NAVIGATING THE PRIVACY-ACCURACY TRADE-OFF FOR LEARNED BLOOM FILTER

Anonymous authors
Paper under double-blind review

ABSTRACT

With the rapid development of membership query applications, there is an increasing demand for efficient queries under rigorous differential privacy guarantees. In this work, we propose a novel data structure, Differentially Private Learned Bloom Filter (DP-LBF), which enables efficient membership queries while ensuring differential privacy. DP-LBF samples the model's routing threshold with a sensitivity-one exponential mechanism whose utility counts the sensitive members accepted directly by the learned model. An adaptive differentially private Bloom filter is employed to store low-confidence elements, maintaining high space efficiency and query performance. Theoretical analysis shows that DP-LBF provides strong privacy guarantees and practical query accuracy under a given privacy budget ϵ. Experiments on synthetic and real-world datasets show that DP-LBF achieves substantially lower RMSE than the tested private membership-query baselines while retaining a compact released backup representation.

1 INTRODUCTION

Membership queries are core operations in many data-intensive applications, such as database indexing (Wang et al., 2022; Ji et al., 2025), network caching (Wang et al., 2021; 2024), and duplicate detection (Ha et al., 2024; Dai et al., 2023). The classic Bloom filter (BF) is widely used for its efficient storage and query performance (Bloom, 1970; Liang et al., 2023; Pandey et al., 2024). However, when dealing with some sensitive data, it is unable to provide privacy protection for such data. In privacy scenarios, how to ensure data accuracy while maintaining privacy requirements has become a critical issue that needs to be addressed (Hassan et al., 2019; Zhu et al., 2020).

In recent years, Kraska et al. (2018) first proposed the Learned Bloom Filter (LBF), which effectively integrates learned models such as neural networks (Borisov et al., 2022; Han et al., 2021), decision trees (Nanfack et al., 2022; Costa & Pedreira, 2023), and gradient boosting trees (Iosipoi & Vakhrushev, 2022; Brophy et al., 2023) into the Bloom Filter. The LBF generalizes the traditional Bloom Filter and can model the data distribution, significantly reducing storage costs and improving data accuracy (Liu et al., 2020; Yang et al., 2022).

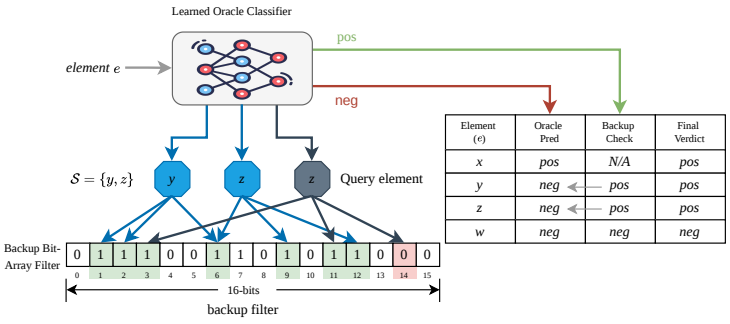


Figure 1: Illustration of a Learned Bloom Filter with elements $\mathcal{S} = \{x, y, z\}$ and a query w, showing the classification stage followed by the backup bit-array for handling false negatives.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Consider a toy element set $\mathcal{S} = \{x, y, z\}$ for constructing a Learned Bloom Filter (LBF), as illustrated in Figure 1. The classifier correctly identifies x as a member of $\mathcal{S}$, requiring no further action. Elements y and z, misclassified as non-members, are inserted into a standard Bloom filter with 16 bits and 3 hash functions. When querying the LBF with an element w, if the classifier predicts w as a non-member, the query is passed to the backup Bloom filter, which in this case confirms that $w \notin \mathcal{S}$.

Despite its advantages, LBF directly relies on the model's prediction scores and backup bloom filter, which may reveal sensitive member information (Vos et al., 2024a). Without privacy protection, attackers could observe the outputs and infer whether specific elements belong to the set, potentially leading to data privacy breaches (Li & Zhang, 2021). This motivates the need for a privacy-preserving variant of LBF.

Our Proposal. To address the privacy risks of learned membership data structures, we propose the Differentially Private Learned Bloom Filter (DP-LBF). DP-LBF uses three private stages to release a routing threshold, a noisy backup capacity, and a randomized backup Bloom filter. The learned model, candidate thresholds, public calibration data, and all hyperparameters are fixed before the sensitive set is accessed. To summarize, this paper makes the following contributions:

- We introduce DP-LBF, a framework with three private stages for membership queries. It combines a fixed public learned model with a privately released backup structure, without requiring the querier to know the sensitive element partition in advance.
- We design three coordinated private mechanisms. PTS selects the routing threshold using a sensitivity-one exponential-mechanism utility with a public FPR penalty; NBSR releases a Laplace-perturbed backup capacity; and ABFDP constructs the filter from that noisy capacity and applies bitwise randomized response.
- We prove that the released state satisfies ϵ-differential privacy under add/remove adjacency, where $\epsilon = \epsilon_\tau + \epsilon_m + \epsilon_b$. We also give an approximate utility characterization under standard Bloom-filter assumptions and derive bounds for space and query complexity.
- We evaluate DP-LBF against representative private membership baselines on synthetic and real-world datasets, reporting RMSE, query time, and memory consumption across multiple privacy budgets.

Roadmap. The rest of this paper is organized as follows. Section 2 reviews closely related private membership mechanisms. Section 3 introduces some preliminaries and formulates the problem of devising a differentially private learned Bloom filter. We present our DP-LBF in Section 4. Section 5 discusses the theoretical analysis for our DP-LBF. We report the experimental results in Section 6 and conclude in Section 7.

2 Related Work

Membership queries are a fundamental problem in data management and security, widely used in scenarios such as network security, database indexing, privacy-preserving data analytics, and large-scale telemetry systems (Paya et al., 2024; Bawa et al., 2009; Xue et al., 2020; Sun et al., 2023; Zaman et al., 2024). Their goal is to determine whether a given query element belongs to a key or a no-key set (Qin et al., 2023; Chen et al., 2024; Filic et al., 2022). Classical solutions such as Bloom filters enable compact and efficient probabilistic membership testing, yet they expose information about the underlying set and therefore raise substantial privacy concerns when applied in adversarial or sensitive environments (Vos et al., 2024b; Clayton et al., 2019).

RAPPOR (Erlingsson et al., 2014) employs a two-stage randomized response mechanism for local differential privacy, targeting frequency estimation and data collection tasks. When applied to membership queries, the mechanism perturbs each Bloom filter bit independently per query. Alaggan et al. (2012) proposed BLIP, which publishes a reusable Bloom-filter representation after independently flipping its bits according to the privacy budget and number of hash functions. Patel et al. (2024) proposed DPSet, which represents a private set through randomized retention and linear constraints over a finite field rather than perturbing a pre-existing non-private representation. Zhang et al. (2023) proposed EBF-DP for range queries under differential privacy in crowdsourcing. Their scheme combines Prefix Encoding (PE) with a Bloom filter and applies Basic Randomized Response (BRR) to perturb query results, enabling privacy-preserving range membership testing. Zaman et al. (2024) proposed a differentially private Bloom filter encoding scheme for sequential numerical data. The method adds Laplace noise to the Bloom filter bit array followed by a thresholding mechanism,

targeting set-based similarity matching tasks such as KNN and SVM. Ke et al. (2025) proposed DPBloomFilter, which applies randomized response to a classic shared m-bit Bloom-filter array. It computes the $(1-\delta)$-quantile $N = F_W^{-1}(1-\delta)$ of the number W of bits changed between neighboring filters, assigns the per-bit parameter $\epsilon_0 = \epsilon/N$, and flips every bit with probability $1/(e^{\epsilon_0}+1)$, yielding an (ϵ, δ)-DP mechanism. Tirmazi (2025) introduced the Mangat and Warner mechanisms for private Bloom filters and extended them to learned variants. The Warner construction applies randomized response to membership indicators over an explicitly specified finite universe.

In summary, these existing methods either target different application scenarios (e.g., frequency estimation, range queries, similarity matching), directly randomize membership over an explicitly enumerated universe, or rely on fixed perturbation mechanisms without confidence-based routing. DP-LBF instead combines a public learned score with private threshold selection and a randomized backup filter for the curator-held set.

3 PRELIMINARIES

This section first overviews the definition and related theorem of differential privacy, then introduces the definition of Learned Bloom Filter (LBF), and finally presents the problem statement. For clarity, all notations used in this paper are summarized in Table 1 in the Appendix.

3.1 DIFFERENTIAL PRIVACY

Differential Privacy (DP) (Dwork, 2006; Abadi et al., 2016) provides a rigorous formalization of privacy by ensuring that the outcome of a mechanism remains nearly unchanged when any single record in the dataset is modified. In other words, DP enforces stability with respect to individual data contributions. One of its notable advantages is its generality: any randomized algorithm that satisfies this stability constraint can be considered differentially private (Ilvento, 2020).

Definition 1 (ϵ-differential privacy under add/remove adjacency (Dwork, 2006; Abadi et al., 2016))**.** *Two datasets $\mathcal{D}$ and $\mathcal{D}'$ are neighboring, denoted $\mathcal{D} \simeq \mathcal{D}'$, if one is obtained from the other by adding or removing one record. A randomized mechanism $\mathcal{M}$ satisfies ϵ-differential privacy if, for every neighboring pair and every measurable event $O \subseteq \mathrm{Range}(\mathcal{M})$,*

$$\Pr[\mathcal{M}(\mathcal{D}) \in O] \leq e^{\epsilon} \Pr[\mathcal{M}(\mathcal{D}') \in O].$$

Here, ϵ is the privacy budget; smaller values provide stronger privacy. This add/remove convention is used throughout the paper (Dwork et al., 2006; Dwork, 2006; Dwork et al., 2010).

Definition 2 (Exponential Mechanism (McSherry & Talwar, 2007))**.** *Let $\mathcal{D}$ be a dataset, and let $\mathcal{R}$ be a set of possible outputs. Let $u : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$ be a utility function that assigns a score to each output $r \in \mathcal{R}$ for dataset $\mathcal{D}$. The **Exponential Mechanism** $\mathcal{M}$ selects an output $r \in \mathcal{R}$ according to the probability distribution*

$$\Pr[\mathcal{M}(\mathcal{D}) = r] = \frac{\exp\left(\frac{\epsilon \cdot u(\mathcal{D}, r)}{2\Delta u}\right)}{\sum_{r' \in \mathcal{R}} \exp\left(\frac{\epsilon \cdot u(\mathcal{D}, r')}{2\Delta u}\right)},$$

where $\epsilon > 0$ is the privacy budget, and Δu is the sensitivity of u. Since for any pair of neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$ and any output $r \in \mathcal{R}$, the inequality $\frac{\Pr[\mathcal{M}(\mathcal{D})=r]}{\Pr[\mathcal{M}(\mathcal{D}')=r]} \leq e^{\epsilon}$ holds, the mechanism satisfies ϵ-differential privacy.

