` class, and is released under the Lesser General Public License (LGPL). The latest version of the library can be found on GitHub:
`https://github.com/spatialbloomfilter/`.

TABLE III
Datasets used in the experiments.

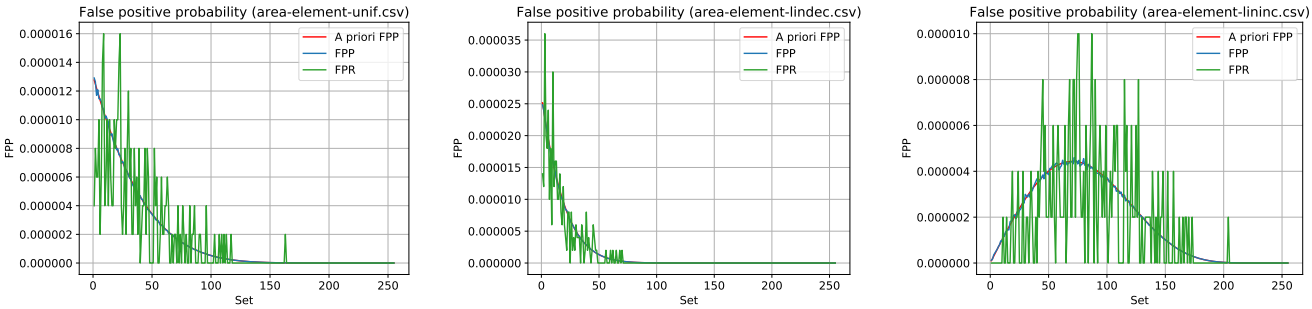
8-bit datasets	Elements (n)	Sets (s)	Elem. allocation, number (n_i)
area-element-unif	65 280	255	Linear, 256
area-element-lindec	65 280	255	Decremental, $510, \ldots, 2$
area-element-lininc	65 280	255	Incremental, $2, \ldots, 510$
area-element-rand	65 280	255	Random, varies in $[209, 298]$
non-elements	500 000	-	-
8-bit-large datasets	**Elements (n)**	**Sets (s)**	**Elem. allocation, number (n_i)**
area-element-unif	16 776 960	255	Linear, 65 792
16-bit datasets	**Elements (n)**	**Sets (s)**	**Elem. allocation, number (n_i)**
area-element-unif	16 776 960	65 535	Linear, 256
area-element-lindec	16 776 960	65 535	Decremental, $510, \ldots, 2$
area-element-lininc	16 776 960	65 535	Incremental, $2, \ldots, 510$
area-element-rand	16 776 960	65 535	Random, varies in $[191, 324]$
non-elements	14 097 123	-	-

The two implementations are equivalent in terms of capabilities, and have been extensively tested in order to ensure they both return the same expected results. However, the C++ implementation is optimized for speed of execution (performance data are available in the supplemental material, see the Appendix for reference), while the Python library provides an easier interface for scripting and plot generation. For the latter purpose, the Python implementation is accompanied by a further `sbfplot` library, providing a number of function-plotting routines for the probabilistic metrics presented in this paper. Both the C++ and the Python implementations use an efficient memory allocation mechanism, adaptively reducing or increasing the memory required to store the filter (in terms



(a) `unif` (b) `lindec` (c) `lininc`

Fig. 2. Emersion for random, decremental and incremental datasets (8-bit). As expected, the `lindec` dataset features the highest emersion over most sets.



(a) `unif` (b) `lindec` (c) `lininc`

Fig. 3. False positive probabilities for the random, decremental and incremental datasets, in the 8-bit version. FPR is the actual false positive rate for the constructed filters, obtained querying each SBFs with 500 000 random elements (contained in the `non-elements` dataset) external to the filter.

of number of bytes per cell) depending on the number of sets. Both implementations also provide a choice of hash standards, including MD4, MD5, and SHA-1[1]. Different hash functions are derived from the same hash standard by using a random hash salt.

