# Mathematical Foundations of Poisoning Attacks on Linear Regression over Cumulative Distribution Functions

ATSUKI SATO, The University of Tokyo, Japan
MARTIN AUMÜLLER, IT University of Copenhagen, Denmark
YUSUKE MATSUI, The University of Tokyo, Japan

Learned indexes are a class of index data structures that enable fast search by approximating the cumulative distribution function (CDF) using machine learning models (Kraska et al., SIGMOD'18). However, recent studies have shown that learned indexes are vulnerable to poisoning attacks, where injecting a small number of poison keys into the training data can significantly degrade model accuracy and reduce index performance (Kornaropoulos et al., SIGMOD'22). In this work, we provide a rigorous theoretical analysis of poisoning attacks targeting linear regression models over CDFs, one of the most basic regression models and a core component in many advanced learned indexes. Our main contributions are as follows: (i) We present a theoretical proof characterizing the optimal single-point poisoning attack and show that the existing method yields the optimal attack. (ii) We show that in multi-point attacks, the existing greedy approach is not always optimal, and we rigorously derive the key properties that an optimal attack should satisfy. (iii) We propose a method to compute an upper bound of the multi-point poisoning attack's impact and empirically demonstrate that the loss under the greedy approach is often close to this bound. Our study deepens the theoretical understanding of attack strategies against linear regression models on CDFs and provides a foundation for the theoretical evaluation of attacks and defenses on learned indexes.

CCS Concepts: • **Information systems** → **Data structures**; • **Security and privacy** → *Cryptanalysis and other attacks*; • **Computing methodologies** → Machine learning approaches.

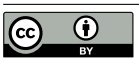
Additional Key Words and Phrases: Learned Systems, Data Poisoning, Attacks, Indexing

**ACM Reference Format:**
Atsuki Sato, Martin Aumüller, and Yusuke Matsui. 2026. Mathematical Foundations of Poisoning Attacks on Linear Regression over Cumulative Distribution Functions. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 208 (June 2026), 25 pages. https://doi.org/10.1145/3802085

## 1 Introduction

In recent years, learned indexes have attracted significant attention as a promising alternative to traditional index structures [17, 22, 29, 36, 54, 61]. One representative example of a learned index is the learned B-tree [17, 29, 52], which approximates the cumulative distribution function (CDF) using regression models. It achieves superior memory efficiency and faster query performance compared to conventional methods. During query processing, the learned B-tree first uses a regression model to predict the position of the query key, and then corrects the prediction error using local search algorithms such as exponential search. Smaller prediction errors lead to narrower search ranges and faster query execution, whereas larger errors increase the cost of local search.

Authors' Contact Information: Atsuki Sato, a_sato@hal.t.u-tokyo.ac.jp, The University of Tokyo, Tokyo, Japan; Martin Aumüller, maau@itu.dk, IT University of Copenhagen, Copenhagen, Denmark; Yusuke Matsui, matsui@hal.t.u-tokyo.ac.jp, The University of Tokyo, Tokyo, Japan.




ACM 2836-6573/2026/6-ART208
https://doi.org/10.1145/3802085

Table 1. Overview over time complexity and empirical performance of poisoning attacks and upper-bounding methods under the original setting [28] (duplicate keys disallowed; Definition 2) and the relaxed setting (duplicate keys allowed; Definition 3). Here, $n$ is the number of legitimate keys, $\lambda$ is the poisoning budget, and $T$ is the number of iterations in our upper-bounding methods. *Empirical Perf. Ratio* is the achieved MSE (attacks) or the bound (upper bounds) normalized by the optimal-attack MSE over 3,000 cases in Section 6.6 (values closer to 1 indicate a better attack or a tighter upper bound).

| | Original Setting (Definition 2) | | Relaxed Setting (Definition 3) | |
|---|---|---|---|---|
| Algorithm | Time Complexity | Empirical Perf. Ratio | Time Complexity | Empirical Perf. Ratio |
| Greedy [28] | $O(n\lambda)$ | $\geq 0.933$ | - | - |
| Exact Seg+E (Ours) | $O(n\lambda^3)$ | 1 | $O(n\lambda)$ | 1 |
| Heuristic Seg+E (Ours) | $O(n\lambda)$ | $\geq 0.998$ | - | - |
| Optimal (Ours) | $O\left((n+\lambda)\binom{2n-2+\lambda}{\lambda}\right)$ | 1 | $O\left((n+\lambda)\binom{n+\lambda-1}{\lambda}\right)$ | 1 |
| Upper Bound (Ours) | $O(T(n+\lambda))$ or $O((n+\lambda)\log(n+\lambda))$ | $\leq 1.092$ | $O(T(n+\lambda))$ or $O((n+\lambda)\log(n+\lambda))$ | $\leq 1.087$ |

While learned indexes achieve outstanding performance on typical datasets, recent studies have revealed that they are vulnerable to poisoning attacks [14, 28]. Specifically, by inserting just a small number of malicious data keys (poisons) into the legitimate training data used to build the index, an attacker can deliberately degrade the model's prediction accuracy. The increase in prediction error leads to higher costs for subsequent local searches, thereby degrading the overall performance of the learned index.

However, existing attack methods proposed in the literature are largely heuristic [48, 62]. Even in the most basic poisoning setting—where a linear regression model is trained to minimize the mean squared error (MSE) and the attacker aims to maximize it—the optimality of the existing attack method [28] remains unclear. This leaves several fundamental questions open:

- Which structures characterize an optimal attack?
- Is the existing algorithm from [28] an optimal attack?
- If not, can we derive a provable and tight upper bound on the maximum impact caused by any attack?

We resolve these questions and establish a theoretical foundation for attack strategies in the same MSE-based linear-regression poisoning setting studied in prior work [28]. Our contributions are as follows:

- In single-point attack scenarios (i.e., where only one poison point is injected), we provide the first formal proof that placing the poison adjacent to a legitimate key is optimal, as conjectured in [28]. Thereby, we show that the existing single-point attack method yields an optimal attack.
- In multi-point attack scenarios, we show that the iterative greedy method of [28] does *not* in general exactly match the optimal solution; we provide instances where it is strictly suboptimal, thereby refuting the implicit assumption in prior work [28] that the greedy method is always optimal.
- In multi-point attack scenarios, we derive the key properties of an optimal attack; we prove that every poison included in an optimal attack is adjacent to a legitimate key, either directly or indirectly through other poison points. This makes it possible to compute the exact optimal solution in realistic time for small-scale settings.
- In multi-point attack scenarios, we propose a computationally efficient method that establishes a rigorous upper bound on the achievable attack impact, i.e., a provable ceiling that no attack

can exceed. This upper bound quantifies how close the greedy algorithm is to optimal, and also offers worst-case guarantees for the linear regression model under adversarial settings.
- In multi-point attack scenarios, we experimentally show that the greedy solution is often close to our upper bound: over 3,000 cases in Section 6.2, the bound is at most 1.25× the greedy MSE and 1.03× on average. This suggests that the bound is tight and that greedy attacks are near-optimal.
- In the multi-point attack setting, we identified a simple structural property that frequently appears in optimal solutions, which we call *Segment + Endpoint (Seg+E)*, and we provide efficient algorithms to find the Seg+E solution. We experimentally demonstrate that Seg+E consistently yields a larger loss than the greedy approach.