Theorem 1 (Post-Processing of Differential Privacy (Dwork et al., 2006))**.** *Let $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ be an ϵ-differentially private mechanism, and let $f : \mathcal{R} \to \mathcal{R}'$ be any (possibly randomized) function. Then the composition $f \circ \mathcal{M} : \mathcal{D} \to \mathcal{R}'$, defined by $(f \circ \mathcal{M})(\mathcal{D}) = f(\mathcal{M}(\mathcal{D}))$, also satisfies ϵ-differential privacy.*

Property 1 (Sequential Composition (Dwork et al., 2006))**.** *Let $\mathcal{M}_1, \ldots, \mathcal{M}_n$ be a sequence of possibly adaptive mechanisms. For each $i \in \{1, \ldots, n\}$ and every fixed history $r_{<i} = (r_1, \ldots, r_{i-1})$, suppose the conditional mechanism*

$$\mathcal{M}_i(\cdot\,; r_{<i}) : \mathcal{D} \to \mathcal{R}_i$$

satisfies ϵ_i-differential privacy. Define the released transcript $\mathcal{M}(D) = (R_1, \ldots, R_n)$ recursively by

$$R_1 = \mathcal{M}_1(D), \qquad R_i = \mathcal{M}_i(D; R_1, \ldots, R_{i-1}), \quad i \geq 2.$$

Then $\mathcal{M} : \mathcal{D} \to \mathcal{R}_1 \times \cdots \times \mathcal{R}_n$ satisfies $(\sum_{i=1}^{n} \epsilon_i)$-differential privacy.

3.2 LEARNED BLOOM FILTER

The Learned Bloom Filter (LBF) (Kraska et al., 2018) combines a learned model with a backup Bloom Filter to enhance query accuracy and space efficiency. The model generates a confidence score for each element, and elements with high confidence are directly classified as set members, while elements with low confidence are verified by the backup Bloom Filter (Bloom, 1970; Mitzenmacher, 2018).

Definition 3 (Learned Bloom Filter (Kraska et al., 2018; Mitzenmacher, 2018; Liu et al., 2020))**.** *Given a set $\mathcal{S}$, a Learned Bloom Filter (LBF) is defined as a tuple $(f_\theta, \tau, \mathrm{BF})$, where $f_\theta : x \to [0, 1]$ is a learned classifier and τ is a decision threshold. An element x is predicted as a member if $f_\theta(x) > \tau$; otherwise, it is queried against the backup Bloom filter* BF*, which stores elements in $\{x \in \mathcal{S} \mid f_\theta(x) \leq \tau\}$.*

4 METHODOLOGY

4.1 OVERVIEW

DP-LBF combines a learned model with a privately released backup Bloom filter for membership queries over a sensitive set $\mathcal{S}$. We assume that the model f_θ is trained on data disjoint from $\mathcal{S}$ and is fixed before $\mathcal{S}$ is accessed. The finite candidate set $\mathcal{T}$, a public negative calibration set $\mathcal{C}^-_{\mathrm{pub}}$, the penalty weight λ, a public count bound B, the target Bloom-filter error rate ρ, and all hash-generation rules are also fixed in advance. Consequently, these quantities consume no privacy budget.

The private construction uses three sequentially composed stages. First, Privacy Threshold Selection (PTS) samples a routing threshold τ^* with budget ϵ_τ. Its sensitivity-one utility rewards direct recovery of sensitive members while penalizing false positives measured only on $\mathcal{C}^-_{\mathrm{pub}}$. Second, Noisy Backup-Size Release (NBSR) uses budget ϵ_m to release a Laplace-perturbed count $\tilde{t}$ of the sensitive members routed to the backup filter. Third, Adaptive Differentially Private Bloom Filter construction (ABFDP) uses $\tilde{t}$ to determine the filter geometry, inserts the low-confidence members internally, and releases only a bitwise-randomized filter $\tilde{b}$ with budget ϵ_b. Thus,

$$\epsilon = \epsilon_\tau + \epsilon_m + \epsilon_b.$$

The released DP-LBF state is $R = (\tau^*, \tilde{t}, \tilde{b})$. Membership answers computed from R and the fixed public model are post-processing of this release.

Algorithm 1: Overall Structure of the DP-LBF Framework

Input: Sensitive set $\mathcal{S}$; query set $\mathcal{Q}$; fixed public model f_θ; fixed public inputs $\mathcal{T}$, $\mathcal{C}^-_{\mathrm{pub}}$, λ, B, and ρ; budgets ϵ_τ, ϵ_m, and ϵ_b.
Output: Released state $R = (\tau^*, \tilde{t}, \tilde{b})$ and query answers $\hat{\mathcal{Y}}$.

$\tau^* \leftarrow \mathrm{PTS}(\mathcal{S}, f_\theta, \mathcal{T}, \mathcal{C}^-_{\mathrm{pub}}, \lambda, B, \epsilon_\tau)$ (Algorithm 2);
$\tilde{t} \leftarrow \mathrm{NBSR}(\mathcal{S}, f_\theta, \tau^*, B, \epsilon_m)$ (Algorithm 3);
$\tilde{b} \leftarrow \mathrm{ABFDP}(\mathcal{S}, f_\theta, \tau^*, \tilde{t}, \rho, \epsilon_b)$ (Algorithm 4);
$R \leftarrow (\tau^*, \tilde{t}, \tilde{b})$;
for *each query element* $x \in \mathcal{Q}$ **do**
  $s_x \leftarrow f_\theta(x)$;
  if $s_x > \tau^*$ **then**
    $\hat{y}_x \leftarrow 1$; `// high confidence: return directly`
  else
    $\hat{y}_x \leftarrow \mathrm{QueryBF}(x, \tilde{b})$; `// low confidence: backup lookup`
return R and $\hat{\mathcal{Y}}$;

4.2 PRIVACY THRESHOLD SELECTION (PTS)

Algorithm 2: Privacy Threshold Selection (PTS)

Input: Sensitive set $\mathcal{S}$; fixed public model f_θ; candidate set $\mathcal{T}$; public negative calibration set $\mathcal{C}^-_{\text{pub}}$; penalty weight λ; public count bound B; budget ϵ_τ.
Output: Private routing threshold τ^*.

$\Delta u \leftarrow 1$;
for *each threshold* $\tau \in \mathcal{T}$ **do**
  $A_\tau \leftarrow |\{x \in \mathcal{S} : f_\theta(x) > \tau\}|$;
  $\text{FPR}_{\text{pub}}(\tau) \leftarrow \frac{1}{|\mathcal{C}^-_{\text{pub}}|} \sum_{z \in \mathcal{C}^-_{\text{pub}}} \mathbf{1}\{f_\theta(z) > \tau\}$;
  $C_{\text{pub}}(\tau) \leftarrow B\,\text{FPR}_{\text{pub}}(\tau)$;
  $u_\tau \leftarrow A_\tau - \lambda C_{\text{pub}}(\tau)$;
  $\ell_\tau \leftarrow \epsilon_\tau u_\tau / (2\Delta u)$;
Set $p_\tau \propto \exp(\ell_\tau - \max_{r \in \mathcal{T}} \ell_r)$ for all $\tau \in \mathcal{T}$;
Sample τ^* from $\mathcal{T}$ according to p_τ ;
return τ^*;

For each public candidate τ, define

$$u(\mathcal{S}, \tau) = |\{x \in \mathcal{S} : f_\theta(x) > \tau\}| - \lambda B\,\text{FPR}_{\text{pub}}(\tau).$$

The first term counts sensitive members answered directly by the model, whereas the second term penalizes thresholds that falsely accept public calibration non-members. Because f_θ, $\mathcal{T}$, $\mathcal{C}^-_{\text{pub}}$, λ, and B are fixed before $\mathcal{S}$ is accessed, the penalty has zero sensitivity. Adding or removing one member changes only the first term, by at most one; hence $\Delta u = 1$.

The exponential mechanism samples

$$p_\tau = \frac{\exp(\epsilon_\tau u(\mathcal{S}, \tau)/2)}{\sum_{r \in \mathcal{T}} \exp(\epsilon_\tau u(\mathcal{S}, r)/2)}.$$

Algorithm 2 evaluates these weights in log space for numerical stability.

4.3 NOISY BACKUP-SIZE RELEASE (NBSR)

Algorithm 3: Noisy Backup-Size Release (NBSR)

Input: Sensitive set $\mathcal{S}$; fixed public model f_θ; released threshold τ^*; public count bound B; budget ϵ_m.
Output: Noisy backup capacity $\tilde{t}$.

$\mathcal{D}_{\text{FN}} \leftarrow \{x \in \mathcal{S} : f_\theta(x) \le \tau^*\}$; `// internal; not released`
$t \leftarrow |\mathcal{D}_{\text{FN}}|$;
Draw $Z \sim \text{Lap}(0, 1/\epsilon_m)$;
$\tilde{t} \leftarrow \text{clip}_{[0,B]}(\text{round}(t + Z))$;
return $\tilde{t}$;

For a fixed released threshold τ^*, the backup count is

$$t(\mathcal{S}, \tau^*) = |\{x \in \mathcal{S} : f_\theta(x) \le \tau^*\}|.$$

Under add/remove adjacency, its sensitivity is one. NBSR adds Laplace noise of scale $1/\epsilon_m$, rounds the result, and clips it to the public interval $[0, B]$. Rounding and clipping are post-processing. Only $\tilde{t}$ is released; the exact count t and the set $\mathcal{D}_{\text{FN}}$ remain internal.

4.4 ADAPTIVE DIFFERENTIALLY PRIVATE BLOOM FILTER (ABFDP)

For a released count $s \in \{1, \dots, B\}$ and a fixed target false-positive rate $\rho \in (0, 1)$, define the public Bloom-filter geometry by

$$k(s, \rho) = \max\left\{1, \left\lceil \log_2 \frac{1}{\rho} \right\rceil\right\},$$
$$M(s, \rho) = k(s, \rho) \left\lceil -\frac{s \ln \rho}{k(s, \rho)(\ln 2)^2} \right\rceil. \tag{1}$$

This sliced-array rule is the implementation-level form of the standard continuous optimum $M \simeq -s \ln \rho/(\ln 2)^2$ and $k \simeq (M/s) \ln 2$, with integer rounding made explicit. Let $\mathcal{H}_{\text{pub}} = (h_1, h_2, \dots)$ be a hash family, including its seeds, fixed before the sensitive set is accessed. For $s > 0$, $\text{PublicGeometry}(s, \rho)$ returns $(M(s, \rho), k(s, \rho), \{h_i\}_{i=1}^{k(s,\rho)})$. For $s = 0$, define

$$\text{PublicGeometry}(0, \rho) = (0, 0, \varnothing),$$

which denotes a fixed empty backup-filter representation whose query result is always false. Thus the complete geometry rule is a deterministic function of the DP release s and public quantities only.