In order to test the implementation, we generated a number of test datasets (also available on GitHub). The datasets are composed by a number of element-set tuples, where elements are randomly generated and the sets are assigned to form datasets of specific geometry (Table III). In particular, we used `unif` datasets, where elements are equally distributed among sets, `lindec` datasets, where elements are assigned to sets following a linearly decreasing distribution function, `lininc` datasets, which follow a linearly increasing distribution instead, and `rand` datasets, where the elements are assigned to sets randomly. For each distribution, we produced a smaller, 8-bit version of the dataset (where 8-bit is the memory space used to store the set labels) with 255 sets and 65 280 total elements, as well as a larger 16-bit version with 65 535 sets and 16 776 960 elements. The different geometry of the datasets enabled the testing of different scenarios and how the false positive and error probabilities vary as a result. For the properties of the SBF, it is evident that a linearly decremental distribution is more advantageous, for example, than a linearly incremental one, as in the former less elements will be member of sets with the highest priority, and therefore the highest overwriting capacity. Each datasets was then used to construct a corresponding filter, enabling a comparison between the expected probabilistic characteristics and the ones featured by an actual filter. The filters were created using optimal values of m and k, with respect to the overall number of elements contained in the dataset. Specifically, $m = 2^{20}$ and $k = 10$ for 8-bit datasets; and $m = 2^{28}$ and $k = 10$ for the 16-bit and 8-bit-large. The MD5 hash standard was used, applying different hash salts for each function. The same salts were used for all datasets.

We analyze the results of the experiments in Section VII. While we include the function plots that are most useful for discussion in this paper, the entire set of plots is available as supplemental material (see the Appendix).

[1]It is important to note here that the security of the hash standard is not a concern in this setting, as the hash functions only serve a mapping purpose. We can therefore safely use MD5, for instance, even though it is subject to known attacks. For the same reason, non-cryptographic uniform mapping functions could be used as an alternative.

VII. Analysis

On the basis of the experiments described above, in the following we analyze how the spatial Bloom filter data structure performs, with respect to the probabilistic features presented in this paper. We discuss each characteristic separately.

A. Number of cells

For each 8-bit dataset we computed the number of expected cells per set according to (26). Then, we compared this value

with the real number of cells representing the set at the end of the filter construction (c_i). The results show that the effective number of cells follows closely the expected value. Intuitively, the higher the set label, the closer the number of cells is to the maximum value ($kn_i - \mu_i$), as overwrite events decrease. Graphs showing the number of cells are provided as supplemental material (see the Appendix) for brevity.

B. Emersion

Figure 2 shows how the emersion (31) observed for each set compares to the expected emersion, defined in (32) (for brevity, the random dataset is included as supplemental material). The results indicate that the expected emersion correctly predicts the *a posteriori* value. In general, higher set labels correspond to a higher emersion value, as could be intuitively expected. The degree to which this phenomenon is displayed for different datasets depends on the number of elements per set: where most elements belong to lower labeled sets, the emersion grows more rapidly, as in the case of the linear decremental distribution. This can be easily explained with the fact that the probability of overwrites depends on the number of elements that belong to sets with higher labels.

C. False positives

The false positive probability is one of the most important characteristics in a spatial Bloom filter. While the absolute value of this probability depends on the size of the filter m and the number of hash functions k, it is interesting to study how the probability varies on a set-by-set basis for the different datasets, when m and k are kept constant. Figure 3 shows that the `lindec` setting is the one which preserves the highest number of sets from false positive events, while at the same time also featuring the highest false positive probability among the datasets. The `lininc` dataset deserves attention as well: as we may see, the curve has a concave part with a maximum value around the 75-th set. This behavior is due to the specific distribution of elements across sets. In fact, the first originating sets have few members and thus, they have few cells representing them in the filter. This is why the false positive probability stays low even if these sets should be the ones most affected by false positives, due to their low labels. We can say that in this part of the graph the number of elements is more prominent in influencing the probability than the set label. When we reach the maximum, the value of the set label becomes predominant in determining the FPP, and the false positive probability starts to decrease.

The charts compare the *a priori* false positive probability (16) with the *a posteriori* one (20), showing the correctness of the proposed model, as the values substantially coincide. The false positive rate (7) follows the expected trend as well.

D. Inter-set errors

While false positives also appear in Bloom filters, the inter-set error phenomenon is characteristic of spatial Bloom filters. Similarly to the case of FPP, the experiments provide a comparison between the *a priori* inter-set error probability (28) with the *a posteriori* one (30). The the effective error rate (8) was obtained after the filter construction by performing membership queries for all elements in the originating sets. These quantities are plotted in Figure 4, which evidences the correctness of the inter-set error probabilistic model. As could be expected, the last sets are the most preserved from inter-set errors events.