All six contributions are original to our paper (i.e., they do not appear in prior work, including [28]), and require nontrivial new technical or experimental insights. These results provide the first theoretical framework for understanding attack strategies against linear regression models trained on CDFs, offering a foundation for analyzing attacks and defenses on learned indexes.

*Summary of Algorithms.* Table 1 summarizes the algorithms studied in this work. Here, $n$ is the number of legitimate keys, $\lambda$ is the poisoning budget, and $T$ is the number of iterations used by our upper-bound search procedures. The column *Empirical Perf. Ratio* reports the achieved MSE or upper bound, divided by the optimal-attack MSE, over 3,000 cases in Section 6.6 (values closer to 1 indicate better attack or tighter upper bound). In addition to the original setting [28], which does not allow duplicate keys (Definition 2), we also define a relaxed setting that allows duplicate keys (Definition 3). We design, for both settings, methods to compute the optimum, upper bounds, and Seg+E attacks. Empirically, we show that Greedy [28] is occasionally suboptimal, whereas our heuristic Seg+E is closer to optimal, and Exact Seg+E matches the optimum in all 3,000 instances under both settings. Moreover, our upper-bound computation is fast and yields tight bounds (at most 1.092× the optimal-attack MSE).

This paper is organized as follows. Section 2 provides an overview of related work, and Section 3 formally introduces the problem setting and known attacks. Section 4 and Section 5 present our theoretical results for single- and multi-point attacks, respectively. Section 6 reports our experimental validation. Section 7 provides further discussion, Section 8 outlines limitations and directions for future research, and Section 9 concludes the paper.

## 2 Related Work

In this section, we first provide a comprehensive overview of learned indexes, followed by a review of attacks on them.

### 2.1 Learned Indexes

Indexing is a fundamental technology for enabling efficient data access and has been applied across a wide range of scenarios, including databases [44], search engines [49], file systems [20], and similarity search systems [3, 4]. Classic examples include B-trees [5], hash maps [27], and Bloom filters [6]. These data structures have been extensively studied for decades as critical components that determine the overall performance of systems where efficient data access is essential.

Recently, learned indexes, which leverage machine learning to improve index performance, have been proposed and have attracted considerable attention [29]. Learned indexes replace or augment traditional data structures with machine learning models, aiming to improve memory efficiency and query speed. Various learned index architectures have been proposed, extending classical data structures such as Bloom filters [10, 38, 47, 54], count-min sketches [13, 22, 41, 67], and LSM-trees [9, 35, 39, 61] using machine learning techniques.

Among these, learned B-trees, which enhance B-trees using machine learning, are the most actively studied type of learned index and are often referred to simply as "learned indexes" [2, 11, 17, 19, 52, 58, 60]. The core idea of learned indexes is to approximate the cumulative distribution function (CDF) of the data using machine learning-based regression. During query processing, the model is used to infer an approximate position of the query key. This prediction is then refined using local search techniques such as exponential search to identify the exact position. Beyond the original one-dimensional, static setting [2, 26, 29, 37, 60], subsequent work has extended learned indexes to multidimensional data [1, 12, 21, 32, 40, 43, 55] and string data [25, 51, 58, 63, 70], as well as to dynamic settings that support updates [11, 17, 19, 30, 31, 34, 53, 57, 59]. In addition, there has been increasing interest in theoretical analyses of worst-case/expected complexities in construction/query [8, 16, 33, 64, 65].

A common structural characteristic shared by many learned indexes is their hierarchical, Mixture-of-Experts-style design [52]. That is, a root model routes queries to localized leaf models, each approximating the CDF of a contiguous subset of sorted keys. Linear regression models are often used as leaf models [11, 30, 52, 58] because they provide sufficient accuracy for small subdatasets, avoiding the computational overhead of complex models.

### 2.2 Attacks on a Learned Index

While learned indexes often offer excellent performance on typical datasets, recent research has revealed that they also exhibit vulnerabilities to various attacks. Attacks on learned Bloom filters include poisoning attacks [14], side-channel attacks [15], and mutation attacks [45]. Additional studies demonstrate attacks on learning-based sketches [24], learned cardinality estimators [66], and learned index advisors [68, 69]. The root of these vulnerabilities lies in two key factors: (i) the increased information flow introduced by fitting models to past data distributions, and (ii) the degradation of worst-case performance caused by optimizing machine learning models for average-case scenarios [48].

Among these, attacks on learned indexes (in the narrow sense) have been particularly actively studied in recent years [28, 48, 62]. Kornaropoulos et al. [28] proposed poisoning attacks against linear regression models trained on CDFs, and further demonstrated attacks on RMI architectures using such poisoning methods. However, the theoretical foundations of these attacks remained unresolved, and addressing this gap is the focus of our study.

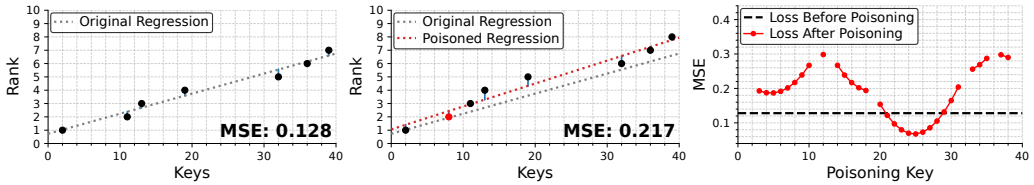
Finally, we briefly review other types of attacks on learned B-trees. For ALEX [11], one of the most prominent dynamic learned indexes, algorithmic complexity attacks (ACAs) have been proposed in [48, 62]. These attacks force the system into exponential time worst-case behavior. Additionally, even the learned indexes with guaranteed worst-case complexity (e.g., the PGM-index [17]) are shown to be vulnerable to timing-channel-based attacks [48]. By measuring and analyzing query latency, an attacker may infer information about the underlying data distribution that should be inaccessible. Such attacks are not the focus of this work.

## 3 Problem Statement and Existing Attacks

In this section, we first define the problem setting of poisoning attacks against linear regression models trained on CDFs (Section 3.1), followed by an overview of existing attack methods (Section 3.2).

### 3.1 Problem Statement

Throughout this paper, we let $\mathcal{K}$ denote the set of *legitimate keys*, i.e., the keys originally present in the training data before any attack, and $n \coloneqq |\mathcal{K}|$. We use $\mathcal{X}$ to denote a general key set, which may include both legitimate and poison keys, and $m \coloneqq |\mathcal{X}|$.




(a) Regression Before Poisoning. (b) Regression After Poisoning. (c) Loss Function.

Fig. 1. Single-point poisoning attack on $\mathcal{K} = \{2, 11, 13, 19, 32, 36, 39\}$. By inserting a poison point, the rank of every key greater than the poison increases by one. The linear regression model is fitted on the poisoned key set, leading to a change in MSE.

We begin by defining linear regression on CDFs similarly to [28].