Algorithm 4: Adaptive Differentially Private Bloom Filter (ABFDP)

Input: Sensitive set $\mathcal{S}$; fixed public model f_θ; released threshold τ^*; released noisy capacity $\tilde{t}$; fixed target error rate ρ; budget ϵ_b.
Output: Randomized backup Bloom filter $\tilde{b}$.

$\mathcal{D}_{\text{FN}} \leftarrow \{x \in \mathcal{S} : f_\theta(x) \le \tau^*\}$; `// internal; not released`
$(M, k, \{h_i\}_{i=1}^{k}) \leftarrow \text{PublicGeometry}(\tilde{t}, \rho)$;
if $M = 0$ **then**
  return the fixed empty backup filter $\varnothing$;
$b \leftarrow (0, \dots, 0) \in \{0, 1\}^M$;
for *each* $x \in \mathcal{D}_{\text{FN}}$ **do**
  for $i \leftarrow 1$ **to** k **do**
    $b[h_i(x)] \leftarrow 1$;
$p \leftarrow \frac{\exp(\epsilon_b/k)}{\exp(\epsilon_b/k)+1}$;
for $j \leftarrow 1$ **to** M **do**
  Draw $r_j \sim \text{Uniform}(0, 1)$;
  $\tilde{b}[j] \leftarrow b[j]$ if $r_j \le p$, and $1 - b[j]$ otherwise;
return $\tilde{b}$;

Conditioned on the released values $(\tau^*, \tilde{t})$, ABFDP reconstructs the internal low-confidence set

$$\mathcal{D}_{\text{FN}} = \{x \in \mathcal{S} : f_\theta(x) \le \tau^*\}.$$

This set is never released. Equation (1) fixes the bit-array length and number of hash functions when $\tilde{t} > 0$, and the first k functions of $\mathcal{H}_{\text{pub}}$ fix their public seeds. When $\tilde{t} = 0$, the same public rule returns the fixed empty representation. All elements of $\mathcal{D}_{\text{FN}}$ are inserted into the resulting geometry when $\tilde{t} > 0$. In particular, construction does not abort when the exact count t exceeds $\tilde{t}$, because such a data-dependent failure would reveal information about t.

For $\tilde{t} > 0$, the internal bit vector is randomized independently at every position. With

$$p = \frac{\exp(\epsilon_b/k)}{\exp(\epsilon_b/k) + 1},$$

the released bit satisfies

$$\tilde{b}[j] = \begin{cases} b[j], & \text{with probability } p, \\ 1 - b[j], & \text{with probability } 1 - p. \end{cases}$$

For fixed $(\tau^*, \tilde{t})$ with $\tilde{t} > 0$, adding or removing one element changes at most k positions of the unperturbed bit vector. Assigning parameter ϵ_b/k to each affected bit bounds the combined privacy loss by ϵ_b. When $\tilde{t} = 0$, the fixed empty representation is data independent and incurs zero additional privacy loss. The only filter representation released to the querier is $\tilde{b}$.

5 THEORETICAL ANALYSIS

5.1 PRIVACY ANALYSIS

Theorem 2 (Privacy of threshold selection)**.** *Algorithm 2 satisfies ϵ_τ-differential privacy under add/remove adjacency.*

Proof. The private count in the utility has sensitivity one, whereas the FPR penalty is computed entirely from public data and has sensitivity zero. The claim then follows from the exponential mechanism with $\Delta u = 1$. The full probability ratio proof is given in Appendix B.1. □

Theorem 3 (Privacy of noisy backup-size release)**.** *Conditioned on a previously released threshold τ^*, Algorithm 3 satisfies ϵ_m-differential privacy.*

Proof. For every fixed value of τ^*, the backup count has sensitivity one. The Laplace mechanism with scale $1/\epsilon_m$ is ϵ_m-DP, and rounding and clipping are post-processing. The full proof is given in Appendix B.2. □

Theorem 4 (Privacy of the randomized backup filter)**.** *Conditioned on the previously released pair $(\tau^*, \widetilde{t})$, Algorithm 4 satisfies ϵ_b-differential privacy.*

Proof. Conditioned on $(\tau^*, \widetilde{t})$, the filter geometry is fixed, and one record can change at most k bits of the unperturbed filter. Assigning privacy parameter ϵ_b/k to the randomized response of each affected bit bounds the total likelihood ratio by e^{ϵ_b}. The full proof is given in Appendix B.3. □

Theorem 5 (Privacy guarantee of DP-LBF)**.** *The released state $R = (\tau^*, \widetilde{t}, \widetilde{\boldsymbol{b}})$ satisfies ϵ-differential privacy for $\epsilon = \epsilon_\tau + \epsilon_m + \epsilon_b$.*

Proof. The three releases satisfy ϵ_τ-, ϵ_m-, and ϵ_b-DP conditionally on the outputs of the preceding stages. Adaptive sequential composition gives the stated sum, and membership queries are post-processing of the released state. The full joint-distribution proof is given in Appendix B.4. □

5.2 UTILITY ANALYSIS

Theorem 6 (Approximate utility of DP-LBF)**.** *Condition on the released pair $(\tau^*, \widetilde{t})$, and let t be the exact internal number of inserted members and $M = M(\widetilde{t}, \rho)$ and $k = k(\widetilde{t}, \rho)$ be given by Equation* (1)*. Let $\mathcal{Q}_{\mathrm{FN}}$ denote low-score members, $\mathcal{Q}_{\mathrm{FP}}$ high-score non-members, and $\mathcal{Q}_{\mathrm{TN}}$ low-score non-members. For $\widetilde{t} > 0$, under the standard Bloom-filter occupancy and independent-position approximations, define*

$$a = 1 - \exp\left(-\frac{kt}{M}\right), \qquad p = \frac{\exp(\epsilon_b/k)}{\exp(\epsilon_b/k) + 1}, \qquad r = ap + (1-a)(1-p).$$

Then

$$\mathbb{E}\left[\sum_{x \in \mathcal{Q}} (\widehat{y}_x - y_x)^2 \,\middle|\, \tau^*, \widetilde{t}\right] \approx |\mathcal{Q}_{\mathrm{FN}}|(1-p^k) + |\mathcal{Q}_{\mathrm{FP}}| + |\mathcal{Q}_{\mathrm{TN}}| r^k. \tag{2}$$

If $\widetilde{t} = 0$, the conditional loss is exactly $|\mathcal{Q}_{\mathrm{FN}}| + |\mathcal{Q}_{\mathrm{FP}}|$.

Proof. Partitioning the query set into the three stated error-producing cases gives the expression by linearity of expectation. The approximation comes only from the standard Bloom-filter occupancy and independent-position assumptions; it is not a finite-sample inequality. The full derivation, including the unconditional loss and its relation to RMSE, is given in Appendix B.5. □

5.3 Space Complexity

Theorem 7 (Space complexity of DP-LBF)**.** *Under the deterministic geometry in Equation* (1)*, conditioned on the released noisy capacity $\widetilde{t}$, the total serving-space complexity, including the fixed public model, is*

$$O\left(|\theta| + \max\{1, \widetilde{t}\} \log \frac{1}{\rho}\right).$$

The released private representation itself requires

$$O\left(\max\{1, \widetilde{t}\} \log \frac{1}{\rho}\right)$$

space. Since $\widetilde{t} \in [0, B]$, the corresponding worst-case bounds are $O(|\theta| + B \log(1/\rho))$ and $O(B \log(1/\rho))$, respectively.

Proof. The fixed public model occupies $O(|\theta|)$ space and introduces no storage that depends on the sensitive set. The two released scalars require $O(1)$ words. The backup-filter geometry is determined by $\widetilde{t}$, not by the exact internal count t. Equation (1) and the clipping $\widetilde{t} \leq B$ give the stated conditioned and worst-case bounds. Appendix B.6 provides the full proof and separates total serving space from the released private representation size. □

Each query evaluates the fixed public model once, while only a low-score query performs backup-filter hashing and bit probes. The formal routing-sensitive and worst-case query-complexity bounds are stated and proved in Appendix B.7.

6 Performance Evaluation

6.1 Datasets and Experiment Setup

Datasets. We evaluate DP-LBF on one controlled synthetic dataset and two real-world binary datasets, URL and Spambase. The synthetic data provide a balanced setting with informative and nuisance features, whereas URL and Spambase represent phishing detection and e-mail spam detection, respectively. Additional dataset details are given in Appendix C.1.

Comparison protocol. We compare DP-LBF with four private set-representation baselines: BLIP (Alaggan et al., 2012), DPSet (Patel et al., 2024), DPBloomFilter (Ke et al., 2025), and Warner-LBF (Tirmazi, 2025). Every method receives the same sensitive member set and is evaluated on the same balanced query workload. We report RMSE between the binary membership answers and ground truth; lower values are better. Each plotted value is the mean of 100 independent randomized runs. Implementation and metric details are deferred to Appendices C.2–C.4.

Parameter settings. We use five uniformly spaced total privacy budgets $\epsilon \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$. DP-LBF uses the fixed allocation $(\epsilon_\tau, \epsilon_m, \epsilon_b) = (0.05\epsilon, 0.05\epsilon, 0.90\epsilon)$, i.e., the ratio 1:1:18, with public penalty weight $\lambda = 1$ and target Bloom-filter error rate $\rho = 0.01$. All settings are fixed before the sensitive set is accessed; complete configurations are reported in Appendices C.3–C.4.

6.2 Membership-Query Accuracy

Figure 2 shows that DP-LBF achieves the lowest mean RMSE across the evaluated privacy budgets. At $\epsilon = 2.5$, the RMSE values of DP-LBF, Warner-LBF, DPSet, BLIP, and DPBloomFilter are 0.1648, 0.2760, 0.2777, 0.7014, and 0.7046 on Synthetic, respectively. On URL, the corresponding values are 0.0129, 0.2830, 0.2775, 0.7013, and 0.7049. On Spambase, they are 0.2727, 0.2729, 0.2786, 0.7012, and 0.7052. DP-LBF therefore provides substantial improvements on Synthetic and URL while remaining comparable to Warner-LBF on Spambase. As ϵ increases, its RMSE decreases clearly on URL and remains stable on Synthetic and Spambase.