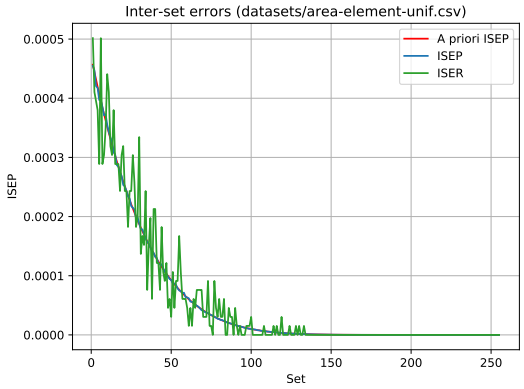


Fig. 4. The inter-set errors for the 8-bit large `unif` dataset. ISER is the actual inter-set error rate observed on the constructed filter.

A final remark: in this case, the set that best displays the inter-set error behavior is the 8-bit large uniform dataset, which contains 65 792 elements per set. This is significantly higher than the number of members per set of the standard 8-bit datasets, which has 256 elements per set. In the latter, because of the reduced number of elements, a single inter-set error alters significantly the ISER for the corresponding set, reducing the statistical significance and resulting in apparently unexpected peaks in the chart. The chart is available as part of the supplemental material (see the Appendix).

E. Safeness

In conclusion, we discuss the probability for each set to be safe, defined in (44). The results are proposed in Figure 5.

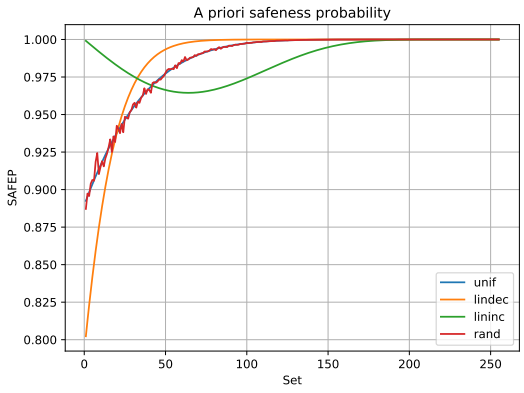


Fig. 5. The safeness probability for the 4 test datasets (8-bit).

As predictable, the `lindec` setting is the one where the highest number of sets are more likely to be safe, while the `lininc` features a lower variance across sets. Similarly to

the false positives case (see Section VII-C), the `lininc` setting produces a convex region in the SAFEP curve. This phenomenon depends on the specific element distribution: the first sets have few elements, hence, their chances to be affected by inter-set errors are few as well, despite their disadvantageous position in the construction order. When we reach the minimum, the set label becomes prominent over the element distribution, and SAFEP increases. Discussion of the overall filter safeness is postponed to Section VIII-B, where we discuss a specific context where this property is considerably meaningful.

VIII. Relevance to security protocols

The spatial Bloom filter is an efficient data structure, that has the potential to become a crucial building block in several of the cryptographic protocols that currently feature the classic Bloom filter. Adoption of the SBF, through the reduction in the number of filters and the aggregation of cryptographic operations can increase both memory and communication efficiency of the constructions, while decreasing the computational burden, as exemplified by the location privacy case [9]. However, this is possible only when the optimal parameters for the filter can be selected. The probabilistic metrics we propose in this paper provide crucial tools in this direction. In the following, we discuss three relevant security scenarios, in order to demonstrate how metrics such as emersion and safeness can be used to construct optimal filters for each context: as each scenario is different, the filter characteristics must be tuned to adapt to the probabilistic properties (false positive and error probabilities) required in each setting.

A. Location privacy and monitoring of critical areas

Spatial Bloom filters were first proposed in the context of location privacy, and have since been used in a number of related protocols [9]. In particular, the SBF data structure can be used to store geographical information of sensitive areas or points of interests by providers of location-based services. A filter encrypted with a homomorphic encryption scheme can provide two-sided privacy protection: the provider can maintain information on the critical areas hidden, while users can be offered the guarantee that their location is disclosed only when inside one of these predefined areas [7].

In this context, the filter would be built over geographical positions considered of interest, classified into different sets. The different sets can also be used to encode the distance from a point of interest, for example by using concentric areas, with the set label increasing with vicinity to the point of interest. In this setting, a false positive or an inter-set error reported for an outer area would have less impact than the same error reported for the central region. Therefore, the construction parameters of the filter should be chosen to minimize the inter-set error probability and false positive probability, in particular for higher-labeled sets. As shown in Figures 3 and 4, the last sets are naturally the most preserved from inter-set errors and false-positives, especially in the linear decremental environment. This dataset exemplifies the discussed scenario effectively, as for concentric areas, set size decreases for inner sets. Thus, the highest label should be used for the sets covering the central region, while marking the outermost (or least significant) area with label 1. Finally, it is important to note that the central region (highest label) is always safe, i.e. it is never affected by inter-set errors (see Proposition 4 for reference).