Definition 1 (**Linear Regression on CDFs [28, Def. 1]**). *Let $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ be a multiset of natural numbers representing the keys stored in the index, where $x_1 \leq x_2 \leq \cdots \leq x_m$ and at least two distinct values appear in $\mathcal{X}$ (i.e., $x_1 \neq x_m$). Define the rank associated with $x_i$ as $r_i := i$. The MSE of fitting a linear regression model with parameters $w, b \in \mathbb{R}$ is defined as*

$$\mathcal{L}(\mathcal{X}; w, b) := \frac{1}{m} \sum_{i=1}^{m} (wx_i + b - r_i)^2 . \tag{1}$$

*The problem of linear regression on CDFs is to find the model parameters that minimize the MSE, that is, to determine*

$$E(\mathcal{X}) := \min_{w,b} \mathcal{L}(\mathcal{X}; w, b). \tag{2}$$

For convenience, define $\mathrm{Cov_{XR}}, \mathrm{Var_X}, \mathrm{Var_R}, \bar{x}$, and $\bar{r}$ as follows:

$$\mathrm{Cov_{XR}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{x})(r_i - \bar{r}), \quad \mathrm{Var_X} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{x})^2,$$
$$\mathrm{Var_R} = \frac{1}{m}\sum_{i=1}^{m}(r_i - \bar{r})^2, \quad \bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i, \quad \bar{r} = \frac{1}{m}\sum_{i=1}^{m} r_i. \tag{3}$$

Then, this problem has the following closed-form solution [28]:

$$w^* = \frac{\mathrm{Cov_{XR}}}{\mathrm{Var_X}}, \quad b^* = \bar{r} - w^*\bar{x}, \quad E(\mathcal{X}) = \mathrm{Var_R} - \frac{{\mathrm{Cov_{XR}}}^2}{\mathrm{Var_X}}. \tag{4}$$

Following prior work [28], we focus on integer keys for simplicity; extensions to other key types are discussed in Section 7.

We address the following poisoning attack problem, which is the same as the one studied in [28]:

Definition 2 (**Poisoning Linear Regression on CDFs [28, Def. 2]**). *Let $\mathcal{K} = \{k_1, k_2, \ldots, k_n\} \subset \mathbb{N}$ be $n$ distinct legitimate keys (i.e., the keys in the original training data before any attack), sorted so that $k_1 < k_2 < \cdots < k_n$. Let $\lambda \in \mathbb{N}$ be the maximum number of poison points. The problem of poisoning linear regression on CDFs is to find a set $\mathcal{P}^* \subset \mathbb{N}$ satisfying $|\mathcal{P}^*| \leq \lambda$ and $\mathcal{P}^* \subset \{k_1 + 1, k_1 + 2, \ldots, k_n - 1\} \setminus \mathcal{K}$ that maximizes $E(\mathcal{K} \cup \mathcal{P}^*)$.*

**Algorithm 1** Single-point Poisoning Attack [28]

**Require:** Legitimate Keys: $\mathcal{K}$

**function** SINGLE-POINTPOISONINGATTACK($\mathcal{K}$)
  $k_{\min} \leftarrow \min(\mathcal{K}), \quad k_{\max} \leftarrow \max(\mathcal{K})$
  $\mathcal{P}_{\text{Cands}} \leftarrow \{k+1 \mid k \in \mathcal{K} \setminus \{k_{\max}\}\} \cup \{k-1 \mid k \in \mathcal{K} \setminus \{k_{\min}\}\}$
  $\mathcal{P}_{\text{Cands}} \leftarrow \mathcal{P}_{\text{Cands}} \setminus \mathcal{K}$
  **if** $\mathcal{P}_{\text{Cands}} = \varnothing$ **then return** None
  $p \leftarrow$ SELECTMAXLOSSCANDIDATE($\mathcal{K}, \mathcal{P}_{\text{Cands}}$)
  **if** $E(\mathcal{K} \cup \{p\}) < E(\mathcal{K})$ **then return** None
  **return** $p$

*Formally,*

$$\mathcal{P}^* := \underset{\mathcal{P} \text{ s.t. } |\mathcal{P}| \le \lambda,\ \mathcal{P} \subset \{k_1+1, k_1+2, \ldots, k_n-1\} \setminus \mathcal{K}}{\operatorname{argmax}} E(\mathcal{K} \cup \mathcal{P}). \tag{5}$$

*Threat Model.* We adopt the same threat model as in [28]. The attacker has full knowledge of the legitimate key set $\mathcal{K}$ and can inject arbitrarily chosen poison keys into the training data, subject only to the constraint in Equation (5). Among common threat models, this setting gives the attacker the strongest information and capabilities, and thus represents the most impactful attack scenario.

As in [28], we restrict poisons to lie strictly between the minimum and maximum legitimate keys to avoid trivial outlier detection and removal. If there are no unused integer values between $k_1$ and $k_n$ (i.e., $\mathcal{K} = \{k_1, k_1+1, \ldots, k_n\}$), then no poisons can be injected. When $\lambda = 1$, the problem is referred to as a *single-point poisoning attack*; when $\lambda \ge 2$, it is called a *multi-point poisoning attack*.

Figure 1 illustrates an example of a single-point attack. Figure 1a shows the legitimate keys and the linear regression model fitted to their CDF. Figure 1b shows how the regression line is distorted when a poison point $\mathcal{P} = \{8\}$ is inserted. A key characteristic of poisoning attacks against linear regression on CDFs is that the insertion of poison points alters the ranks of all keys following the poisons. For example, in Figures 1a and 1b, inserting the poison point 8 does not affect the ranks of keys smaller than 8, but increases the rank of each key larger than 8 by one. This rank-shifting effect is one of the main factors that makes the mathematical analysis of such attacks challenging.

### 3.2 Existing Attacks

*Single-point poisoning attack.* We first recap the single-point poisoning attack strategy proposed by [28]. Importantly, the following statement is an *empirical observation* reported in prior work and is *not* one of our contributions:

OBSERVATION 1 (**EMPIRICAL OBSERVATION IN [28]**). *The optimal single-point attack $\mathcal{P}^*$ is either the empty set or a singleton containing an integer adjacent to a legitimate key;*

$$\mathcal{P}^* \subset \{k+1 \mid k \in \mathcal{K}\} \cup \{k-1 \mid k \in \mathcal{K}\}. \tag{6}$$

Figure 1 presents an example illustrating this observation. Figure 1c shows a plot of $E(\mathcal{K} \cup \{p\})$ for all possible poison points $p$. Note that integers already occupied by legitimate keys are excluded from the plot, as they cannot serve as poison keys. In this example, $E(\mathcal{K} \cup \{p\})$ forms a convex shape within each interval between two consecutive legitimate keys. Therefore, the poison point

**Algorithm 2** Greedy Multi-point Poisoning Attack [28]

**Require:** Legitimate Keys: $\mathcal{K}$, Maximum Number of Poisons: $\lambda$

**function** MULTI-POINTPOISONINGATTACK($\mathcal{K}, \lambda$)
  $\mathcal{P} \leftarrow \emptyset$
  **while** $|\mathcal{P}| < \lambda$ **do**
    $p \leftarrow$ SINGLE-POINTPOISONINGATTACK($\mathcal{K} \cup \mathcal{P}$)
    **if** $p$ = None **then break**
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{p\}$
  **return** $\mathcal{P}$

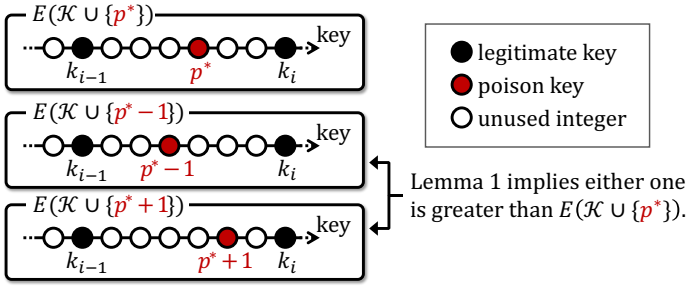



Fig. 2. Proof of Theorem 1. Lemma 1 implies that either $E(\mathcal{K} \cup \{p^* - 1\})$ or $E(\mathcal{K} \cup \{p^* + 1\})$ is greater than $E(\mathcal{K} \cup \{p^*\})$.

