A Privacy Model for Classical & Learned Bloom Filters

Hayder Tirmazi
City College of New York
hayder.research@gmail.com

Keywords: Differential Privacy, Adversarial Artificial Intelligence, Probabilistic Data Structures

Abstract: The Classical Bloom Filter (CBF) is a class of Probabilistic Data Structures (PDS) for handling Approximate Query Membership (AMQ). The Learned Bloom Filter (LBF) is a recently proposed class of PDS that combines the Classical Bloom Filter with a Learning Model while preserving the Bloom Filter's one-sided error guarantees. Bloom Filters have been used in settings where inputs are sensitive and need to be private in the presence of an adversary with access to the Bloom Filter through an API or in the presence of an adversary who has access to the internal state of the Bloom Filter. This paper conducts a rigorous differential privacy-based analysis for the Bloom Filter. We propose constructions that satisfy differential privacy and asymmetric differential privacy. This is also the first work that analyses and addresses the privacy of the Learned Bloom Filter under any rigorous model, which is an open problem.

1 INTRODUCTION

Probabilistic Data Structures that give approximate answers to membership queries are called AMQ-PDS (Filić et al., 2022). The Bloom Filter is one of the most common AMQ-PDS. The Bloom Filter has numerous applications including in databases, cryptography, computer networking, social networking (Bose et al., 2008), and network security (Broder and Mitzenmacher, 2003). The Learned Bloom Filter (LBF) is a novel data structure invented in 2017 (Kraska et al., 2018). We refer to a Bloom Filter that is not an LBF as a Classical Bloom Filter (CBF).

A CBF that stores a set S may have false positives ($s \notin S$ may return true) but it never has false negatives ($s \in S$ is always true). An LBF provides the same one-sided error guarantee (no false negatives) as a CBF but with potentially better performance for the same memory budget (Mitzenmacher, 2018a; Mitzenmacher, 2018b; Bishop and Tirmazi, 2024). In this work, we use Bloom Filter (BF) as a blanket term that includes both LBFs and CBFs. An LBF can be thought of as a CBF working in collaboration with a Learning Model. Figure 1 shows a Standard LBF.

1.1 Contributions

The fundamental open problem this work tries to solve is: *How can we provably protect the privacy of data stored in a Bloom Filter?*

Differential privacy for Bloom Filters. We propose the first rigorous privacy framework for BFs with provable guarantees under the (ε, δ)-differential privacy model and a well-known relaxation of it called the $(\varepsilon_1, \varepsilon_2, \delta)$-asymmetric differential privacy model. We also discuss set privacy which provides an intuitive and rigorous measure of privacy for unordered sets in the context of AMQ-PDS and enables privacy-preserving algorithms that can be generalized to any BF or AMQ-PDS construction.

Provably private constructions. We introduce Nickel and Dime, privacy-preserving algorithms based on randomized response mechanisms. The algorithms apply to both Classical and Learned BFs. Instead of changing the internal state of the BF, the algorithms modify the input set stored by the BF. This approach can be generalized to other AMQ-PDS. Nickel adds privacy without sacrificing the one-sided error guarantees of the BF. Dime adds privacy but introduces a false negative probability to the BF.

First privacy analysis for Learned Bloom Filters. To the best of our knowledge, this work is the first to address and analyze the privacy of LBFs under a rigorous mathematical framework.

1.2 Related Work

CBF Security. (Gerbet et al., 2015) suggests practical attacks on CBFs and the use of universal hash functions and MACs to mitigate a subset of those at-

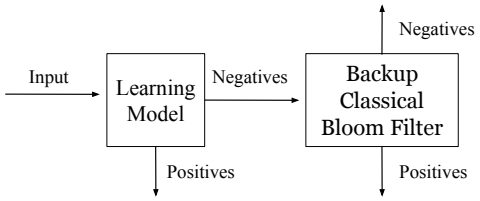

Figure 1: Standard Learned Bloom Filter (SLBF) with a Learning Model (LM) and a Backup Classical Bloom Filter (CBF). The Backup CBF only checks values that, according to the LM, are not in S_Π. This ensures a one-sided error bound (no false negatives) on the entire construction. For an element x, A positive output implies the Learning Model determined that $x \in S_\Pi$, and a negative output implies it determined $x \notin S_\Pi$.

tacks. (Naor and Eylon, 2019) define an adversarial model for CBFs and provide a method for constructing adversary-resilient CBFs. (Naor and Oved, 2022) present several robustness notions in a generalized adversarial model for CBFs. (Clayton et al., 2019) and (Filić et al., 2022) also provide secure constructions for CBFs using a game-based and a simulator-based model respectively. None of these works address *Learned* PDS including the LBF.

LBF Security. We are aware of only two prior works that address the Learned Bloom Filter in an adversarial setting, (Reviriego et al., 2021) and (Bishop and Tirmazi, 2024). Reviriego et al. propose a practical attack on LBFs. They suggest possible mitigations e.g. swapping to a CBF upon attack detection. However, they do not provide any provable security guarantees for LBFs. Reviriego et al. leave the security of the LBF as an open problem. Bishop and Tirmazi propose provably secure LBF constructions by extending the adversarial model of Naor and Eylon.