This performance is mainly attributed to DP-LBF's selective routing design. The fixed public model directly answers high-confidence queries, while only low-score queries are verified by the private backup filter. Consequently, performance on Synthetic and Spambase is largely determined by the

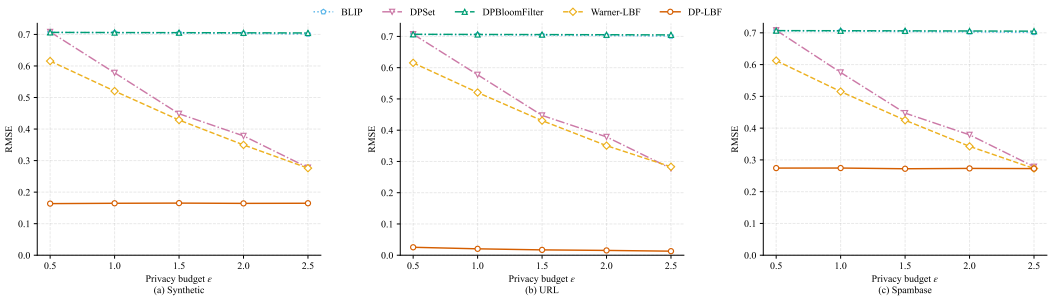


Figure 2: RMSE comparison of DP-LBF and private baselines.

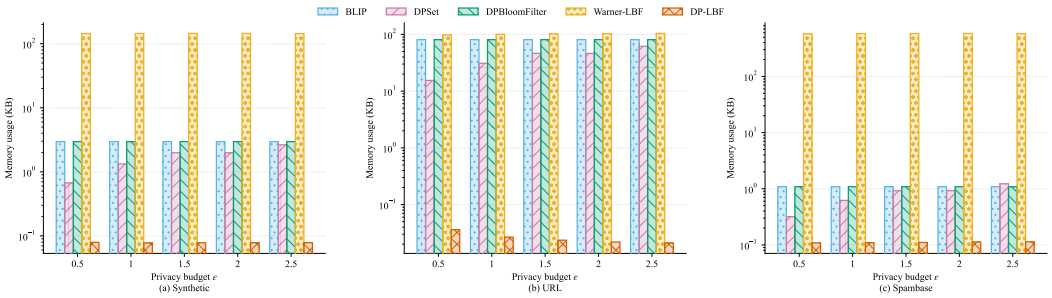


Figure 3: Memory-usage comparison of DP-LBF and private baselines on a logarithmic scale.

public model and routing structure, whereas URL benefits more from the increased budgets assigned to the private stages. The ablations in Figures 5, 6, and 7 further examine the effects of budget allocation, the public FPR penalty, and backup-filter geometry.

6.3 Released-Representation Size

Figure 3 shows that DP-LBF has the smallest released private representation in all 15 tested settings. Across the three datasets, its packed released state occupies 0.021–0.114 KB, whereas the smallest DPSet representation is 0.318 KB and Warner-LBF requires 98.126–583.688 KB. DP-LBF is compact because only low-score members are routed to a backup filter whose geometry is determined from the private noisy count. Complete per-dataset values and the accounting boundary are given in Appendix C.6.

7 Conclusion

In this paper, we proposed the Differentially Private Learned Bloom Filter (DP-LBF), which integrates a fixed public learned model with private threshold selection, noisy backup sizing, and a randomized backup Bloom filter. We established differential privacy under add/remove adjacency and evaluated the proposed method on synthetic, URL, and Spambase datasets. Experimental results demonstrate that DP-LBF achieves the lowest RMSE among the evaluated private baselines while maintaining a compact released representation. Future work will extend DP-LBF to dynamic and streaming settings while preserving rigorous privacy guarantees and storage efficiency.

AI use statement

Generative AI tools were used solely for language polishing and improving the readability of the manuscript. The authors reviewed and approved the final content.

Reproducibility statement

The DP-LBF mechanism and experimental setup are described in the main paper. Complete theoretical proofs and additional experimental details are provided in the appendix. The supplementary materials include the source code required to reproduce the experimental results.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Mohammad Alaggan, Sébastien Gambs, and Anne-Marie Kermarrec. Blip: Non-interactive differentially-private similarity computation on bloom filters. In *Stabilization, Safety, and Security of Distributed Systems*, pp. 202–216. Springer, 2012. doi: 10.1007/978-3-642-33536-5_20.

Mayank Bawa, Roberto J Bayardo Jr, Rakesh Agrawal, and Jaideep Vaidya. Privacy-preserving indexing of documents on the network. *The VLDB Journal*, 18(4):837–856, 2009.

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

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*, 35(6):7499–7519, 2022.

Jonathan Brophy, Zayd Hammoudeh, and Daniel Lowd. Adapting and evaluating influence-estimation methods for gradient-boosted decision trees. *Journal of Machine Learning Research*, 24(154): 1–48, 2023.

Guanduo Chen, Zhenying He, Meng Li, and Siqiang Luo. Oasis: An optimal disjoint segmented learned range filter. *Proceedings of the VLDB Endowment*, 17(8):1911–1924, 2024.

David Clayton, Christopher Patton, and Thomas Shrimpton. Probabilistic data structures in adversarial environments. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1317–1334, 2019.

Vinícius G Costa and Carlos E Pedreira. Recent advances in decision trees: an updated survey. *Artificial Intelligence Review*, 56(5):4765–4800, 2023.

Haipeng Dai, Hancheng Wang, Zhipeng Chen, Jiaqi Zheng, Meng Li, Rong Gu, Chen Tian, and Wanchun Dou. Variable-length encoding framework: A generic framework for enhancing the accuracy of approximate membership queries. In *2023 IEEE International Conference on Data Mining (ICDM)*, pp. 61–70. IEEE, 2023.

Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pp. 1–12. Springer, 2006.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pp. 265–284. Springer, 2006.

Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 715–724, 2010.

Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 1054–1067, 2014.

Mia Filic, Kenneth G Paterson, Anupama Unnikrishnan, and Fernando Virdia. Adversarial correctness and privacy for probabilistic data structures. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1037–1050, 2022.

Guanxiong Ha, Xiaowei Ge, Chunfu Jia, Yuchen Chen, and Zhen Su. Revisiting sgx-based encrypted deduplication via pow-before-encryption and eliminating redundant computations. *IEEE Transactions on Dependable and Secure Computing*, 2024.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(11): 7436–7456, 2021.

Muneeb Ul Hassan, Mubashir Husain Rehmani, and Jinjun Chen. Differential privacy techniques for cyber physical systems: A survey. *IEEE Communications Surveys & Tutorials*, 22(1):746–789, 2019.

Christina Ilvento. Implementing the exponential mechanism with base-2 differential privacy. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 717–742, 2020.

Leonid Iosipoi and Anton Vakhrushev. Sketchboost: Fast gradient boosted decision tree for multioutput problems. *Advances in Neural Information Processing Systems*, 35:25422–25435, 2022.

Shankui Ji, Yang Du, He Huang, Yu-E Sun, Jia Liu, and Yapeng Shu. Pipefilter: Parallelizable and space-efficient filter for approximate membership query. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Yekun Ke, Yingyu Liang, Zhizhou Sha, Zhenmei Shi, Zhao Song, and Jiahao Zhang. Dpbloomfilter: Securing bloom filters with differential privacy. *arXiv preprint arXiv:2502.00693*, 2025.

Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 international conference on management of data*, pp. 489–504, 2018.

Zheng Li and Yang Zhang. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 880–895, 2021.

Yanrong Liang, Jianfeng Ma, Yinbin Miao, Da Kuang, Xiangdong Meng, and Robert H Deng. Privacy-preserving bloom filter-based keyword search over large encrypted cloud data. *IEEE Transactions on Computers*, 72(11):3086–3098, 2023.

Qiyu Liu, Libin Zheng, Yanyan Shen, and Lei Chen. Stable learned bloom filters for data streams. *Proceedings of the VLDB Endowment*, 13(12):2355–2367, 2020.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pp. 94–103. IEEE, 2007.

Michael Mitzenmacher. A model for learned bloom filters and optimizing by sandwiching. *Advances in neural information processing systems*, 31, 2018.

Géraldin Nanfack, Paul Temple, and Benoît Frénay. Constraint enforcement on decision trees: A survey. *ACM Computing Surveys (CSUR)*, 54(10s):1–36, 2022.

Prashant Pandey, Martín Farach-Colton, Niv Dayan, and Huanchen Zhang. Beyond bloom: A tutorial on future feature-rich filters. In *Companion of the 2024 International Conference on Management of Data*, pp. 636–644, 2024.

Sarvar Patel, Giuseppe Persiano, Joon Young Seo, and Kevin Yeo. Differentially private set representations. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2602.

Antonio Paya, Sergio Arroni, Vicente García-Díaz, and Alberto Gómez. Apollon: a robust defense system against adversarial machine learning attacks in intrusion detection systems. *Computers & Security*, 136:103546, 2024.

Mian Qin, Qing Zheng, Jason Lee, Bradley Settlemyer, Fei Wen, Narasimha Reddy, and Paul Gratz. Kvrangedb: Range queries for a hash-based key–value device. *ACM transactions on storage*, 19 (3):1–21, 2023.

Haifeng Sun, Jiaheng Li, Jintao He, Jie Gui, and Qun Huang. Omniwindow: A general and efficient window mechanism framework for network telemetry. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pp. 867–880, 2023.

Hayder Tirmazi. Adversarially robust bloom filters: Privacy, reductions, and open problems. *arXiv preprint arXiv:2501.15751*, 2025.

Jelle Vos, Sikha Pentyala, Steven Golob, Ricardo Maia, Dean Kelley, Zekeriya Erkin, Martine De Cock, and Anderson Nascimento. Privacy-preserving membership queries for federated anomaly detection. *Proceedings on Privacy Enhancing Technologies*, 2024a.

Jelle Vos, Jorrit van Assen, Tjitske Koster, Evangelia Anna Markatou, and Zekeriya Erkin. On the insecurity of bloom filter-based private set intersections. *Cryptology ePrint Archive*, 2024b.

Cong Wang, Chen Chen, QingQi Pei, Zhiyuan Jiang, and Shugong Xu. An information-centric in-network caching scheme for 5g-enabled internet of connected vehicles. *IEEE Transactions on Mobile Computing*, 22(6):3137–3150, 2021.