$p$ that maximizes the loss in each interval lies at either endpoint of the interval. Specifically, the optimal single-point attack in this case is $\mathcal{P} = \{12\}$, which aligns with Observation 1.

Based on this observation, [28] proposed the following single-point attack algorithm: exhaustively evaluating $E(\mathcal{K} \cup \{p\})$ for every candidate poison point $p$ (i.e., integers adjacent to legitimate keys). The algorithm is summarized in Algorithm 1. By employing differential calculation, this evaluation can be performed in time $O(n)$. However, [28] did not prove the optimality of this algorithm.

*Multi-point poisoning attack.* For multi-point attacks, [28] iteratively applies the above single-point attack $\lambda$ times, see Algorithm 2. The time complexity of this multi-point poisoning algorithm is $O((n + \lambda)\lambda)$. Empirically, they report that this greedy approach results in an optimal attack, again missing a formal proof.

## 4 Optimality of Single-Point Poisoning Attack

We theoretically guarantee the optimality of the single-point attack proposed in [28] by proving the following theorem.

THEOREM 1. *Observation 1 always holds. In other words, the optimal single-point attack $\mathcal{P}^*$ satisfies:*

$$\mathcal{P}^* \subset \{k + 1 \mid k \in \mathcal{K}\} \cup \{k - 1 \mid k \in \mathcal{K}\}. \tag{7}$$

To prove Theorem 1, we first establish the following key lemma.

Lemma 1. *Let $\mathcal{X}$ be the set of keys stored in the index, with $m = |\mathcal{X}| \geq 3$, and let $x_1 \leq x_2 \leq \cdots \leq x_m$ denote the elements of $\mathcal{X}$ in increasing order. Then, for each $i \in \{2, 3, \ldots, m-1\}$, $\operatorname{sign}\left(\frac{dE(\mathcal{X})}{dx_i}\right)$ is non-decreasing over the interval $x_i \in (x_{i-1}, x_{i+1})$, and there exists at most one value of $x_i$ in this interval for which $\frac{dE(\mathcal{X})}{dx_i} = 0$. Here, the function $\operatorname{sign}(x)$ is defined as $-1$ if $x < 0$, $0$ if $x = 0$, and $1$ if $x > 0$.*

Proof sketch of Lemma 1. Computing $\operatorname{sign}(dE(\mathcal{X})/dx_i)$ shows that it equals the sign of the covariance between $\boldsymbol{x}'$ and $\boldsymbol{r}'$, where $\boldsymbol{x}'$ and $\boldsymbol{r}'$ denote the vectors obtained from $\boldsymbol{x}$ and $\boldsymbol{r}$, respectively, by removing their $i$-th elements (i.e., $x_i$ and $i$). Since both $\boldsymbol{x}'$ and $\boldsymbol{r}'$ are monotonically increasing, the sign of covariance is positive. □

We defer the full, formal proof to Appendix A of the full version of this paper [46]. Using Lemma 1, we can prove Theorem 1 as follows.

Proof of Theorem 1. To facilitate comprehension of the proof structure, an illustrative figure is presented in Figure 2. Assume, for the sake of contradiction, that there exists an optimal single-point attack $\mathcal{P}^* = \{p^*\}$ such that $p^* \notin \{k+1 \mid k \in \mathcal{K}\} \cup \{k-1 \mid k \in \mathcal{K}\}$.

Let $i \in \{2, 3, \ldots, n\}$ be the unique index satisfying $k_{i-1} < p^* < k_i$. Under our assumption, the integers adjacent to $p^*$ are not legitimate keys, meaning $k_{i-1} < p^* - 1$ and $p^* + 1 < k_i$ (see Figure 2). Since $p^*$ is assumed to be an optimal single poison point, we have:

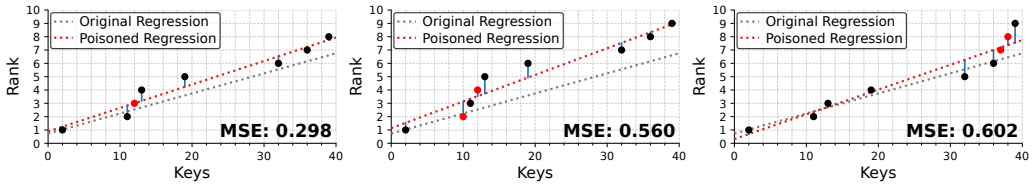
$$E(\mathcal{K} \cup \{p^*-1\}) \leq E(\mathcal{K} \cup \{p^*\}) \ \wedge \ E(\mathcal{K} \cup \{p^*\}) \geq E(\mathcal{K} \cup \{p^*+1\}). \tag{8}$$

According to Lemma 1, the function $\operatorname{sign}\left(\frac{dE(\mathcal{K}\cup\{p\})}{dp}\right)$ is non-decreasing over the interval $p \in (k_{i-1}, k_i)$, and there exists at most one point $p$ in this interval where $\frac{dE(\mathcal{K}\cup\{p\})}{dp} = 0$. Therefore, the behavior of $E(\mathcal{K} \cup \{p\})$ over $p \in (k_{i-1}, k_i)$ must fall into one of the following three cases: (i) strictly increasing, (ii) strictly decreasing, or (iii) strictly decreasing over $(k_{i-1}, p')$ and strictly increasing over $(p', k_i)$ for some $p' \in (k_{i-1}, k_i)$. In all cases, either $E(\mathcal{K} \cup \{p^* - 1\}) > E(\mathcal{K} \cup \{p^*\})$ or $E(\mathcal{K} \cup \{p^*\}) < E(\mathcal{K} \cup \{p^* + 1\})$. This contradicts the assumption that $p^*$ is optimal. □

By Theorem 1, we have established that the single-point attack proposed in [28] is guaranteed to find the optimal solution. In other words, by exhaustively searching only the integers adjacent to legitimate keys, one can identify the optimal single-point attack.

## 5 Multi-Point Poisoning Attack

In this section, we first demonstrate that the iterative greedy algorithm (Algorithm 2) proposed in [28] is not optimal, and we then describe the key properties that can be rigorously asserted about the structure of the optimal solution (Section 5.1). Next, leveraging these properties, we propose a method to upper-bound the loss incurred by multi-point attacks (Section 5.2). Finally, motivated by the structural insights described above, we introduce the Segment + Endpoint (Seg+E) class of poison sets and provide efficient exact and heuristic algorithms to find Seg+E solutions (Section 5.3).




(a) Optimal 1-point Poisoning. (b) Greedy 2-point Poisoning. (c) Optimal 2-point Poisoning.

Fig. 3. Multi-point poisoning attack. The greedy two-point attack [28] injects the poison point 12 first (Figure 3a), followed by the injection of the poison point 10 (Figure 3b). In contrast, the optimal attack is $\{37, 38\}$, resulting in a higher MSE (Figure 3c).