CBF Privacy. Many works including (Sengupta et al., 2018), (Reviriego et al., 2023), and (Galan et al., 2023) have shown that CBFs are vulnerable to set reconstruction attacks i.e. given the internal state σ it is possible to infer the set the CBF stores with high probability. (Bianchi et al., 2012) provides privacy metrics for CBFs. Bianchi et al.'s metrics are based on k-anonymity (Sweeney, 2002). (Filić et al., 2022) propose a simulator-based notion of privacy for CBFs based on information leakage profiles. They provide privacy bounds for CBFs that use pseudo-random functions on their input set. Filić et al.'s proposal does not achieve meaningful privacy for CBFs whose input sets have low min-entropy and their notion of **Elem-Rep** privacy is not immune to set reconstruction attacks from computationally unbounded adversaries.

LBF Privacy. We are not aware of any specific prior work analyzing the privacy of Learned Bloom Filters.

2 PRELIMINARIES

We borrow and unify the treatment of AMQ-PDS from a large body of prior work (Filić et al., 2022; Filić et al., 2024; Gerbet et al., 2015; Bishop and Tirmazi, 2024; Broder and Mitzenmacher, 2003; Naor and Eylon, 2019).

Notation. Given a set S, we write $x \leftarrow\!\$\ S$ to mean that x is sampled uniformly randomly from S. For a set S, we denote by $|S|$ the number of elements in S. Similarly, for a list L, $|L|$ is the number of elements in L. A fixed-length list of length m initialized empty is denoted by $L \leftarrow \perp^n$. The i^{th} entry in list l is $l[i]$. We write variable assignments using $\leftarrow$. If the output is the value of a randomized algorithm, we use $\leftarrow\!\$$ instead. For a randomized algorithm A, we write output $\leftarrow \mathrm{A}_r(\mathrm{input}_1, \mathrm{input}_2, \cdots, \mathrm{input}_l)$, where $r \in \mathcal{R}$ are the random coins that can be used by A and $\mathcal{R}$ is the set of possible coins. For a natural number n, we denote the set $\{1, \cdots, n\}$ by $[n]$. $\textcircled{¢}_p$ indicates a biased coin with probability p of returning heads. $\textcircled{h}$ indicates heads, $\textcircled{t}$ indicates tails.

2.1 AMQ-PDS

We formalize the general syntax and behavior of AMQ-PDS. Given an AMQ-PDS, Π, we denote the set of public parameters of an AMQ-PDS by Φ. We denote the set of elements stored in Π by S_Π. We denote the state of Π by $\sigma \in \Sigma$ where Σ is the space of all possible states of Π. Π can store elements from any finite domain $\mathfrak{D}$, where $\mathfrak{D} = \cup_{l=0}^{L} \{0,1\}^l$ for any natural number $L \in \mathbf{N}$. An AMQ-PDS, Π consists of two algorithms.

Construction. $\sigma \leftarrow C_r(\Phi, S_\Pi)$ sets up the initial state of an empty AMQ-PDS with public parameters Φ and a given set $S_\Pi \subseteq \mathfrak{D}$.

Query. $b \leftarrow Q(x, \sigma)$, given an element $x \in \mathfrak{D}$ returns a boolean $b \in \{\perp, \top\}$. The return value approximately answers whether $x \in S_\Pi$ ($b = \top$) or $x \notin S_\Pi$ ($b \neq \perp$).

The construction algorithm C_r is called first to initialize Π. The query algorithm Q is not allowed to change the value of the state. While C_r is randomized, Q is deterministic. Both algorithms always succeed. A class of AMQ-PDS can be uniquely identified by its algorithms: $\Pi = (C_r, Q)$. All AMQ-PDS have the following properties.

Definition 2.1 (AMQ-PDS)**.** *$\Pi = (C_r, Q)$ is an $(n, \varepsilon_p, \varepsilon_n)$-AMQ-PDS if for all sets $S_\Pi \subseteq \mathfrak{D}$ of cardinality n and suitable public parameters Φ, the following two properties hold.*
P-Soundness: $\forall x \notin S_\Pi : P[Q(x, C_r(\Phi, S_\Pi)) = \top] \leq \varepsilon_p$

N-Soundness: $\forall x \in S_\Pi : P[Q(x, C_r(\Phi, S_\Pi)) = \bot] \leq \varepsilon_n$
where the probabilities are over the coins of C_r.

2.2 Bloom Filter

A Bloom Filter (BF) is any member of a class of AMQ-PDS whose query algorithm can yield false positives but not false negatives (Bloom, 1970). We formalize this notion with the following definition (Naor and Eylon, 2019):

Definition 2.2 (Bloom Filter)**.** $\Pi = (C_r, Q)$ *is an* (n, ε)*-BF if for all sets* $S_\Pi \subseteq \mathfrak{D}$ *of cardinality n and suitable public parameters* Φ*, the following two properties hold.*
Completeness: $\forall x \in S_\Pi : P[Q(x, C_r(\Phi, S_\Pi)) = \top] = 1$
Soundness: $\forall x \notin S_\Pi : P[Q(x, C_r(\Phi, S_\Pi)) = \top] \leq \varepsilon$
where the probabilities are over the coins of C_r.

Note that Def. 2.2 holds for both CBFs and LBFs.

2.3 Learned Bloom Filter

We use definitions consistent with common mathematical treatments of the Learned Bloom Filter (Mitzenmacher, 2018a; Mitzenmacher, 2018b). Let $S_\Pi \subseteq \mathfrak{D}$ be the set stored in the LBF. Consider any set $P \subseteq S_\Pi$ and $N \subseteq \mathfrak{D} \setminus S_\Pi$. We denote the set $\mathcal{T} = \{(x_i, y_i = 1) | x_i \in P\} \cup \{(x_i, y_i = 0) | x_i \in N\}$ as a training dataset. LBFs use machine learning models (LMs) trained from the training dataset, which we formalize below.