Hancheng Wang, Haipeng Dai, Meng Li, Jun Yu, Rong Gu, Jiaqi Zheng, and Guihai Chen. Bamboo filters: Make resizing smooth. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pp. 979–991. IEEE, 2022.

Hancheng Wang, Haipeng Dai, Shusen Chen, Meng Li, Rong Gu, Huayi Chai, Jiaqi Zheng, Zhiyuan Chen, Shuaituan Li, Xianjun Deng, et al. Bamboo filters: make resizing smooth and adaptive. *IEEE/ACM Transactions on Networking*, 32(5):3776–3791, 2024.

Wanli Xue, Dinusha Vatsalan, Wen Hu, and Aruna Seneviratne. Sequence data matching and beyond: New privacy-preserving primitives based on bloom filters. *IEEE Transactions on Information Forensics and Security*, 15:2973–2987, 2020.

Meiyi Yang, Deyun Gao, Chuan Heng Foh, Yajuan Qin, and Victor CM Leung. A learned bloom filter-assisted scheme for packet classification in software-defined networking. *IEEE Transactions on Network and Service Management*, 19(4):5064–5077, 2022.

Zakia Zaman, Wanli Xue, Praveen Gauravaram, Wen Hu, Jiaojiao Jiang, and Sanjay K Jha. Privacy-preserving probabilistic data encoding for iot data analysis. *IEEE Transactions on Information Forensics and Security*, 19:9173–9187, 2024.

Ellen Z Zhang, Yunguo Guan, Rongxing Lu, and Harry Zhang. An efficient bloom filter-based range query scheme under local differential privacy. In *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6. IEEE, 2023.

Tianqing Zhu, Dayong Ye, Wei Wang, Wanlei Zhou, and Philip S Yu. More than privacy: Applying differential privacy in key areas of artificial intelligence. *IEEE Transactions on Knowledge and Data Engineering*, 34(6):2824–2843, 2020.

A Notations and Definitions

Table 1: Main notation used in DP-LBF.

Notation	Description
$\mathcal{S}$	Sensitive member set.
$\mathcal{Q}$	Query set.
f_θ	Fixed public scoring model; $f_\theta(x)$ is the confidence score of element x.
$\mathcal{T}$	Fixed finite public set of candidate routing thresholds.
$\mathcal{C}^-_{\mathrm{pub}}$	Fixed public calibration set containing non-members.
τ, τ^*	A candidate threshold and the private threshold sampled by PTS, respectively.
B, λ, ρ	Fixed public count bound, FPR-penalty weight, and target backup-filter error rate.
ϵ	Total privacy budget of the released DP-LBF state.
$\epsilon_\tau, \epsilon_m, \epsilon_b$	Budgets for PTS, NBSR, and ABFDP, with $\epsilon = \epsilon_\tau + \epsilon_m + \epsilon_b$.
$\mathcal{D}_{\mathrm{FN}}$	Internal low-confidence member set routed to the backup Bloom filter; it is not released.
$t, \widetilde{t}$	Exact internal backup count $t = \lvert\mathcal{D}_{\mathrm{FN}}\rvert$ and its rounded, clipped Laplace release.
$M, k, \{h_i\}_{i=1}^k$	Backup bit-array length, number of hash functions, and public hash functions determined from $(\widetilde{t}, \rho)$ by Equation (1).
$b, \widetilde{b}$	Internal unperturbed backup bit vector and the released randomized bit vector.
p	Per-bit retention probability in ABFDP, $p = e^{\epsilon_b/k}/(e^{\epsilon_b/k}+1)$.
R	Released DP-LBF state $R = (\tau^*, \widetilde{t}, \widetilde{b})$.
$\widehat{y}_x, y_x$	Predicted membership of query x and its true membership label.

B Full Proofs of theoretical analysis

B.1 Full Proof of Theorem 2

Proof. Let $f_\theta, \mathcal{T}, \mathcal{C}^-_{\mathrm{pub}}, \lambda$, and B be fixed before the sensitive set is accessed. For each $\tau \in \mathcal{T}$, define

$$A_\tau(\mathcal{S}) = |\{x \in \mathcal{S} : f_\theta(x) > \tau\}|, \qquad u(\mathcal{S}, \tau) = A_\tau(\mathcal{S}) - \lambda B\, \mathrm{FPR}_{\mathrm{pub}}(\tau).$$

If $\mathcal{S} \simeq \mathcal{S}'$ under add/remove adjacency, then for every candidate τ,

$$|A_\tau(\mathcal{S}) - A_\tau(\mathcal{S}')| \le 1.$$

The FPR term is evaluated only on the public calibration set and is therefore identical on $\mathcal{S}$ and $\mathcal{S}'$. Consequently,

$$|u(\mathcal{S}, \tau) - u(\mathcal{S}', \tau)| \le 1,$$

so the data-independent global sensitivity bound is $\Delta u = 1$.

Let

$$Z(\mathcal{S}) = \sum_{r \in \mathcal{T}} \exp\left(\frac{\epsilon_\tau u(\mathcal{S}, r)}{2}\right).$$

The exponential mechanism samples

$$\Pr[\tau^* = \tau \mid \mathcal{S}] = \frac{\exp(\epsilon_\tau u(\mathcal{S}, \tau)/2)}{Z(\mathcal{S})}.$$

For every neighboring pair and every candidate τ, sensitivity one gives

$$\exp\left(\frac{\epsilon_\tau[u(\mathcal{S}, \tau) - u(\mathcal{S}', \tau)]}{2}\right) \le e^{\epsilon_\tau/2}.$$

It also gives $u(\mathcal{S}', r) \le u(\mathcal{S}, r) + 1$ for every $r \in \mathcal{T}$, and hence

$$Z(\mathcal{S}') \le e^{\epsilon_\tau/2} Z(\mathcal{S}).$$

Combining the numerator and normalizer bounds yields

$$\frac{\Pr[\tau^* = \tau \mid \mathcal{S}]}{\Pr[\tau^* = \tau \mid \mathcal{S}']} = \exp\left(\frac{\epsilon_\tau[u(\mathcal{S},\tau) - u(\mathcal{S}',\tau)]}{2}\right)\frac{Z(\mathcal{S}')}{Z(\mathcal{S})} \leq e^{\epsilon_\tau}.$$

Therefore, for every event $O \subseteq \mathcal{T}$,

$$\begin{aligned}\Pr[\tau^* \in O \mid \mathcal{S}] &= \sum_{\tau \in O} \Pr[\tau^* = \tau \mid \mathcal{S}] \\ &\leq e^{\epsilon_\tau} \sum_{\tau \in O} \Pr[\tau^* = \tau \mid \mathcal{S}'] \\ &= e^{\epsilon_\tau} \Pr[\tau^* \in O \mid \mathcal{S}'].\end{aligned}$$

Thus PTS satisfies ϵ_τ-differential privacy. □

B.2 Full Proof of Theorem 3

Proof. Fix an arbitrary threshold value τ that may have been released by the first stage, and define

$$t_\tau(\mathcal{S}) = |\{x \in \mathcal{S} : f_\theta(x) \leq \tau\}|.$$

For neighboring sets $\mathcal{S} \simeq \mathcal{S}'$, adding or removing one record changes this count by at most one:

$$|t_\tau(\mathcal{S}) - t_\tau(\mathcal{S}')| \leq 1.$$

Before rounding and clipping, NBSR releases $Y = t_\tau(\mathcal{S}) + Z$, where $Z \sim \mathrm{Lap}(0, 1/\epsilon_m)$. Its conditional density is

$$q_{\mathcal{S},\tau}(y) = \frac{\epsilon_m}{2}\exp(-\epsilon_m|y - t_\tau(\mathcal{S})|).$$

For every $y \in \mathbb{R}$, the reverse triangle inequality gives

$$\begin{aligned}\frac{q_{\mathcal{S},\tau}(y)}{q_{\mathcal{S}',\tau}(y)} &= \exp\big(\epsilon_m\big[|y - t_\tau(\mathcal{S}')| - |y - t_\tau(\mathcal{S})|\big]\big) \\ &\leq \exp(\epsilon_m|t_\tau(\mathcal{S}) - t_\tau(\mathcal{S}')|) \\ &\leq e^{\epsilon_m}.\end{aligned}$$

Integrating this pointwise inequality over any measurable set establishes ϵ_m-DP for Y, conditional on the fixed earlier output τ.

The released capacity is $\widetilde{t} = g(Y)$, where $g(y) = \mathrm{clip}_{[0,B]}(\mathrm{round}(y))$ is deterministic and data independent. For every event $O \subseteq \{0, 1, \ldots, B\}$,

$$\begin{aligned}\Pr[\widetilde{t} \in O \mid \mathcal{S}, \tau] &= \Pr[Y \in g^{-1}(O) \mid \mathcal{S}, \tau] \\ &\leq e^{\epsilon_m} \Pr[Y \in g^{-1}(O) \mid \mathcal{S}', \tau] \\ &= e^{\epsilon_m} \Pr[\widetilde{t} \in O \mid \mathcal{S}', \tau].\end{aligned}$$

Thus rounding and clipping preserve the ϵ_m guarantee. □

B.3 Full Proof of Theorem 4

Proof. Fix arbitrary outputs (τ, s) of the first two stages. If $s = 0$, the zero-count definition of $\mathrm{PublicGeometry}$ gives $(M, k, \{h_i\}) = (0, 0, \varnothing)$, so Algorithm 4 returns the same public empty-filter representation for every sensitive set. The conditional mechanism is then data independent and satisfies 0-DP, which implies ϵ_b-DP.

Now suppose $s > 0$. Because s, ρ, and all hash-generation rules are public, conditioning on $\widetilde{t} = s$ fixes $M = M(s, \rho)$ and $k = k(s, \rho)$ through Equation (1), together with the first k functions of $\mathcal{H}_{\mathrm{pub}}$. Let

$$\boldsymbol{b}(\mathcal{S}; \tau, s) = (b_1, \ldots, b_M) \in \{0, 1\}^M$$

be the unperturbed filter obtained by inserting $\mathcal{D}_{\mathrm{FN}}(\mathcal{S};\tau) = \{x \in \mathcal{S} : f_\theta(x) \le \tau\}$. For neighboring sets $\mathcal{S} \simeq \mathcal{S}'$, the differing record either is routed directly by the model, in which case the two bit vectors are identical, or is routed to the backup filter. In the latter case it can affect only its k hash locations. Hash collisions can only reduce this number. Therefore the index set

$$\mathcal{I} = \{i : b_i(\mathcal{S};\tau,s) \neq b_i(\mathcal{S}';\tau,s)\}$$

satisfies $|\mathcal{I}| \le k$.