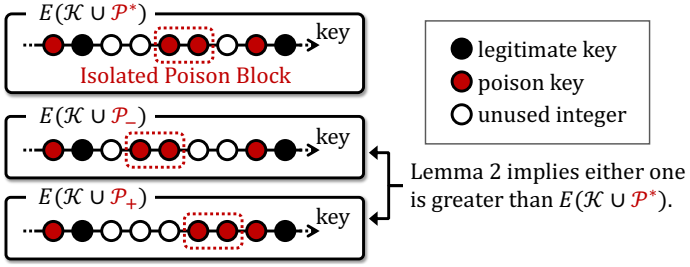



Fig. 4. Proof of Theorem 2. Lemma 2 implies that $E(\mathcal{K} \cup \mathcal{P}_-)$ or $E(\mathcal{K} \cup \mathcal{P}_+)$ is greater than $E(\mathcal{K} \cup \mathcal{P}^*)$.

### 5.1 Structure of Optimal Multi-Point Attacks

Observation 2. *The iterative greedy approach (Algorithm 2) does not guarantee an optimal multi-point poisoning attack.*

Figure 3 provides an illustrative example where the greedy approach does not lead to the optimal solution. As shown in the previous section, for the single-point attack case, the optimal solution can be found by exhaustively evaluating all integers adjacent to the legitimate keys (Figure 3a). The greedy poisoning method of [28] performs a two-point attack by adding one additional optimal poison key on top of this (Figure 3b). However, the true optimal two-point attack involves adding poison keys as shown in Figure 3c. Thus, the greedy poisoning method does not always yield the optimal solution. This finding is valuable for preventing confusion among future researchers, as [28] reports the following: "*Even though we do not provide a proof of optimality for the multiple-point poisoning, we experimentally observed that our approach matched the performance of the brute-force attack in every tested dataset.*"

Next, we establish the following theorem, which characterizes the structure of the optimal solution for multi-point attacks. This result can be seen as an extension of Theorem 1.

Theorem 2. *The optimal multi-point attack consists only of poison keys that are either adjacent to legitimate keys directly or connected transitively to legitimate keys via chains of neighboring poison keys. Formally, the optimal multi-point attack $\mathcal{P}^*$ satisfies:*

$$\forall p \in \mathcal{P}^*,\ \exists k \in \mathcal{K},\ \{\min(p,k)+1, \ldots, \max(p,k)-1\} \subset \mathcal{P}^*. \tag{9}$$

Proof sketch of Theorem 2. To facilitate comprehension of the proof structure, an illustrative figure is presented in Figure 4. First, we prove an extension of Lemma 1, denoted Lemma 2 (in Appendix B of the full version [46]), which implies that if there exists an isolated block of consecutive poison keys (an *Isolated Poison Block*), then moving that block either left or right can strictly increase the loss. Applying the same argument as in Theorem 1, but with an Isolated Poison Block instead of a single poison key, yields Theorem 2. □

We defer the full, formal proof to Appendix B of the full version [46]. An immediate consequence of this theorem is that we can find an optimal solution by examining at most $\binom{2n-2+\lambda}{\lambda}$ candidate multi-point attacks. This follows because the number of candidate optimal multi-point attacks is bounded by the number of ways to distribute the $\lambda$ units of poison among $2n-1$ groups (corresponding to poison placed to the right of $k_1$, to the left of $k_2$, to the right of $k_2$, ..., to the left of $k_n$, and the unused portion of the budget). Since the MSE for each poisoning configuration can be computed in $O(n+\lambda)$ time, the overall running time to find the optimum is $O\left((n+\lambda)\binom{2n-2+\lambda}{\lambda}\right)$, as summarized in the Optimal entry of Table 1. Without our structural result, one would have to examine all combinations of integers in the domain that are not occupied by legitimate keys. The number of such combinations is $O\left(\binom{k_n-k_1}{\lambda}\right)$, which is computationally infeasible in practice because $k_n - k_1$ is typically at least $10^3$ and can easily exceed $10^9$. Our theorem drastically shrinks the search space, making it realistic to compute the optimal multi-point attack for moderately small values of $n$ and $\lambda$.

Next, we highlight two potential misconceptions around the structure of optimal multi-point attacks. The first potential misconception is that Theorem 1 trivially implies Theorem 2. For the sake of an example, let the legitimate key set be $\mathcal{K} = \{1, 5\}$. While Theorem 2 allows us to rigorously conclude that $\mathcal{P} = \{3, 4\}$ is not an optimal solution, the statement of Theorem 1 alone does not allow us to rule out the possibility that $\mathcal{P} = \{3, 4\}$ could be the optimal two-point attack. Although $\mathcal{P} = \{3\}$ or $\mathcal{P} = \{4\}$ are not optimal in the single-point case, Theorem 1 does not provide any guarantee about whether $\mathcal{P} = \{3, 4\}$ could become optimal in the multi-point case. Only with Theorem 2 can we rigorously conclude that $\mathcal{P} = \{3, 4\}$ is not the optimal solution.

The second potential misconception is that Theorem 2 serves as a proof of the optimality of the greedy approach. This is also a misconception, which we demonstrated in Figure 3, where the greedy poisoning method is not optimal. Theorem 2 merely shows that the solution obtained by the greedy poisoning method **can be** an optimal solution in the sense that all poison keys are directly or indirectly adjacent to some legitimate key. It does not claim that the greedy poisoning method **is guaranteed to be** optimal.

### 5.2 Upper Bound on Impact of Optimal Multi-Point Poisoning Attacks

Here, we propose methods to upper-bound the impact of multi-point attacks. The resulting upper bound can be computed efficiently: either in $O(T(n+\lambda))$ time using an iterative method, where $T$ is the number of iterations, or in $O((n+\lambda)\log(n+\lambda))$ time using an exact method. This upper bound is important for both attackers and defenders: it quantifies the maximum possible improvement over existing attacks and provides a worst-case performance guarantee for linear regression (see Section 7 for further discussion). We begin by relaxing the poisoning problem, and then describe an efficient method for solving it.

*Deriving the Upper-Bounding Problem.* To upper-bound the loss of multi-point attacks, we relax Definition 2 by allowing duplicate poisons and permitting poisons to lie in $\mathcal{K}$. Formally, we arrive at the following problem:

Definition 3 (**Relaxed Poisoning Problem**). *Let $\mathcal{K} = \{k_1, k_2, \ldots, k_n\} \subset \mathbb{N}$ be $n$ distinct legitimate keys arranged in increasing order, i.e., $k_1 < k_2 < \cdots < k_n$. Let $\lambda \in \mathbb{N}$ be the maximum number of poison points. The Relaxed Poisoning Problem is to determine the multiset $\mathcal{P}^*$ of integers satisfying $|\mathcal{P}^*| \leq \lambda$ and $\mathcal{P}^* \subset \{k_1, k_1 + 1, \ldots, k_n\}$ that maximizes $E(\mathcal{K} \uplus \mathcal{P}^*)$, where $\uplus$ denotes multiset addition. Formally,*

$$\mathcal{P}^* \coloneqq \underset{\mathcal{P} \text{ s.t. } |\mathcal{P}| \leq \lambda, \mathcal{P} \subset \{k_1, k_1+1, \ldots, k_n\}}{\operatorname{argmax}} E(\mathcal{K} \uplus \mathcal{P}), \tag{10}$$

The solution to Definition 3 provides an upper bound on the loss of the original poisoning problem (Definition 2), because Definition 3 relaxes the constraints on the poison set $\mathcal{P}$. Note that unlike Definition 2, this relaxed formulation remains feasible even when $\mathcal{K} = \{k_1, k_1 + 1, \ldots, k_n\}$, because it allows $\mathcal{P}$ to be a multiset and to include values from $\mathcal{K}$.