Definition 2.3 (Learning Model)**.** *Let* $\mathcal{L} : \mathfrak{D} \mapsto \{\bot, \top\}$ *be any function that maps elements in* $\mathfrak{D}$ *to a boolean.* $\mathcal{L}$ *is an* $(S_\Pi, \varepsilon_p, \varepsilon_n)$*-LM, if for set* $S_\Pi \subseteq \mathfrak{D}$ *the following two properties hold.*
P-Soundness: $\forall x \notin S_\Pi : P[\mathcal{L}(x) = \top] \leq \varepsilon_p$
N-Soundness: $\forall x \in S_\Pi : P[\mathcal{L}(x) = \bot] \leq \varepsilon_n$

Definition 2.4 (Learned Bloom Filter)**.** $\Pi = (C_r, Q)$ *is an* $(n, \varepsilon, \varepsilon_p, \varepsilon_n)$*-LBF if for all sets* $S_\Pi \subseteq \mathfrak{D}$ *of cardinality n and suitable public parameters* Φ*:* Π *is an* (n, ε)*-BF and* Q *uses an* $(S_\Pi, \varepsilon_p, \varepsilon_n)$*-LM.*

The training dataset $\mathcal{T}$ is typically used internally by the construction algorithm C_r to create an $(S_\Pi, \varepsilon_p, \varepsilon_n)$-LM $\mathcal{L}$ which is stored as part of the state σ of the AMQ-PDS Π. Q then extracts this LM from σ and invokes it when answering a membership query.

In many scenarios, LBFs provide lower false positive rates than CBFs for the same memory budget (Kraska et al., 2018; Mitzenmacher, 2018a). We provide intuition for this with an example. Consider a database storing (key, value) pairs. The database wants to check whether a given key exists with minimal delay. A CBF is memory-efficient but may introduce a higher false positive count, leading to wasted lookups. An LBF, on the other hand, can train on the data distribution to filter queries more effectively, reducing the false positive rate and thus the computational overhead of unnecessary database accesses.

$C_r(\Phi, S_\Pi)$	$Q(x, \sigma)$
$m, k \leftarrow \Phi$; $\sigma \leftarrow 0^m$	$b \leftarrow 0^m$
$\forall_{x \in S_\Pi} \forall_{i \in [k]}\ \sigma \leftarrow \sigma \vee \hat{h}_{i,m}(x)$	$\forall_{i \in [k]}\ b \leftarrow b \vee \hat{h}_{i,m}(x)$
return σ	**return** $[b = \sigma \wedge b]$

Figure 2: AMQ-PDS syntax instantiation for the Standard Classical Bloom Filter (SCBF).

2.4 Constructions

Standard CBF. There are many constructions of the Classical Bloom Filter (Broder and Mitzenmacher, 2003; Gupta and Batra, 2017; Abdennebi and Kaya, 2021). We discuss the most common construction, the Standard Classical Bloom Filter (SCBF). The SCBF construction requires a family of k independent hash functions, $h_{i,m} : \mathfrak{D} \mapsto [m]$ for all $i \in [k]$.

In SCBF, σ is a zero-initialized array of m bits. Upon setup, For each element $x \in S_\Pi$ (recall S_Π is the set being encoded by the Bloom Filter), the bits $h_i(x)$ are set to 1 for $i \in [k]$. When querying an element x, we return true if all $h_i(x)$ map to bits that are set to 1. If there exists an $h_i(x)$ that maps to a bit that is 0, we return false.

Definition 2.5. $\hat{h}_{i,m} : \mathfrak{D} \mapsto \{0,1\}^m$ *maps an input* $x \in \mathfrak{D}$ *to an m-bit array where all bits are* 0 *except the bit at index* $h_{i,m}(x)$

Definition 2.6. *Let* m, k *be positive integers. We define an* (m,k)*-SCBF to be any* (n, ε)*-BF with algorithms defined in Figure 2, with* $\Phi = (m,k)$.

Standard LBF. The Standard Learned Bloom Filter (SLBF) uses a Learning Model as a pre-filter in front of a Classical Bloom Filter. The CBF is called the Backup CBF as it is only queried on inputs x for which the LM decides that x is not an element of the stored set ($x \notin S_\Pi$). This maintains the completeness property of the BF (Definition 2.2) ensuring that it never outputs false negatives. The public parameters of LBFs typically include a memory budget m_b in addition to parameters for the Backup CBF, m, k. m_b is an upper bound on the memory the LM can use. Figure 1 illustrates an SLBF.

Definition 2.7. *Let* m_b, m, k *be positive integers. We define an* (m_b, m, k)*-SLBF to be any* $(n, \varepsilon, \varepsilon_p, \varepsilon_n)$*-LBF with memory budget* m_b *and algorithms defined in Figure 3, with* $\Phi = (m,k)$.

$C_r(\Phi, S_\Pi)$	$Q(x, \sigma)$
$m_b, m, k \leftarrow \Phi$;	$\mathcal{L}, \sigma_c \leftarrow \sigma$
$\mathcal{L} \leftarrow T_r(G_r(S_\Pi), m_b)$	$l \leftarrow \mathcal{L}(x)$
$S'_\Pi \leftarrow \{x \in S_\Pi : \mathcal{L}(x) = \perp\}$	$b \leftarrow BCBF.Q(x, \sigma_c)$
$\sigma_c \leftarrow BCBF.C_r(S'_\Pi)$	**return** $[l \vee b]$
return $\sigma = (\mathcal{L}, \sigma_c)$	

Figure 3: AMQ-PDS syntax instantiation for the SLBF. G_r is any algorithm that generates a training dataset (Section 2.3) from S_Π. T_r is any algorithm that trains a Learning Model from a training dataset and a memory budget. BCBF is the Backup Classical Bloom Filter.