B. (Anonymous) network routing

Bloom filters, and more recently the SBF have been used in private routing protocols [10]. In this context, the filters are used to identify the correct network (or in the case of Tor like protocols, the correct relay) to which the current routing node needs to forward the traffic. In particular, filters map the network addresses of the nodes of the network as well as the subnetwork they belong to. A SBF can be used to store this information efficiently: the network addresses are the elements to be inserted into the filter, and their corresponding subnetworks are the sets. An encrypted filter can be used to preserve the privacy of the traffic destination: the use of partially homomorphic encryption will enable a routing node to determine correctly the needed routing information (the destination network) without disclosure of the identity of the destination node.

In this setting, it is crucial to avoid inter-set errors: in fact, if querying the filter with the address of a node results in the the wrong subnetwork, the node becomes unreachable. False positives, to the contrary, are not relevant in this scenario: packets addressed to unknown nodes will always be discarded eventually. Therefore, a protocol based on spatial Bloom filters will need to minimize or avoid altogether inter-set errors, and therefore maximize the SAFEP; with the liberty of ignoring even high false positive probability values. As only the actual (*a posteriori*) safeness is relevant, and in order to minimize the filter size and hence increase the efficiency, the filter construction can be iterated until a safe SBF is returned (see Section V for reference). In Table IV we show the probability to obtain a safe SBF as the size of the filter m varies. For a fixed overall number of elements and number of sets, members distribution among different sets does not affect significantly the safeness probability. For each of the listed datasets, a value of $m = 2^{21}$ is sufficient to reach a probability near 1 to build a safe filter. Using the iteration strategy described above, a significantly smaller $m = 2^{20}$ filter will require roughly 32 iterations on average to produce a safe version. For the inter-network routing scenarios, the randomly assigned dataset `rand` is arguably the most fitting test.

TABLE IV
THE SAFENESS PROBABILITY OF A FILTER IN RELATION TO ITS SIZE.

8-bit datasets	$(m = 2^{20})$	$(m = 2^{21})$	$(m = 2^{22})$	$(m = 2^{23})$
area-element-unif	0.03131	0.98764	0.99998	1.00000
area-element-lindec	0.03292	0.98784	0.99998	1.00000
area-element-lininc	0.03062	0.98754	0.99998	1.00000
area-element-rand	0.03131	0.98764	0.99998	1.00000

C. Border controls

Border controls are increasingly relevant as both terrorist episodes and migration flows grow in frequency and size

respectively. Since *electronic Machine-Readable Travel Documents* (eMRTD) were introduced and e-Passport issuance specifications were outlined by *International Civil Aviation Organization* (ICAO) [13], the automated inspection procedure is one of the most critical elements concerning border controls, as it was designed to enhance the inspection quality (thanks to a number of security protocols aimed at protecting the chip from data tampering or counterfeiting) while speeding up the border outflow.

Similarly to the classic Bloom filter, which has been successfully used within this domain for biometric recognition [14], SBF might be used to enhance border security through a privacy-preserving partition of people among several sets representing the associated level of risk. Lists of people (uniquely identified by e-Passport numbers) deemed to require additional border attention by national authorities can be stored into an SBF, with the risk level representing the SBF set label. During automated passport checks (which ensure the passport is authentic and belongs to the bearer) border officers may be advised to perform an in-depth inspection on specific individuals, depending on the risk level returned by the SBF. SBF encryption can preserve the privacy of the individuals, while homomorphic comparison can allow sharing of the lists between different countries.

This scenario is characterized by a few originating sets (the risk classes) and a lot of elements per set (individuals to be subject to additional checks) similarly to the case represented by the 8-bit large dataset. In this case, the SBF should be constructed preserving the most risky classes (i.e. assigning them an higher set label), following a linearly decremental distribution of members across sets, and focusing on false positives. In fact, border controls are often performed in a crowded environment and should not be slowed down by a mass of useless in-depth inspections caused by false positives. Inter-set errors are meaningful but not so much problematic here, as they can occur only in one direction: a risky person may be exchanged with a more risky one. The opposite condition can never occur (see Section III-E for reference).