By applying an argument similar to that in Theorem 2, we can establish the following theorem regarding the structure of the optimal solution $\mathcal{P}^*$ to the Relaxed Poisoning Problem.

Theorem 3. *The optimal solution $\mathcal{P}^*$ to the Relaxed Poisoning Problem is a multiset over $\mathcal{K}$, i.e.,* $\mathrm{Supp}(\mathcal{P}^*) \subset \mathcal{K}$, *where* Supp *denotes the support of the multiset.*

Proof of Theorem 3. The proof is analogous to that of Theorem 2: if there exists poison points outside of $\mathcal{K}$, we can increase the loss by shifting them to the keys within $\mathcal{K}$. □

Thus, letting $Q_{\mathcal{K}}(\boldsymbol{d})$ be the multiset containing $d_i$ copies of $k_i$ for each $i$ (with $\boldsymbol{d} \in \mathbb{Z}^n_{\geq 0}$), the relaxed problem reduces to finding

$$\boldsymbol{d}^* \coloneqq \underset{\boldsymbol{d} \text{ s.t. } \sum_{i=1}^n d_i \leq \lambda}{\operatorname{argmax}} E(\mathcal{K} \uplus Q_{\mathcal{K}}(\boldsymbol{d})). \tag{11}$$

The following theorem states that in the Relaxed Poisoning Problem, it is always optimal to fully exhaust the given budget $\lambda$:

Theorem 4. *The optimal solution $\boldsymbol{d}^*$ to the Relaxed Poisoning Problem satisfies $\sum_{i=1}^n d_i^* = \lambda$.*

Proof Sketch of Theorem 4. We first prove Lemma 3 (in Appendix C of the full version [46]), which states that in the relaxed setting, there exists some $i \in \{1, 2, \ldots, n\}$ such that adding a poison at $x_i$ increases the loss. This implies that using the full poisoning budget is optimal. □

The full proof is provided in Appendix C of the full version [46]. Next, we derive the following inequality to further upper-bound the solution of the Relaxed Poisoning Problem:

$$\max_{\boldsymbol{d} \text{ s.t. } \sum_{i=1}^n d_i = \lambda} E(\mathcal{K} \uplus Q_{\mathcal{K}}(\boldsymbol{d})) \tag{12}$$

$$= \max_{\boldsymbol{d} \text{ s.t. } \sum_{i=1}^n d_i = \lambda} \min_{w,b} \mathcal{L}(\mathcal{K} \uplus Q_{\mathcal{K}}(\boldsymbol{d}); w, b) \tag{13}$$

$$\leq \min_{w} \max_{\boldsymbol{d} \text{ s.t. } \sum_{i=1}^n d_i = \lambda} \min_{b} \mathcal{L}(\mathcal{K} \uplus Q_{\mathcal{K}}(\boldsymbol{d}); w, b). \tag{14}$$

The equality follows from the definition of $E(\mathcal{K} \uplus Q_{\mathcal{K}}(\boldsymbol{d}))$ in Definition 1. The inequality follows from the standard max-min inequality: $\max_x \min_y f(x, y) \leq \min_y \max_x f(x, y)$, for any function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. By computing the value of Equation (14), we obtain an upper bound on the loss achievable by multi-point attacks.

*Computing the Upper Bound.* To determine the value of Equation (14), we make use of the following two theorems.

THEOREM 5. *For fixed $\boldsymbol{d}$, $\min_b \mathcal{L}(\mathcal{K} \uplus Q_{\mathcal{K}}(\boldsymbol{d}); w, b)$ is a convex quadratic function of $w$.*

PROOF SKETCH OF THEOREM 5. For fixed $w$, the optimal $b^*$ can be solved in closed form, and substituting it shows the loss is a quadratic function in $w$ with positive quadratic coefficient. □

THEOREM 6. *For fixed $w > 0$, the choice of $\boldsymbol{d}$ (subject to $\sum_{i=1}^{n} d_i = \lambda$) that maximizes $\min_b \mathcal{L}(\mathcal{K} \uplus Q_{\mathcal{K}}(\boldsymbol{d}); w, b)$ is characterized as follows:*

- *If $w(k_n - k_1) < n + \lambda$, all poison points are concentrated on a single key; that is, $\exists i \in \{1, 2, \ldots, n\}$ such that $d_i = \lambda$.*
- *Otherwise, all poison points are concentrated at the endpoints $k_1$ and $k_n$; that is, $d_1 + d_n = \lambda$.*

PROOF SKETCH OF THEOREM 6. When $w(k_n - k_1) < n + \lambda$, for any $1 \le i < j \le n$ with $d_i, d_j \ge 1$, moving a poison from $i$ to $j$, or from $j$ to $i$, always increases the loss. When $w(k_n - k_1) \ge n + \lambda$, for any $2 \le i \le n - 1$ with $d_i \ge 1$, moving a poison from $i$ to 1, or from $i$ to $n$, always increases the loss. □

The full proofs are provided in Appendix C of the full version [46]. By Theorem 6, the number of possible $\boldsymbol{d}$ configurations that maximize $\min_b \mathcal{L}(\mathcal{K} \uplus Q_{\mathcal{K}}(\boldsymbol{d}); w, b)$ is at most $O(n + \lambda)$. For each such $\boldsymbol{d}$, Theorem 5 implies that $\min_b \mathcal{L}(\mathcal{K} \uplus Q_{\mathcal{K}}(\boldsymbol{d}); w, b)$ is a convex quadratic function of $w$. By properly implementing differential computation, we can compute the coefficients of these quadratic functions in time $O(n + \lambda)$, as detailed in Appendix D of the full version [46]. Thus, evaluating Equation (14) reduces to the following problem: given convex quadratic functions $f_i(w) : \mathbb{R} \to \mathbb{R}$ for $i \in \{1, 2, \ldots, O(n + \lambda)\}$, determine the value of $\min_w \max_i f_i(w)$. There exist several methods to efficiently solve this problem. Three representative approaches are as follows:

(1) **Golden section search over $w$**: Minimize the convex function $\max_i f_i(w)$ via golden section search [42]. With $T$ iterations, the time complexity is $O(T(n + \lambda))$.
(2) **Binary search over $y$ (function values)**: Given $y \in \mathbb{R}$, we can test in $O(n + \lambda)$ time whether $y \ge \min_w \max_i f_i(w)$. Thus, binary search [7] with $T$ iterations can solve the problem in $O(T(n + \lambda))$ time.
(3) **Exact structural analysis**: Since $\max_i f_i(w)$ is piecewise quadratic with $O(n + \lambda)$ pieces by Davenport–Schinzel sequence theory [50], we can construct its exact representation in $O((n + \lambda) \log(n + \lambda))$ time and then compute the exact minimum directly (unlike the previous two iterative methods).

Detailed algorithms are given in Appendix D of the full version [46]. By employing one of these methods, we can evaluate Equation (14) with a time complexity of either $O(T(n + \lambda))$ or $O((n + \lambda) \log(n + \lambda))$, thereby obtaining an upper bound on the impact of multi-point poisoning attacks (see Table 1).