2.5 Randomized Response

Warner's randomized response is one of the most common private set membership techniques, first proposed in 1965 (Warner, 1965). Scientists have used it to survey set membership among a population for things that individual members wish to retain confidentiality about. A commonly used example is "Are you a member of the Communist Party?" (Hox and Lensvelt-Mulders, 2008). Other examples include surveying the number of abortion recipients (Abernathy et al., 1970) and surveys regarding sexual orientation (Chen et al., 2014). In Warner's randomized response, the respondent answers a Yes/No question truthfully with probability p. With probability $1-p$, the respondent flips a fair coin. The respondent answers *Yes* if the coin is heads, and *No* otherwise. Thanks to this technique each respondent has *plausible deniability* regarding their membership. The following is a well-known differential privacy result (Dwork and Roth, 2014) for the differential privacy of Warner's randomized response. This result will be used later in the paper to prove the correctness of one of our set release algorithms.

Theorem 2.1 (Randomized Response)**.** *Let X be a binary random variable. Define the generalized randomized response mechanism $\mathcal{M}_r$ as follows:*

$$P[x \in \mathcal{M}_r(\{x\})] = p, \quad P[x \notin \mathcal{M}_r(\{x\})] = 1-p.$$

for probability p. Then, $\mathcal{M}_r$ satisfies $\left(\ln\left(\frac{p}{1-p}\right), 0\right)$-differential privacy.

Mangat (Mangat, 1994) proposed a variant of Warner's randomized response. In Mangat's randomized response, a respondent answers truthfully to a Yes/No question with probability p. With probability $1-p$, the respondent always answers *Yes*. Unlike Warner's randomized response, Mangat's variant only introduces one-sided error into the dataset.

3 PRIVACY MODEL

We formulate an adaption of differential privacy (Dwork and Roth, 2014) and asymmetric differential privacy (Takagi et al., 2022) for unordered sets. For any two sets, we introduce the notion of the Simple-Jaccard distance[1] to measure set similarity.

Definition 3.1 (Simple-Jaccard)**.** *For any two sets A, B: $d_{sj}(A,B) = |A \cup B| - |A \cap B|$*

We define (ε, δ)-differential privacy for sets and BFs.

Definition 3.2 (Set Privacy)**.** *A randomized algorithm $\mathcal{A}_r$ satisfies (ε, δ)-differential privacy if for any two sets S, S' s.t $d_{sj}(S, S') \leq 1$ and for any possible output range $O \subseteq Range(\mathcal{A}_r)$,*

$$P[\mathcal{A}_r(S) \in O] \leq e^{\varepsilon} P[A_r(S' \in O] + \delta$$

where the probabilities are over the coins of $\mathcal{A}_r$.

Definition 3.3 (Asymmetric Set Privacy)**.** *A randomized algorithm $\mathcal{A}_r$ satisfies $(\varepsilon_1, \varepsilon_2, \delta)$-asymmetric differential privacy if for any two sets S, S' such that $d_{sj}(S, S') \leq 1$, and for any possible output range $O \subseteq Range(\mathcal{A}_r)$, the following two properties hold:*

1. *If $S' = S \setminus \{x\}$ for some $x \in S$, then*

$$\Pr[\mathcal{A}_r(S) \in O] \leq e^{\varepsilon_1} \Pr[\mathcal{A}_r(S') \in O] + \delta$$

2. *If $S' = S \cup \{x\}$ for some $x \notin S$, then*

$$\Pr[\mathcal{A}_r(S) \in O] \leq e^{\varepsilon_2} \Pr[\mathcal{A}_r(S') \in O] + \delta$$

where probabilities are over the coins of $\mathcal{A}_r$.

Definition 3.4 (Bloom Filter Privacy)**.** *An AMQ-PDS $\Pi = (C_r, Q)$ is an $(n, \varepsilon, \tilde{\varepsilon}, \tilde{\delta})$-P̃BF if Π is an (n, ε)-BF and if for all $S_\Pi, S'_\Pi \subseteq \mathfrak{D}$ such that $d_{sj}(S_\Pi, S'_\Pi) \leq 1$,*

$$\forall \sigma \in \Sigma : P[C_r(\Phi, S_\Pi) = \sigma] \leq e^{\tilde{\varepsilon}} P[C_r(\Phi, S'_\Pi) = \sigma] + \tilde{\delta}$$

where the probabilities are over the coins of C_r.

An asymmetric private Bloom Filter, $(n, \varepsilon, \tilde{\varepsilon}_1, \tilde{\varepsilon}_2, \tilde{\delta})$-aP̃BF can be defined the same way. We also use identical definitions for a Private Learned Bloom Filter, $(n, \varepsilon, \varepsilon_p, \varepsilon_n, \tilde{\varepsilon}, \tilde{\delta})$-P̃LBF, Private AMQ-PDS, $(n', \varepsilon'_p, \varepsilon'_n, \tilde{\varepsilon}, \tilde{\delta})$-P̃AMQ-PDS and so on.

3.1 Set Release Games

We introduce two games, which will prove useful later on in establishing bounds on BF privacy. The games enforce no bounds on the adversary's computational power or auxiliary information.