The mechanism independently randomizes each bit with retention probability

$$p = \frac{e^{\epsilon_b/k}}{e^{\epsilon_b/k}+1},$$

and flipping probability $1-p$. Thus, for every i,

$$\Pr[\widetilde{b}_i = b_i \mid b_i] = p, \qquad \Pr[\widetilde{b}_i = 1-b_i \mid b_i] = 1-p.$$

For any possible released bit vector $\boldsymbol{z} \in \{0,1\}^M$, independence across positions gives

$$\begin{aligned}
&\frac{\Pr[\widetilde{\boldsymbol{b}} = \boldsymbol{z} \mid \mathcal{S},\tau,s]}{\Pr[\widetilde{\boldsymbol{b}} = \boldsymbol{z} \mid \mathcal{S}',\tau,s]} \\
&= \prod_{i=1}^{M} \frac{\Pr[\widetilde{b}_i = z_i \mid b_i(\mathcal{S};\tau,s)]}{\Pr[\widetilde{b}_i = z_i \mid b_i(\mathcal{S}';\tau,s)]} \\
&= \prod_{i\in\mathcal{I}} \frac{\Pr[\widetilde{b}_i = z_i \mid b_i(\mathcal{S};\tau,s)]}{\Pr[\widetilde{b}_i = z_i \mid b_i(\mathcal{S}';\tau,s)]}.
\end{aligned}$$

The factors outside $\mathcal{I}$ cancel. For every changed position, the remaining factor is either $p/(1-p)$ or $(1-p)/p$, and hence

$$\max\left\{\frac{p}{1-p}, \frac{1-p}{p}\right\} = e^{\epsilon_b/k}.$$

Therefore,

$$\frac{\Pr[\widetilde{\boldsymbol{b}} = \boldsymbol{z} \mid \mathcal{S},\tau,s]}{\Pr[\widetilde{\boldsymbol{b}} = \boldsymbol{z} \mid \mathcal{S}',\tau,s]} \le e^{|\mathcal{I}|\epsilon_b/k} \le e^{\epsilon_b}.$$

Summing this pointwise inequality over any event in the output range proves that ABFDP is conditionally ϵ_b-differentially private for every fixed pair of earlier outputs (τ, s). □

B.4 Full Proof of Theorem 5

Proof. Let $R = (\tau^*, \widetilde{t}, \widetilde{\boldsymbol{b}})$ denote the complete released state. For a possible output $r = (\tau, s, \boldsymbol{z})$, the chain rule gives

$$\begin{aligned}
\Pr[R = r \mid \mathcal{S}] = \; &\Pr[\tau^* = \tau \mid \mathcal{S}] \\
&\cdot \Pr[\widetilde{t} = s \mid \mathcal{S}, \tau^* = \tau] \\
&\cdot \Pr[\widetilde{\boldsymbol{b}} = \boldsymbol{z} \mid \mathcal{S}, \tau^* = \tau, \widetilde{t} = s].
\end{aligned}$$

For neighboring sets $\mathcal{S} \simeq \mathcal{S}'$, the full proofs above establish, for every fixed value of the preceding outputs,

$$\begin{aligned}
\frac{\Pr[\tau^* = \tau \mid \mathcal{S}]}{\Pr[\tau^* = \tau \mid \mathcal{S}']} &\le e^{\epsilon_\tau}, \\
\frac{\Pr[\widetilde{t} = s \mid \mathcal{S}, \tau^* = \tau]}{\Pr[\widetilde{t} = s \mid \mathcal{S}', \tau^* = \tau]} &\le e^{\epsilon_m}, \\
\frac{\Pr[\widetilde{\boldsymbol{b}} = \boldsymbol{z} \mid \mathcal{S}, \tau^* = \tau, \widetilde{t} = s]}{\Pr[\widetilde{\boldsymbol{b}} = \boldsymbol{z} \mid \mathcal{S}', \tau^* = \tau, \widetilde{t} = s]} &\le e^{\epsilon_b}.
\end{aligned}$$

Multiplying the three conditional likelihood-ratio bounds yields

$$\frac{\Pr[R = r \mid \mathcal{S}]}{\Pr[R = r \mid \mathcal{S}']} \leq e^{\epsilon_\tau + \epsilon_m + \epsilon_b}.$$

Summing over every r in an arbitrary event O in the range of R gives

$$\Pr[R \in O \mid \mathcal{S}] \leq e^{\epsilon_\tau + \epsilon_m + \epsilon_b} \Pr[R \in O \mid \mathcal{S}'].$$

This is exactly $(\epsilon_\tau + \epsilon_m + \epsilon_b)$-differential privacy. Finally, the membership answers are deterministic functions of R and the fixed public model, so the post-processing property gives them the same privacy guarantee. □

B.5 Derivation of the Approximate Utility Analysis

Proof. Fix the sensitive set and condition on the released pair $(\tau^*, \widetilde{t}) = (\tau, s)$. The exact set inserted into the backup filter is

$$\mathcal{D}_{\mathrm{FN}}(\tau) = \{x \in \mathcal{S} : f_\theta(x) \leq \tau\}, \qquad t = |\mathcal{D}_{\mathrm{FN}}(\tau)|.$$

Although s determines the public filter geometry, the construction inserts all t elements; in particular, the analysis does not replace t by the noisy capacity s. For $s > 0$, set $M = M(s, \rho)$ and $k = k(s, \rho)$ according to Equation (1).

Partition the query set into

$$\mathcal{Q}_{\mathrm{FN}}(\tau) = \{x \in \mathcal{Q} : y_x = 1, f_\theta(x) \leq \tau\},$$
$$\mathcal{Q}_{\mathrm{FP}}(\tau) = \{x \in \mathcal{Q} : y_x = 0, f_\theta(x) > \tau\},$$
$$\mathcal{Q}_{\mathrm{TN}}(\tau) = \{x \in \mathcal{Q} : y_x = 0, f_\theta(x) \leq \tau\}.$$

High-score members are answered correctly by the model and contribute no error. Because both the prediction and label are binary,

$$(\widehat{y}_x - y_x)^2 = \mathbf{1}\{\widehat{y}_x \neq y_x\}.$$

Therefore, with $L_{\mathcal{Q}} = \sum_{x \in \mathcal{Q}} (\widehat{y}_x - y_x)^2$, linearity of expectation gives

$$\mathbb{E}[L_{\mathcal{Q}} \mid \tau, s] = \sum_{x \in \mathcal{Q}} \Pr[\widehat{y}_x \neq y_x \mid \tau, s].$$

The third stage retains each unperturbed bit independently with probability

$$p = \frac{e^{\epsilon_b / k}}{e^{\epsilon_b / k} + 1}$$

and flips it with probability $1 - p$. Every member in $\mathcal{Q}_{\mathrm{FN}}(\tau)$ was inserted into the unperturbed filter, so its queried positions are initially one. Under the standard approximation that the k positions are distinct, all of them remain one after randomized response with probability p^k. Hence the approximate false-negative contribution is

$$|\mathcal{Q}_{\mathrm{FN}}(\tau)|(1 - p^k).$$

Every non-member in $\mathcal{Q}_{\mathrm{FP}}(\tau)$ is accepted directly by the learned model and therefore contributes exactly one error. Importantly, these queries do not also enter the Bloom-filter term. Only non-members in $\mathcal{Q}_{\mathrm{TN}}(\tau)$ are sent to the backup filter.

Under the standard random-hashing model, after inserting t elements the marginal probability that an unperturbed bit equals one is

$$1 - \left(1 - \frac{1}{M}\right)^{kt} \approx 1 - e^{-kt/M} =: a.$$

After randomized response, a released bit equals one either because an original one is retained or because an original zero is flipped. Its marginal one probability is therefore

$$r = ap + (1 - a)(1 - p).$$

Approximating the k queried positions as independent gives false-positive probability r^k for each element of $\mathcal{Q}_{\mathrm{TN}}(\tau)$. Combining the three disjoint error cases yields

$$\mathbb{E}[L_{\mathcal{Q}} \mid \tau, s] \approx |\mathcal{Q}_{\mathrm{FN}}(\tau)|(1-p^k) + |\mathcal{Q}_{\mathrm{FP}}(\tau)| + |\mathcal{Q}_{\mathrm{TN}}(\tau)|r^k, \qquad s > 0.$$

If $s = 0$, Algorithm 4 releases the empty-filter representation. All low-score members are then rejected, while all low-score non-members are correctly rejected. Thus this boundary case is exact:

$$\mathbb{E}[L_{\mathcal{Q}} \mid \tau, 0] = |\mathcal{Q}_{\mathrm{FN}}(\tau)| + |\mathcal{Q}_{\mathrm{FP}}(\tau)|.$$

Finally, the law of total expectation averages over the two earlier private releases:

$$\mathbb{E}[L_{\mathcal{Q}}] = \sum_{\tau \in \mathcal{T}} \sum_{s=0}^{B} \Pr[\tau^* = \tau, \widetilde{t} = s]\mathbb{E}[L_{\mathcal{Q}} \mid \tau, s].$$

For $q = |\mathcal{Q}|$, the corresponding expected empirical MSE is $\mathbb{E}[L_{\mathcal{Q}}]/q$. The commonly reported summary $\sqrt{\mathbb{E}[L_{\mathcal{Q}}]/q}$ may be used as an approximate RMSE characterization, but it is not generally equal to $\mathbb{E}[\sqrt{L_{\mathcal{Q}}/q}]$.

The approximation sign in the theorem is therefore deliberate. It arises from the random-hashing occupancy model, the distinct-position assumption for an inserted member, and the independence approximation for queried positions; no finite-sample deterministic inequality is claimed. □

B.6 Full Proof of Theorem 7

Proof. The released DP-LBF state is

$$R = (\tau^*, \widetilde{t}, \widetilde{\boldsymbol{b}}),$$

where the learned model f_θ is trained on public data disjoint from the sensitive set $\mathcal{S}$ and is fixed before $\mathcal{S}$ is accessed. Therefore, the model does not introduce any additional data-dependent storage with respect to $\mathcal{S}$, and it consumes no privacy budget. Its storage cost is simply

$$O(|\theta|).$$

The released threshold τ^* and noisy backup count $\widetilde{t}$ are scalar values and therefore require $O(1)$ words.

It remains to analyze the backup Bloom filter. Importantly, its geometry is determined from the differentially private release $\widetilde{t}$, rather than from the exact internal count

$$t = |\mathcal{D}_{\mathrm{FN}}|.$$

Conditioned on the released value $\widetilde{t}$, the procedure $\mathrm{PublicGeometry}(\widetilde{t}, \rho)$ deterministically selects the Bloom-filter length M, the number of hash functions k, and the first k functions of the fixed public hash family $\mathcal{H}_{\mathrm{pub}}$ according to Equation (1).