### 5.3 Segment + Endpoint Attack

Motivated by Theorem 6—which implies that under the relaxed setting with a fixed $w$, the optimal poison mass concentrates either at the two extremes or at a single interior point—we propose a simple structured attack strategy called *Segment + Endpoint (Seg+E)*. A Seg+E attack uses as poisons at most three consecutive blocks: two on the extreme blocks adjacent to $k_1$ and $k_n$, and a single consecutive segment not adjacent to these extremes. We define Seg+E in the relaxed setting

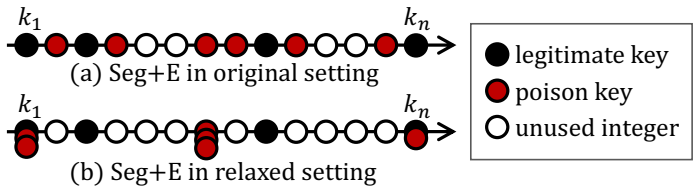



Fig. 5. Examples of the Segment + Endpoint (Seg+E) attack in original and relaxed settings.

(Definition 3) analogously: a Seg+E multiset in the relaxed setting consists of at most three integers, namely those corresponding to $k_1$, $k_n$, and a single interior integer. Figure 5 illustrates Seg+E attacks in the original and relaxed settings. The formal mathematical definition is given in Appendix E of the full version [46].

We propose three algorithms for finding the Seg+E solution. As summarized in Table 1, we refer to the exact algorithms as *Exact Seg+E* and to the fast approximation for the original setting as *Heuristic Seg+E*. First, we present an $O(n\lambda^3)$-time Exact Seg+E algorithm for the original setting, followed by an $O(n\lambda)$-time Exact Seg+E algorithm for the relaxed setting. Finally, by using the relaxed-setting solution as guidance, we design an $O(n\lambda)$-time Heuristic Seg+E algorithm for the original setting, which empirically finds a solution very close to the optimal one.

*$O(n\lambda^3)$-time Exact Seg+E algorithm for the original setting.* In the original setting, we can find an exact Seg+E solution in $O(n\lambda^3)$ time. The number of ways to allocate the $\lambda$ poison budget to the left endpoint, the middle segment, and the right endpoint is $O(\lambda^3)$. For each such allocation, an argument analogous to the proof of Theorem 2 shows that the middle segment admits only $O(n)$ candidate locations that need to be considered; see Theorem 7 in Appendix E.2 of the full version [46] for a formal proof. Therefore the total number of Seg+E candidates to evaluate is $O(n\lambda^3)$, and by computing the MSE for each candidate efficiently via differential updates we obtain the exact Seg+E solution in $O(n\lambda^3)$ time.

*$O(n\lambda)$-time Exact Seg+E algorithm for the relaxed setting.* In the relaxed setting, we can find an exact Seg+E solution in $O(n\lambda)$ time. Using an argument similar to the proof of Theorem 4, we first show that any exact Seg+E solution in the relaxed problem exhausts the budget $\lambda$. As in the original setting, the location of the middle segment admits only $O(n)$ candidate positions. Crucially, when the number of poisons placed at the left endpoint and the middle-segment location are fixed, the optimal number of poisons assigned to the middle segment can be determined in $O(1)$ time, by extending the loss to a continuous function in the middle-segment multiplicity and using its derivative to identify the optimum. Hence, iterating over all choices of left-endpoint count and middle-segment location (there are $O(n\lambda)$ such combinations) and evaluating the corresponding MSEs yields an $O(n\lambda)$-time algorithm for finding the exact Seg+E solution in the relaxed setting.

*$O(n\lambda)$-time Heuristic Seg+E algorithm for the original setting.* We also design an $O(n\lambda)$-time Heuristic Seg+E algorithm that yields a Seg+E solution that is not guaranteed to be exact but is empirically very close to the exact one. The heuristic algorithm leverages the Seg+E solution in the relaxed setting as a guide: we fix the numbers of poisons at the left endpoint, middle segment, and right endpoint to match those in the relaxed-setting solution. Then, we try the $O(n)$ candidate middle-segment positions and select the best placement. Because the Seg+E solution in the relaxed setting can be found in $O(n\lambda)$ time and we can evaluate the MSE for each candidate position in $O(1)$ time, the overall time complexity of the heuristic algorithm is $O(n\lambda)$.

## 6 Experimental Evaluation

In this section, we experimentally evaluate our upper-bounding methods and Seg+E algorithms. We first assess how tightly the proposed bounds match the attack impact achieved by the greedy method (Section 6.2), then compare the loss achieved by Seg+E and the greedy attack (Section 6.3). We also measure running time (Section 6.4), conduct ablation studies on key parameters (Section 6.5), and present a detailed analysis in the small-scale setting, where the optimum is computable (Section 6.6). Finally, we evaluate the impact of poisoning on *lookup time*, i.e., the time to retrieve values from a sorted array using the linear regression model (Section 6.7).

*Key take-aways.* The main findings are as follows:

(1) For both real-world and synthetic datasets, there is only a small gap between the upper bound and the iterative greedy approach. This indicates that the bound is tight and that the greedy approach achieves near-optimal solutions.
(2) The Exact Seg+E attack is never worse than the greedy attack, and the Heuristic Seg+E attack always attains a loss very close to that of the Exact Seg+E attack.
(3) The upper bound can be determined in time faster than running the greedy algorithm, allowing one to effectively judge the quality of the computed solution.
(4) The tightness of the bound remains stable across both key density and dataset size, with only moderate degradation observed under extremely dense distributions.
(5) The gap decomposition analysis identifies the main source of looseness in the upper bounds: the max-min step dominates when key density is low, while the duplicate-allowing relaxation dominates when key density is extremely high.
(6) Poisoning substantially increases lookup time: at a 20% poisoning ratio, it slows down lookups by up to 1.6×.

### 6.1 Experimental Setup

We implemented all upper-bounding, Seg+E, and greedy poisoning algorithms, in C++. All experiments were conducted on a Linux machine equipped with an Intel® Core™ i9-11900H CPU @ 2.50GHz and 64GB of memory. We compiled the code using GCC version 9.4.0 with the `-O3` optimization flag enabled and ran all experiments in a single-threaded setting. To verify that our claims hold across typical key-distribution patterns, we evaluate on several synthetic and real-world datasets used in a standard learned-index benchmark.

*Synthetic Datasets.* Each synthetic legitimate key set is generated based on a specified distribution (Uniform, Normal, or Exponential), random seed, range $R \in \mathbb{N}$, and value of $n \in \mathbb{N}$, using the following procedure: We sampled $n$ real values, scaled them to $[0, R]$, rounded to integers, and removed duplicates. When $R$ is small, a larger fraction of the integer domain is occupied by legitimate keys, leaving fewer integer values available for the attacker to use.

*Real-World Datasets.* Each real-world legitimate key set is constructed by extracting $n$ consecutive keys from a real sorted dataset, where the starting position is selected randomly. We use Amzn, Face, and Osmc datasets from SOSD [36], a benchmark for learned indexes. This setup reflects the practical use case where linear regression models are often deployed as leaf models handling consecutive keys in learned indexes [11, 30, 52, 58].