Game 3.1 (STRICT-SET-RELEASE)**.** *We have a dealer Υ, a player P, and an adversary A. Sets in the game are subsets of a suitable universe U.*

[1] An unweighted version of the Jaccard distance

Dime(S,U,ε)

$\tilde{S} \leftarrow S$; $q \leftarrow \frac{1}{1+e^{\varepsilon}}$

for each $x \in U \setminus S$: **if** $(¢)_q = (h)$ **then** $\tilde{S} \leftarrow S \cup \{x\}$

for each $x \in S$: **if** $(¢)_q = (h)$ **then** $\tilde{S} \leftarrow S \setminus \{x\}$

return $\tilde{S}$

Figure 4: Algorithm for SET-RELEASE (Game 3.2)

Round 1 *A gives* Υ *any two sets* S_0 *and* S_1 *s.t* $d_{sj}(S_0,S_1) \leq 1$.

Round 2 Υ *flips a bit* $b \leftarrow\$ \{0,1\}$ *and gives* S_b *to P. P can add elements to S but not remove elements. P returns a modified set* $\tilde{S}$.

Round 3 Υ *gives A the set* $\tilde{S}$. *A returns bit* b'. *If* $b = b'$, *A wins. Otherwise, P wins.*

We also define a more lenient Set Release game, which also allows element deletion.

Game 3.2 (SET-RELEASE)**.** *Identical to* STRICT-SET-RELEASE *(Game 3.1) except that in Round 2, player P can add elements to set S as well as remove elements from S.*

It follows from Definition 3.2 that any algorithm that bounds the winning probability of an adversary in a Set Release Game satisfies (ε,δ)-differential privacy in the same setting for suitable ε,δ values.

3.2 Nickel & Dime Algorithms

Figure 4 introduces Dime, an algorithm for player P in SET-RELEASE (Game 3.2) that satisfies the notion of $(\varepsilon,0)$-differential privacy. $\tilde{S}$ is initialized to S. For each element x in the universe U that is not in S, we flip a biased coin with probability $\frac{1}{1+e^{\varepsilon}}$. If the coin flips heads, we add x to $\tilde{S}$. For each element x in S, we flip another biased coin with the same probability. If the coin flips heads, we remove x from $\tilde{S}$.

Theorem 3.1. *Dime satisfies* $(\varepsilon,0)$*-differential privacy in the* SET-RELEASE *Game setting.*

Proof. The Dime algorithm is a special case of the randomized response mechanism from Theorem 2.1 applied to set membership. Let S_0 and S_1 be two sets with Simple-Jaccard distance $ds_j(S_0,S_1) \leq 1$. The Dime algorithm modifies S by either 1) adding each element $x \notin S$ with probability $p = \frac{1}{1+e^{-\varepsilon}}$, or 2) removing each element $x \in S$ with the same probability p. This strictly follows the structure of the randomized response mechanism from Theorem 2.1, where:

$$\Pr[x \in \tilde{S} \mid x \in S] = p = 1 - q = \frac{e^{\varepsilon}}{1+e^{\varepsilon}}$$

Nickel(S,U,ε)

$\tilde{S} \leftarrow S$; $q \leftarrow e^{\varepsilon}$

for each $x \in U \setminus S$: **if** $(¢)_q = (h)$ **then** $\tilde{S} \leftarrow S \cup \{x\}$

return $\tilde{S}$

Figure 5: Algorithm for STRICT-SET-RELEASE (Game 3.1)

$$\Pr[x \in \tilde{S} \mid x \notin S] = 1 - p = q = \frac{1}{1+e^{\varepsilon}}.$$

We can use this to derive $\ln(\frac{p}{1-p}) = \ln(e^{\varepsilon}) = \varepsilon$. By Theorem 2.1, this satisfies $(\varepsilon,0)$-differential privacy, ensuring that

$$\frac{\Pr[\text{Dime}(S_0,U,\varepsilon) = \tilde{S}]}{\Pr[\text{Dime}(S_1,U,\varepsilon) = \tilde{S}]} \leq e^{\varepsilon}.$$

□

Figure 5 introduces Nickel, an algorithm for player P in STRICT-SET-RELEASE (Game 3.1) that satisfies the notion of $(\varepsilon_1,\varepsilon_2,0)$-asymmetric differential privacy. $\tilde{S}$ is initialized to S. For each element x in the universe U that is not in S, we flip a biased coin with probability $\frac{1}{e^{\varepsilon}}$. If the coin flips heads, we add x to $\tilde{S}$. To prove Nickel satisfies asymmetric differential privacy, we first prove a result regarding Mangat's randomized response (§ 2.5).

Theorem 3.2. *Mangat's randomized response satisfies* $(\ln(\frac{1}{1-p}),\ln(1-p),0)$*-asymmetric differential privacy.*

Proof. Let S,S' be two sets s.t $d_{sj}(S,S') = 1$. Let $\mathcal{M}_r$ be Mangat's randomized response. The probabilities are taken over the random coins of $\mathcal{M}_r$. First, consider the case where $S' \subset S$.

Case 1: $S' = S \setminus \{x\}$ for some $x \in S$.

$$\frac{\Pr[\mathcal{M}(S) \in O]}{\Pr[\mathcal{M}(S') \in O]} = \frac{\Pr[x \in \mathcal{M}(S)]}{\Pr[x \in \mathcal{M}(S')]} = \frac{1}{1-p}$$

Therefore $\varepsilon_1 = \ln(\frac{1}{1-p})$. Next, consider the case where $S \subset S'$.