IX. Conclusion

In this paper we propose a comprehensive study of the probabilistic properties of spatial Bloom filters, and we introduce several novel metrics that provide an immediate understanding of the characteristics of a filter, thus facilitating SBF understanding and usage. Each of the discussed properties and metrics is supported by extensive experimental evidence, obtained using the SBF implementation we propose as part of this work. Two libraries implementing the data structure are provided, in C++ and Python, released under LGPL terms. The presented results prove the correctness of the proposed probabilistic model and will enable both the scientific community and industry to embed SBF in a number of privacy and security settings, as discussed in the final part of this work.

Appendix
Supplemental material

For brevity, only some of the graphs plotting the experimental results are included here. The comprehensive set of charts for all datasets in Table III is provided as supplemental material, available online at: `https://git.io/v73lY`.

Acknowledgment

The authors would like to thank Fabrizio Caselli for insightful discussions on the subject.

References

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

[2] S. Geravand and M. Ahmadi, "Bloom filter applications in network security: A state-of-the-art survey," *Computer Networks*, vol. 57, no. 18, pp. 4047–4064, 2013.

[3] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Proceeding*, ser. Lecture Notes in Computer Science, J. Stern, Ed., vol. 1592. Springer, 1999, pp. 223–238.

[4] S. M. Bellovin and W. R. Cheswick, "Privacy-enhanced searches using encrypted bloom filters," *IACR Cryptology ePrint Archive*, vol. 2004, p. 22, 2004.

[5] R. Schnell, T. Bachteler, and J. Reiher, "Privacy-preserving record linkage using bloom filters," *BMC Med. Inf. & Decision Making*, vol. 9, p. 41, 2009.

[6] N. Zeilemaker, Z. Erkin, P. Palmieri, and J. A. Pouwelse, "Building a privacy-preserving semantic overlay for peer-to-peer networks," in *2013 IEEE International Workshop on Information Forensics and Security, WIFS 2013*. IEEE, 2013, pp. 79–84.

[7] P. Palmieri, L. Calderoni, and D. Maio, "Spatial bloom filters: Enabling privacy in location-aware applications," in *Information Security and Cryptology - 10th International Conference, Inscrypt 2014, Revised Selected Papers*, ser. Lecture Notes in Computer Science, D. Lin, M. Yung, and J. Zhou, Eds., vol. 8957. Springer, 2014, pp. 16–36.

[8] L. Calderoni, P. Palmieri, and D. Maio, "Location privacy without mutual trust: The spatial bloom filter," *Computer Communications*, vol. 68, pp. 4–16, 2015.

[9] M. G. Solomon, V. S. Sunderam, L. Xiong, and M. Li, "Enabling mutually private location proximity services in smart cities: A comparative assessment," in *IEEE International Smart Cities Conference, ISC2 2016*. IEEE, 2016, pp. 1–8.

[10] P. Palmieri, L. Calderoni, and D. Maio, "Private inter-network routing for wireless sensor networks and the internet of things," in *Proceedings of the Computing Frontiers Conference, CF 2017*. ACM, 2017, pp. 396–401.

[11] J. K. Mullin, "A second look at bloom filters," *Commun. ACM*, vol. 26, no. 8, pp. 570–571, 1983.

[12] K. J. Christensen, A. Roginsky, and M. Jimeno, "A new analysis of the false positive rate of a bloom filter," *Inf. Process. Lett.*, vol. 110, no. 21, pp. 944–949, 2010.

[13] International Civil Aviation Organization, *Doc 9303 - Machine Readable Travel Documents*, 7th ed. ICAO, 2015, vol. 1-12.

[14] C. Rathgeb, F. Breitinger, and C. Busch, "Alignment-free cancelable iris biometric templates based on adaptive bloom filters," in *International Conference on Biometrics, ICB 2013*, 2013, pp. 1–8.

Luca Calderoni received a Ph.D. degree in computer science from the University of Bologna, where he is currently a Post-doctoral Researcher. His research activity focuses on privacy and security in digital systems and smart cities. He has published on location privacy, border controls, secure and privacy-preserving tracking technologies and cryptographic protocols.

Paolo Palmieri is a Lecturer in Cyber Security at the Dept. of Computer Science, University College Cork. He holds a PhD in cryptography from the Université Catholique de Louvain. His research work focuses on cryptographic protocols for privacy and anonymity, and he has worked on secure computation, location privacy, and the security of smart cities and IoT.

Dario Maio is a Full Professor of Information Systems with the Dept. of Computer Science and Engineering, University of Bologna. He is a member of IEEE, ACM and IAPR. He has published more than 200 research papers investigating various aspects of computer science including distributed computer systems, computer performance evaluation, database design, information systems, autonomous agents, pattern recognition, and biometric systems.