For $\widetilde{t} > 0$, the standard Bloom-filter sizing rule for target false-positive rate $\rho \in (0, 1)$ gives

$$M = \Theta\left(\widetilde{t} \log \frac{1}{\rho}\right).$$

The corresponding number of hash functions is $k = k(\widetilde{t}, \rho)$ from Equation (1). Selecting k public hash functions does not change the asymptotic bit length of the released representation. The randomized-response step changes only the values of the M bits and does not change the length of the released bit vector. Hence the backup representation requires

$$O\left(\widetilde{t} \log \frac{1}{\rho}\right)$$

bits.

When $\widetilde{t} = 0$, Algorithm 4 releases the fixed empty backup-filter representation, which requires only $O(1)$ space. Combining the two cases, the backup representation occupies

$$O\left(\max\{1, \widetilde{t}\} \log \frac{1}{\rho}\right)$$

space.

Therefore, including the fixed public model, the total serving-space complexity of DP-LBF is

$$O\left(|\theta| + \max\{1, \widetilde{t}\} \log \frac{1}{\rho}\right).$$

Since NBSR clips the released count to the public interval $[0, B]$, we have $\widetilde{t} \le B$. Consequently, the worst-case space complexity is

$$O\left(|\theta| + B \log \frac{1}{\rho}\right).$$

If only the released private state is counted and the fixed public model is excluded, its representation size is

$$O\left(\max\{1, \widetilde{t}\} \log \frac{1}{\rho}\right),$$

with worst-case size

$$O\left(B \log \frac{1}{\rho}\right).$$

□

B.7 Query Complexity

Theorem 8 (Query Complexity of DP-LBF)**.** *Let $\mathcal{Q}$ contain q query elements, and let T_f be the time required to evaluate f_θ once. The worst-case batch query time is $O(q(T_f + k))$. Conditioned on the released state, if q_{low} queries satisfy $f_\theta(x) \le \tau^*$, the routing-sensitive bound is $O(qT_f + q_{\text{low}}k)$.*

Proof. Fix the released state $R = (\tau^*, \widetilde{t}, \widetilde{\boldsymbol{b}})$ and a query set $\mathcal{Q}$. The three private construction mechanisms have already been executed; a query does not resample the threshold, add Laplace noise, or rerun randomized response.

For each $x \in \mathcal{Q}$, DP-LBF first evaluates $f_\theta(x)$ at cost T_f and compares the resulting score with τ^* at constant cost. Define the routing indicator

$$I_x = \mathbf{1}\{f_\theta(x) \le \tau^*\}.$$

If $I_x = 0$, the model answers the query directly. If $I_x = 1$ and $\widetilde{t} > 0$, the backup-filter lookup evaluates at most k hash functions and inspects at most k bits. Assuming constant-time hash evaluation and bit access, the per-query cost is therefore bounded by

$$O(T_f + 1 + I_x k).$$

When $\widetilde{t} = 0$, the public empty-filter representation returns false in constant time, so the same expression remains a valid upper bound.

Summing over all $q = |\mathcal{Q}|$ queries yields

$$\begin{aligned} T_{\text{query}}(\mathcal{Q} \mid R) &= O\left(\sum_{x \in \mathcal{Q}} (T_f + 1 + I_x k)\right) \\ &= O\left(qT_f + q + k \sum_{x \in \mathcal{Q}} I_x\right). \end{aligned}$$

Let

$$q_{\text{low}} = \sum_{x \in \mathcal{Q}} I_x = |\{x \in \mathcal{Q} : f_\theta(x) \le \tau^*\}|.$$

Absorbing the linear constant-time comparisons into the model-evaluation term gives the routing-sensitive bound

$$T_{\text{query}}(\mathcal{Q} \mid R) = O(qT_f + q_{\text{low}}k).$$

Because $0 \le q_{\text{low}} \le q$, setting $q_{\text{low}} = q$ gives the worst case

$$T_{\text{query}}(\mathcal{Q} \mid R) = O(q(T_f + k)).$$

For an optional workload-average interpretation, suppose the queries are drawn independently from a distribution $\mathcal{P}_Q$ and define, for a fixed released threshold,

$$\pi_{\tau^*} = \Pr_{X \sim \mathcal{P}_Q}[f_\theta(X) \le \tau^*].$$

Then $\mathbb{E}[q_{\text{low}} \mid \tau^*] = q\pi_{\tau^*}$, and hence

$$\mathbb{E}[T_{\text{query}} \mid R] = O(qT_f + q\pi_{\tau^*}k).$$

This last expression is a workload-dependent expectation, not a universal improvement guarantee. In all cases, querying the released structure is post-processing and consumes no additional privacy budget. □

C Experimental Details and Reproducibility

C.1 Dataset

Synthetic. We generate $10{,}000$ records with 20 features from two equally weighted Gaussian components centered at -2 and 2. The first ten features carry the component shift and the remaining ten are nuisance features. To avoid perfect separation, we reverse the labels of 2% of each component while preserving an overall 1:1 class ratio. We use $5{,}000$ records as public data and $5{,}000$ as the private partition; the public portion is further split into model-training and threshold-calibration subsets.

URL[1]**.** We use the dataset-provided URL and webpage features from the PHIUSIIL Phishing URL dataset, which contains $100{,}945$ phishing and $134{,}850$ legitimate URLs before key deduplication. Representative features include CharContinuationRate, URLTitleMatchScore, URLCharProb, and TLDLegitimateProb. After deduplicating URL keys, we assign 50% of the records to the public partition and 50% to the private benchmark partition.

Spambase[2] We use all $4{,}601$ e-mail records and 57 input features. The binary label indicates whether an e-mail is spam. We assign a unique key to each row for membership evaluation; the learned score is computed from the 57 features. As for URL, 50% of the records form the public partition and the remainder form the private benchmark partition. For both real-world datasets, the public partition is further split into disjoint model-training and threshold-calibration subsets, and the balanced query workload is sampled only from the private partition.

C.2 Metric

We evaluate the private membership mechanisms from three aspects: prediction accuracy, query efficiency, and representation size. All measurements are averaged over 100 independent randomized runs. Smaller values are better.

- **Root Mean Square Error (RMSE).** RMSE quantifies the average deviation between the predicted membership outputs $\hat{y}_i$ and the ground truth labels $y_i \in \{0, 1\}$. Let n be the total number of query elements. RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}.$$

[1]The URL dataset is publicly available at `https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset`

[2]The Spambase dataset is publicly available at `https://archive.ics.uci.edu/dataset/94/spambase`.

RMSE measures the total membership-prediction error of the released private mechanism, including model-routing errors and errors introduced by the private set representation.

- **Query Time.** We report amortized wall-clock time per query over a fixed batch. The timed path includes batched model scoring, hash computations, and bit-array lookups, but excludes construction and privatization. This is a throughput-oriented measurement and should not be interpreted as the latency of an isolated online query.
- **Memory Usage.** We report the packed serialized size in KB of each released private membership representation, excluding Python object overhead and runtime memory. For BLIP and DPBloomFilter this is the randomized bit vector; for DPSet it is the finite-field vector and the seeds required for decoding. For Warner-LBF it includes the serialized model trained from randomized membership indicators, the backup-filter bits, and the threshold. For DP-LBF it includes the complete released private state $R = (\tau^*, \widetilde{t}, \widetilde{\boldsymbol{b}})$. The fixed public DP-LBF model and public hash-generation rules are excluded.

C.3 Experimental Setup and Implementation Details

All experiments were conducted on a Windows laptop equipped with an AMD R5-7735H CPU running at 3.20 GHz and 16 GB of RAM. The proposed methods, as well as all existing methods for comparison, were implemented in Python 3.12 and evaluated on a machine running Windows 11. To ensure reproducibility, all experiments were executed under identical software and hardware conditions, and any randomization in the algorithms was controlled via fixed random seeds. We execute all experiments 100 times and take the average results.

We evaluate the five total privacy budgets $\epsilon \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$. DP-LBF uses the fixed allocation $(\epsilon_\tau, \epsilon_m, \epsilon_b) = (0.05\epsilon, 0.05\epsilon, 0.90\epsilon)$, the direct ratio 1:1:18, public FPR-penalty weight $\lambda = 1$, and target Bloom-filter error rate $\rho = 0.01$. The candidate thresholds, public calibration split, allocation, and all other hyperparameters are fixed without inspecting the sensitive member set. LightGBM (Ke et al., 2017) is trained only on the public training split and then held fixed. Baseline-specific public parameters and effective guarantees are stated next.

C.4 Mechanisms for Comparison

We compare DP-LBF with four mechanisms that release reusable private set representations. Each implementation is evaluated on the same sensitive set, query keys, labels, requested privacy budgets, and number of trials. Because the mechanisms have different release interfaces, we state the public parameters and effective guarantees used by each implementation below.

- **BLIP** (Alaggan et al., 2012). BLIP first constructs a standard Bloom filter with k hashes and then independently flips each bit with probability $1/(1 + e^{\epsilon/k})$. We query the released noisy bit array with the same k hash positions used during construction. Its filter geometry is fixed using the benchmark member-set cardinality, which is treated as a public parameter for this fixed-cardinality baseline.
- **DPSet** (Patel et al., 2024). DPSet randomly excludes input members, maps retained members to banded rows over a finite field, solves the resulting linear system, and answers membership by checking the associated linear constraint. We use the paper's expansion parameter $\beta = 0.05$ and a public band width of 384. Finite-field discretization yields effective privacy parameters 0, $\ln 2$, $\ln 4$, $\ln 6$, and $\ln 12$ for the five requested budgets; each is no larger than its requested value. This random-band construction is approximate DP, with the linear-system solve failure probability corresponding to δ. We record every failure; none occurred in the reported 100 runs per setting.
- **DPBloomFilter** (Ke et al., 2025). This method first inserts all elements of the sensitive set $\mathcal{S}$ into a classic Bloom filter whose k hash functions share one m-bit array. Let W be the number of bit positions that differ between neighboring filters and let $N = F_W^{-1}(1 - \delta)$. The method sets $\epsilon_0 = \epsilon/N$ and independently retains each bit with probability $e^{\epsilon_0}/(e^{\epsilon_0} + 1)$ (otherwise flipping it). We use the explicit public setting $\delta = 10^{-5}$. Its filter geometry and the cardinality used to compute N follow the method's fixed-cardinality interface and are treated as public benchmark parameters.