Case 2: $S' = S \cup \{x\}$ for some $x \notin S$.

$$\frac{\Pr[\mathcal{M}(S) \in O]}{\Pr[\mathcal{M}(S') \in O]} = \frac{\Pr[x \in \mathcal{M}(S)]}{\Pr[x \in \mathcal{M}(S')]} = 1 - p$$

□

Theorem 3.3. *Nickel satisfies asymmetric differential privacy in the* STRICT-SET-RELEASE *Game setting.*

Proof. The Nickel algorithm is a special case of Mangat's randomized response mechanism applied to set membership. Let S_0 and S_1 be two sets s.t

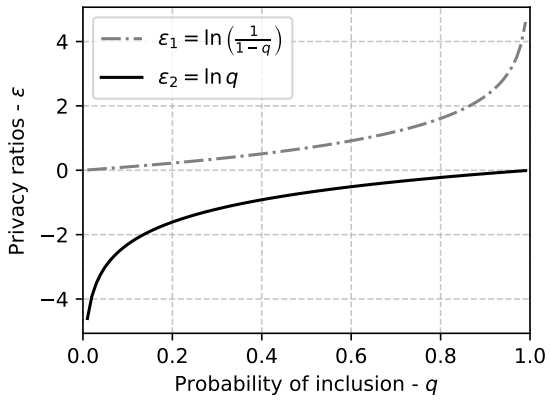

Figure 6: The asymmetric privacy bounds of the Nickel algorithm as q varies. ε_1 is the privacy of an element not in the output set, while ε_2 is the privacy of an element in the output set.

$d_{sj}(S_0, S_1) \leq 1$. The Nickel algorithm modifies input set S by adding each element $x \notin S$ with probability $q = e^{\varepsilon}$. This follows the structure of Mangat's randomized response mechanism from Theorem 3.2, where $p = 1 - q = 1 - e^{\varepsilon}$. We can use this to derive $\varepsilon_1 = \ln(\frac{1}{1-p}) = \ln(\frac{1-e^{\varepsilon}}{e^{\varepsilon}})$ and $\varepsilon_2 = \ln e^{\varepsilon} = \varepsilon$. By Theorem 2.1, this satisfies $(\ln(\frac{1-e^{\varepsilon}}{e^{\varepsilon}}), \varepsilon, 0)$-asymmetric differential privacy. □

Note that since $q \in [0,1]$ in the Nickel algorithm, ε has an upper bound of 0 i.e $\varepsilon \leq 0$. Theorem 3.3 proves that Nickel satisfies *asymmetric* differential privacy by adding elements with a controlled probability q. This ensures that an observer cannot reliably distinguish between an element in the original set and an element added in the output set probabilistically by Nickel. Asymmetric differential privacy allows us to explicitly model scenarios where A's ability to infer the *presence* of an element in the original set may be much weaker than A's ability to infer the *absence* of an element in the original set. We illustrate this in Figure 6. ε_1 and ε_2 models A's ability to infer the absence and presence of an element in the original set respectively. Since Nickel never removes an element in the original set, ε_1 does not meaningfully constrain the A's ability to infer absence. ε_2, however, provides meaningful privacy for presence. The privacy increases as $\varepsilon_2 \to 0$ (i.e. $q \to 1$). Intuitively, when all elements in the universe appear in the output set, A has little probability of distinguishing which elements were in the original set. The privacy decreases as $\varepsilon_2 \to -\infty$ (i.e. $q \to 0$) i.e as fewer elements are probabilistically added by Nickel. This asymmetry aligns well with many known set membership scenarios where presence in a set is sensitive, while absence is not. For example, knowing that an individual belongs to the set of Communist party members, HIV patients, or recipients of abortions can be highly sensitive, whereas knowing that an individual is not in these sets often does not reveal sensitive information. There are also situations where this type of privacy guarantee is *necessary*. For example, in epidemic analysis, when creating a set of the number of infected individuals that visited a location (Takagi et al., 2022), having a two-sided error is not useful.

3.3 Privacy Theorems

We now show how we can create Private Bloom Filters from algorithms that satisfy privacy guarantees in the setting of Set Release Games. The following two theorems hold for any BF including CBFs and LBFs. For ease of exposition, we only use BF notation. However, identical proofs work in LBF notation. Just like in the context of Set Release Games, we assume the adversary is computationally unbounded and has unbounded auxiliary information. We also assume that the adversary can invoke the BF's algorithms (C_r, Q), and access the BF's internal state (σ).

Theorem 3.4. *Let $\Pi = (C_r, Q)$ be any (n, ε)-BF. If algorithm $\mathcal{A}_r$ satisfies $(\tilde{\varepsilon}_1, \tilde{\varepsilon}_2, \tilde{\delta})$-asymmetric differential privacy in the setting of the* STRICT-SET-RELEASE *Game, then $\tilde{\Pi} = (\tilde{C}_r, Q)$ is an $(n', \varepsilon', \tilde{\varepsilon}_1, \tilde{\varepsilon}_2, \tilde{\delta})$-aP̃BF, for some $n' \geq n, \varepsilon' \geq \varepsilon$ where $\tilde{C}_r(\Phi, S_\Pi) = C_r(\Phi, \mathcal{A}_r(S_\Pi))$.*

Proof. $\tilde{\Pi}$ is an (n', ε')-BF because the player in a STRICT-SET-RELEASE game is only allowed to add elements, which does not invalidate the completeness and soundness properties of the Bloom Filter. We can prove our privacy bound by contradiction. Assume $\tilde{\Pi}$ is not an $(n, \varepsilon, \tilde{\varepsilon}_1, \tilde{\varepsilon}_2, \tilde{\delta})$-aP̃BF. This means there exist S_Π, S'_Π with $d_{sj}(S_\Pi, S'_\Pi) \leq 1$ and some state σ for which either $S'_\Pi = S_\Pi \setminus \{x\}$ for some $x \in S_\Pi$ and

$$P[C_r(\Phi, \mathcal{A}_r(S_\Pi)) = \sigma] > e^{\tilde{\varepsilon}_1} P[C_r(\Phi, \mathcal{A}_r(S'_\Pi)) = \sigma] + \tilde{\delta}$$

or $S'_\Pi = S_\Pi \cup \{x\}$ for some $x \notin S_\Pi$ and

$$P[C_r(\Phi, \mathcal{A}_r(S_\Pi)) = \sigma] > e^{\tilde{\varepsilon}_2} P[C_r(\Phi, \mathcal{A}_r(S'_\Pi)) = \sigma] + \tilde{\delta}$$

However, this allows an adversary to win the STRICT-SET-RELEASE game with probability larger than our asymmetric differential privacy bounds and contradicts the assumption that $\mathcal{A}_r$ satisfies $(\tilde{\varepsilon}_1, \tilde{\varepsilon}_2, \tilde{\delta})$-asymmetric differential privacy in our setting. □

Theorem 3.5. *Let $\Pi = (C_r, Q)$ be any (n, ε)-BF. If algorithm $\mathcal{A}_r$ satisfies $(\tilde{\varepsilon}, \tilde{\delta})$-differential privacy in the setting of the* SET-RELEASE *Game, then $\tilde{\Pi} = (\tilde{C}_r, Q)$ is an $(n', \varepsilon'_p, \varepsilon'_n, \tilde{\varepsilon}, \tilde{\delta})$-P̃AMQ-PDS, for some $n', \varepsilon'_p, \varepsilon'_n$ where $\tilde{C}_r(\Phi, S_\Pi) = C_r(\Phi, \mathcal{A}_r(S_\Pi))$.*

Proof. We can prove our privacy bound by contradiction identical to Thm. 3.4. The difference here is that the SET-RELEASE game allows removing elements from the input set. Therefore the completeness property of the BF is no longer guaranteed. However, $\tilde{\Pi}$ still satisfies the positive and negative soundness properties of an $(n', \varepsilon'_p, \varepsilon'_n)$-AMQ-PDS for suitable values of n', ε'_p, and ε'_n. □

4 DISCUSSION

4.1 Performance Analysis

We investigate how our method for adding privacy affects the performance of a given BF in terms of its False Positive Rate (FPR) and its False Negative Rate (FNR). We do not classify queries to elements added by our privacy-preserving algorithms as false positives i.e. a query on $x \in \tilde{S} \setminus S$ is not a false positive. These elements are not representative of typical false positives, which arise naturally due to the probabilistic nature of the BF. As such, we exclude these elements from the FPR calculations to focus on the inherent accuracy of the BF under privacy-preserving conditions. This distinction ensures a clear separation between errors due to the BF's one-sided guarantees and those intentionally introduced for privacy.

For any BF that stores set S_Π from a suitable universe U, has public parameters Φ, and internal state σ: let $FPR(S_\Pi, \Phi, \sigma)$ and $FNR(S_\Pi, \Phi, \sigma)$ be functions that return the expected FPR and expected FNR of the BF respectively. Then the FPR and FNR of a Private BF on the same set constructed using the Nickel algorithm will be $FPR(Nickel(S_\Pi, U, \varepsilon), \Phi, \sigma)$ and $FNR(Nickel(S_\Pi, U, \varepsilon), \Phi, \sigma)$ respectively. Similar expressions hold for the Dime algorithm.

We demonstrate a simple instance of our analysis, taking the Standard Classical Bloom Filter (SCBF) as a case study. An (m,k)-SCBF has zero FNR and its FPR is approximately given by (Broder and Mitzenmacher, 2003),

$$\text{FPR}(S_\Pi, (m,k)) = (1 - e^{-k \cdot |S_\Pi|/m})^k$$

For a given set input S, the expected cardinality of the set output by the Nickel algorithm and the Dime algorithm is

$$|Nickel(S_\Pi, U, \varepsilon)| = |S_\Pi| + e^\varepsilon(|U| - |S_\Pi|)$$

$$|Dime(S_\Pi, U, \varepsilon)| = |S_\Pi| + \frac{1}{1+e^\varepsilon}(|U| - |S_\Pi|) - \frac{1}{1+e^\varepsilon}|S_\Pi|$$

We can replace $|S_\Pi|$ in the FPR equation for the SCBF with these expressions to get FPR expressions for Private SCBFs. When using the Dime algorithm, we will also have a non-zero FNR which will simply be the probability that a given $x \in S_\Pi$ was removed from the set S_Π by the algorithm i.e $\frac{1}{1+e^\varepsilon}$.

4.2 Illustrative Example

We illustrate our privacy-preserving techniques in a popular randomized response setting. Assume there are 50 citizens in a repressive regime and we wish to release a contact list of 10 citizens in the form of a queryable Bloom Filter. These citizens have volunteered as people who minority citizens can contact for support. However, each citizen in the contact list can be incarcerated if found out, so they want to maintain plausible deniability. Let the privacy parameter we need to maintain plausible deniability be $\varepsilon = -3$. They can use the Nickel algorithm to store an expected $40e^{-3} \approx 2$ more members in the Bloom Filter who are uniformly randomly chosen from the remaining population to maintain plausible deniability for the members in the original set.