- **Warner-LBF** (Tirmazi, 2025). For every element in the fixed finite universe U, Warner randomized response retains the true membership indicator with probability $p = e^{\epsilon}/(1+e^{\epsilon})$ and flips it otherwise. We take the fixed private benchmark partition as U, train the learned model from the randomized membership indicators, select the LBF threshold, and construct the backup filter from the resulting randomized set. Query time includes model scoring and backup-filter lookup, while representation memory includes both the privately trained model and the backup filter.

C.5 Additional Query-Time Results

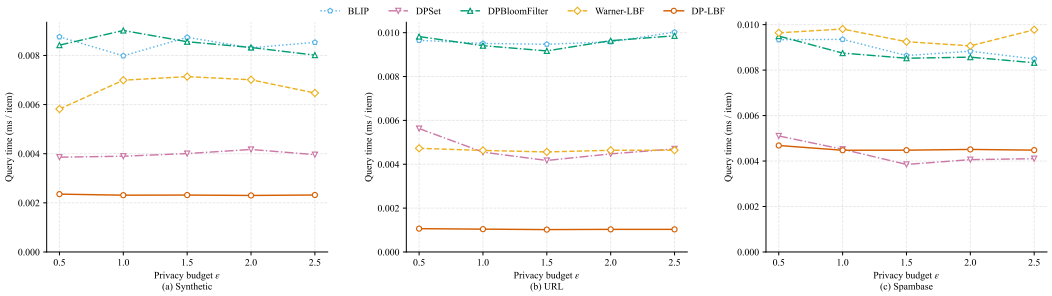


Figure 4: Amortized query time over a fixed query batch for DP-LBF and private baselines.

Figure 4 reports amortized wall-clock time per query over a fixed query batch. The timed path includes batched LightGBM scoring for DP-LBF and on-the-fly query hashing for every method; construction and privatization are excluded. DP-LBF requires 0.00230–0.00235 ms on Synthetic, 0.00102–0.00106 ms on URL, and 0.00447–0.00468 ms on Spambase. Approximately 50.3%, 50.0%, and 52.2–52.6% of the queries on the three datasets, respectively, are routed to the backup filter; high-confidence queries return directly without computing backup-filter hashes. Consequently, DP-LBF is faster than BLIP, DPBloomFilter, and Warner-LBF at every tested budget on all three datasets. It is also faster than the serial DPSet implementation at every budget on Synthetic and URL. On Spambase, DP-LBF is faster at $\epsilon \in \{0.5, 1.0\}$, whereas DPSet is faster at the other three budgets. The small fluctuations across ϵ are implementation-level timing variation. These measurements demonstrate the benefit of conditional backup lookups without claiming a universal query-time advantage.

C.6 Additional Representation-Memory Results

Figure 3 shows that DP-LBF has the smallest released private representation in every tested setting. Its measured size is 0.078–0.079 KB on Synthetic, 0.021–0.036 KB on URL, and 0.108–0.114 KB on Spambase. By comparison, BLIP and DPBloomFilter use 2.964, 80.785, and 1.087 KB on the three datasets, respectively, while DPSet ranges over 0.672–2.641, 15.462–61.801, and 0.318–1.224 KB. Warner-LBF uses 144.234–145.018 KB, 98.126–104.172 KB, and 571.468–583.688 KB, respectively, including its privately trained model and backup filter. DP-LBF is compact because only low-score members are routed to the backup filter, whose geometry is determined from the private noisy count. The reported quantity covers the released private representation only and uses packed serialized sizes; Python object overhead and runtime memory are excluded for every method. The privately trained Warner-LBF model is part of its released private state and is therefore included, whereas the fixed public DP-LBF model is excluded. The measurement should not be interpreted as total system memory.

C.7 Ablation Studies

We conduct three complete ablations to isolate the effects of privacy-budget allocation, the public false-positive penalty, and the target false-positive rate of the backup filter. Every grid is fixed before examining its outcomes. All settings use the same public/private split as the main experiments, $\epsilon = 1.5$, seed 42, and 100 independent randomized releases. Unless the parameter under study is

varied, we use the main-experiment settings: the direct budget ratio 1:1:18, public penalty weight $\lambda = 1$, and backup-filter target FPR $\rho = 0.01$.

Privacy-budget allocation. Figure 5 reports the complete grid $\{1{:}1{:}1, 1{:}1{:}3, 1{:}1{:}8, 1{:}1{:}18\}$ for the threshold, noisy-count, and bit-randomization stages. The effect is dataset dependent. RMSE varies only modestly on Synthetic and Spambase, whereas URL favors a more balanced allocation. Thus, no ratio is universally optimal; importantly, the ratio in the main comparison is fixed globally rather than tuned using the sensitive set or per-dataset test outcomes.

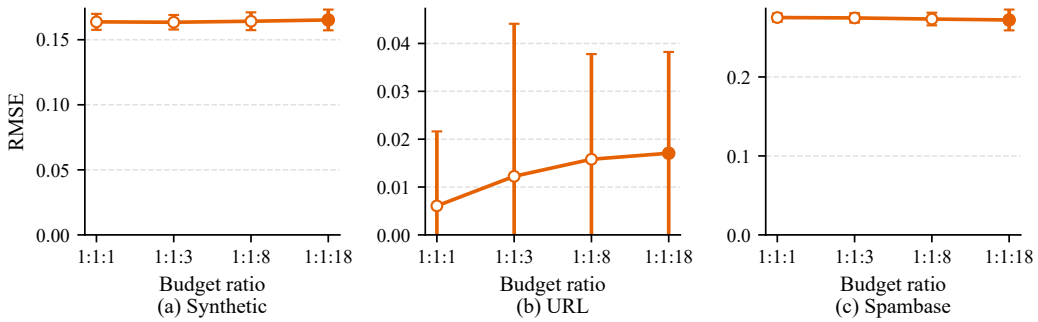


Figure 5: RMSE under four complete allocations of the three-stage privacy budget; error bars show one standard deviation and the filled marker denotes the main-experiment setting.

Public false-positive penalty. Figure 6 varies the public penalty weight over $\lambda \in \{0, 0.25, 0.5, 1, 2\}$. On URL, omitting the penalty increases mean RMSE from approximately 0.017 to 0.132, while every positive weight in the grid prevents this collapse. Synthetic is insensitive across the tested range. Spambase exhibits the expected FPR–FNR trade-off: a moderate weight $\lambda = 0.25$ gives the lowest RMSE, while larger weights increasingly favor false-positive suppression. These results support using a penalty computed entirely from public calibration data, while also showing that it is not a privacy-dependent tuning parameter.

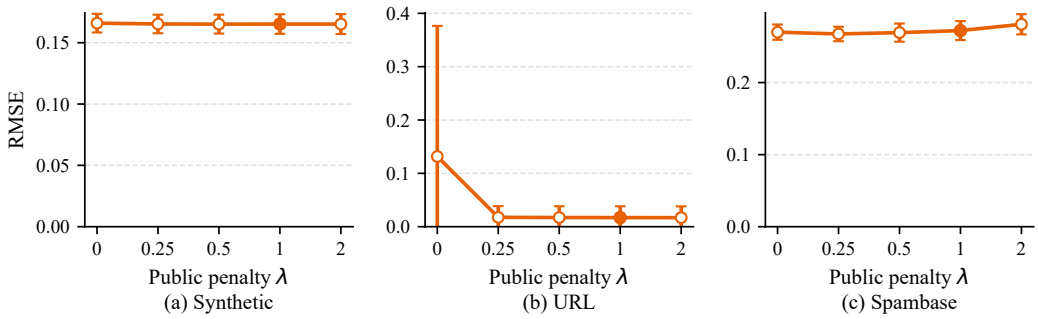


Figure 6: RMSE as the public calibration penalty weight varies; error bars show one standard deviation and the filled marker denotes the main-experiment setting.

Backup-filter target FPR. Figure 7 varies $\rho \in \{0.1, 0.05, 0.01, 0.005\}$. Tightening the target FPR consistently reduces mean RMSE on all three datasets: from 0.237 to 0.159 on Synthetic, from 0.049 to 0.013 on URL, and from 0.315 to 0.270 on Spambase. The improvement requires only a small representation increase over the same grid, from 0.111 to 0.151 KB, 0.039 to 0.044 KB, and 0.127 to 0.188 KB, respectively. This ablation exposes the accuracy–space trade-off controlled by public geometry rather than by additional access to the sensitive set.

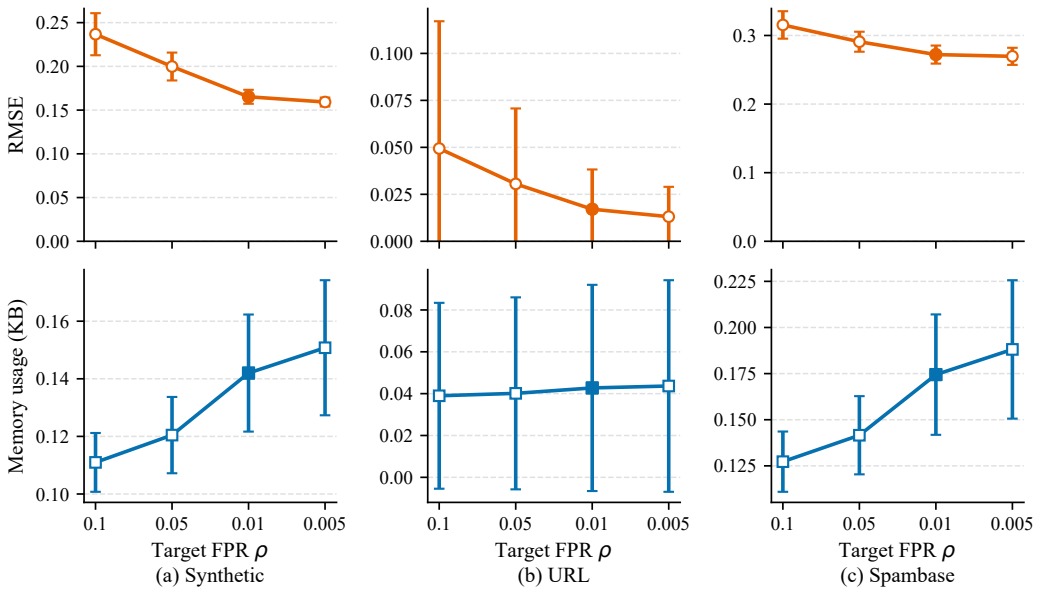


Figure 7: RMSE and released-representation memory as the backup-filter target FPR varies; error bars show one standard deviation and the filled marker denotes the main-experiment setting.