5 CONCLUSIONS

In this paper, we address the privacy of AMQ-PDS, focusing on the Classical and Learned Bloom Filter. We provide formal guarantees based on differential privacy notions that protect the privacy of the set stored by the Bloom Filter against adversarial inference. Our work is the first to conduct a privacy analysis for the Learned Bloom Filter and provide provably private constructions for the Learned Bloom Filter. We complemented our theoretical contributions with a performance analysis discussing the trade-offs between privacy and membership query error rates. This research lays the foundation for privacy-preserving uses of probabilistic data structures in diverse domains. We leave the problem of extending our private constructions or designing new ones for other PDS such as the Count-Min Sketch (Cormode and Muthukrishnan, 2005) to future work.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for helpful feedback and corrections. We thank Dr. Allison Bishop for helpful insights from her graduate course in Data Privacy at the City College of New York.

REFERENCES

Abdennebi, A. and Kaya, K. (2021). A bloom filter survey: Variants for different domain applications.

Abernathy, J. R., Greenberg, B. G., and Horvitz, D. G. (1970). Estimates of induced abortion in urban north carolina. *Demography*, 7(1):19–29.

Bianchi, G., Bracciale, L., and Loreti, P. (2012). "better than nothing" privacy with bloom filters: To what extent? In Domingo-Ferrer, J. and Tinnirello, I., editors, *Privacy in Statistical Databases*, pages 348–363, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bishop, A. and Tirmazi, H. (2024). Adversary resilient learned bloom filters. Cryptology ePrint Archive, Paper 2024/754.

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426.

Bose, P., Guo, H., Kranakis, E., Maheshwari, A., Morin, P., Morrison, J., Smid, M., and Tang, Y. (2008). On the false-positive rate of bloom filters. *Information processing letters*, 108(4):210–213.

Broder, A. and Mitzenmacher, M. (2003). Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1(4):485 – 509.

Chen, X., Du, Q., Jin, Z., Xu, T., Shi, J., and Gao, G. (2014). The randomized response technique application in the survey of homosexual commercial sex among men in beijing. *Iranian journal of public health*, 43(4):416–422.

Clayton, D., Patton, C., and Shrimpton, T. (2019). Probabilistic data structures in adversarial environments. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1317–1334, New York, NY, USA. Association for Computing Machinery.

Cormode, G. and Muthukrishnan, S. (2005). An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75.

Dwork, C. and Roth, A. (2014). The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407.

Filić, M., Kocher, K., Kummer, E., and Unnikrishnan, A. (2024). Deletions and dishonesty: Probabilistic data structures in adversarial settings. In *Asiacrypt '24*.

Filić, M., Paterson, K., Unnikrishnan, A., and Virdia, F. (2022). Adversarial Correctness and Privacy for Probabilistic Data Structures. In *CCS '22*, pages 1037–1050.

Galan, S., Reviriego, P., Walzer, S., Sanchez-Macian, A., Liu, S., and Lombardi, F. (2023). On the Privacy of Counting Bloom Filters Under a Black-Box Attacker . *IEEE Transactions on Dependable and Secure Computing*, 20(05):4434–4440.

Gerbet, T., Kumar, A., and Lauradoux, C. (2015). The power of evil choices in bloom filters. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 101–112.

Gupta, D. and Batra, S. (2017). A short survey on bloom filter and its variants. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 1086–1092.

Hox, J. and Lensvelt-Mulders, G. (2008). Encyclopedia of survey research methods. Randomized Response.

Kraska, T., Beutel, A., Chi, E. H., Dean, J., and Polyzotis, N. (2018). The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 489–504, New York, NY, USA. Association for Computing Machinery.

Mangat, N. S. (1994). An improved randomized response strategy. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(1):93–95.

Mitzenmacher, M. (2018a). A model for learned bloom filters, and optimizing by sandwiching. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 462–471, Red Hook, NY, USA. Curran Associates Inc.

Mitzenmacher, M. (2018b). A model for learned bloom filters and related structures.

Naor, M. and Eylon, Y. (2019). Bloom filters in adversarial environments. *ACM Trans. Algorithms*, 15(3).

Naor, M. and Oved, N. (2022). Bet-or-pass: Adversarially robust bloom filters. In Kiltz, E. and Vaikuntanathan, V., editors, *Theory of Cryptography*, pages 777–808, Cham. Springer Nature Switzerland.

Reviriego, P., Alberto Hernández, J., Dai, Z., and Shrivastava, A. (2021). Learned bloom filters in adversarial environments: A malicious url detection use-case. In *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*, pages 1–6.

Reviriego, P., Sánchez-Macian, A., Walzer, S., Merino-Gómez, E., Liu, S., and Lombardi, F. (2023). On the privacy of counting bloom filters. *IEEE Trans. Dependable Secur. Comput.*, 20(2):1488–1499.

Sengupta, N., Bagchi, A., Bedathur, S., and Ramanath, M. (2018). Sampling and reconstruction using bloom filters. *IEEE Transactions on Knowledge and Data Engineering*, 30(7):1324–1337.

Sweeney, L. (2002). k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570.

Takagi, S., Kato, F., Cao, Y., and Yoshikawa, M. (2022). Asymmetric differential privacy. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 1576–1581. IEEE.

Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American statistical association*, 60(309):63–69.