*Upper Bound Algorithms.* We evaluate the three upper bound computation algorithms described in Section 5.2. For the first (golden section search) and second (binary search) approaches, we fix the number of iterations $T$ to 50. Since the difference among their results is always very small (less

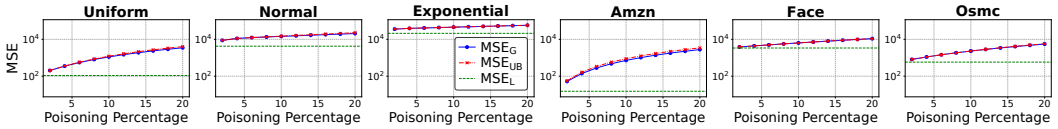



Fig. 6. $\mathrm{MSE}_{\mathrm{UB}}$ (our upper bound), $\mathrm{MSE}_{\mathrm{G}}$ (greedy attack), and $\mathrm{MSE}_{\mathrm{L}}$ (legitimate) for seed = 0. We set $n = 1{,}000$ and $R = 100{,}000$. We observe that $\mathrm{MSE}_{\mathrm{UB}} \geq \mathrm{MSE}_{\mathrm{G}}$ in all cases and the gap is small, which indicates that our upper bound is tight and greedy attack is close to the optimal attack.

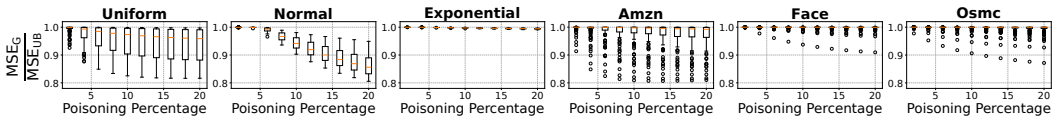



Fig. 7. Boxplots of $\mathrm{MSE}_{\mathrm{G}}/\mathrm{MSE}_{\mathrm{UB}}$ over 100 seeds for each dataset and poisoning percentage. We set $n = 1{,}000$ and $R = 100{,}000$. The ratio never exceeds 1 and is always greater than 0.8. In particular, for Exponential, Amzn, Face, and Osmc, the median ratio is $\geq 0.99$, which indicates that our upper bound is tight and greedy attack is close to the optimal attack.

than $10^{-9}$), we report results from the first algorithm unless otherwise noted (e.g., for running time evaluation in Section 6.4).

### 6.2 Upper Bound vs Greedy Poisoning

Here, we show that the gap between the upper bound computed by our proposed method ($\mathrm{MSE}_{\mathrm{UB}}$) and the loss from the greedy poisoning method ($\mathrm{MSE}_{\mathrm{G}}$) is consistently small. This demonstrates that our upper bound method provides an efficient approach to measuring the impact of multi-point attacks, and that the greedy attack approximates the optimal attack reasonably well.

First, Figure 6 shows the comparison among $\mathrm{MSE}_{\mathrm{G}}$, $\mathrm{MSE}_{\mathrm{UB}}$, and the MSE of legitimate keys ($\mathrm{MSE}_{\mathrm{L}}$) for seed = 0. We set $n = 1{,}000$ for all datasets, and set $R = 100{,}000$ for synthetic data. The poisoning percentage is defined as $\lambda/n$.

We observe that $\mathrm{MSE}_{\mathrm{G}}$ increases with larger poisoning percentages. The MSE of legitimate keys varies significantly depending on the distribution and $n$, and so does the increase in MSE caused by poisoning. In all cases, $\mathrm{MSE}_{\mathrm{UB}}$ is greater than $\mathrm{MSE}_{\mathrm{G}}$ and *tight* in the sense that $\mathrm{MSE}_{\mathrm{G}}/\mathrm{MSE}_{\mathrm{UB}}$ is always at least 0.81.

We further observe differences across datasets: settings that are locally closer to uniform (e.g., Uniform and Amzn) tend to yield smaller MSE on legitimate keys, yet can exhibit larger increases in MSE after the attack. For such near-uniform datasets, we also find that the discrepancy between $\mathrm{MSE}_{\mathrm{UB}}$ and $\mathrm{MSE}_{\mathrm{G}}$ tends to be larger.

Next, Figure 7 presents the distribution of the ratio $\mathrm{MSE}_{\mathrm{G}}/\mathrm{MSE}_{\mathrm{UB}}$ over 100 seeds. Values of this ratio close to 1 indicate a tight upper bound and near-optimal performance of the greedy attack, whereas values close to 0 indicate a loose bound or weak greedy performance. Each boxplot represents the distribution across 100 seeds for a given configuration of ($n$, dataset, poisoning percentage). Note that the randomness comes only from data generation; all algorithms are deterministic. We set $n = 1{,}000$ for all datasets, and set $R = 100{,}000$ for synthetic data.

We observe that, in every configuration, $\mathrm{MSE}_{\mathrm{G}}/\mathrm{MSE}_{\mathrm{UB}}$ never exceeds 1, and the median over 100 trials is always above 0.85. In particular, for Exponential, Amzn, Face, and Osmc, the median is consistently above 0.99, showing that our upper bound and the greedy results are very close. For Uniform and Normal, the gap between upper bound and greedy results tends to grow, especially

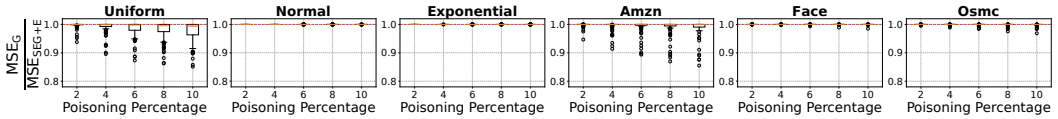



Fig. 8. Boxplots of $\mathrm{MSE_G}/\mathrm{MSE_{Seg+E}}$ over 100 seeds for each dataset and poisoning percentage. We set $n = 1{,}000$ and $R = 100{,}000$. The ratio never exceeds 1, that is, the Seg+E attack is never worse than the greedy attack.

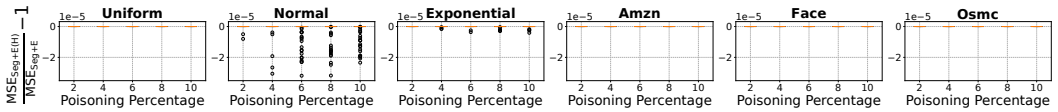



Fig. 9. Boxplots of $\mathrm{MSE_{Seg+E(H)}}/\mathrm{MSE_{Seg+E}} - 1$ over 100 seeds for each dataset and poisoning percentage (for better visibility, we plot the value obtained by subtracting 1 from the ratio). We set $n = 1{,}000$ and $R = 100{,}000$. The ratio consistently remains very close to 1, indicating that Heuristic Seg+E achieves losses nearly identical to those of Exact Seg+E.

when the poisoning percentage is large. That said, across all 6,000 trials, the ratio never falls below 0.8, and the overall average remains as high as 0.97. This demonstrates that (i) the greedy multi-point poisoning attack leaves at most about 25% room for improvement (and only around 3% on average), and (ii) our proposed upper bound is consistently tight across datasets and settings.

### 6.3 Greedy Poisoning vs Seg+E

Here, we compare $\mathrm{MSE_G}$, $\mathrm{MSE_{Seg+E}}$ (the MSE obtained by Exact Seg+E), and $\mathrm{MSE_{Seg+E(H)}}$ (the MSE obtained by Heuristic Seg+E).

Figure 8 shows the distribution of $\mathrm{MSE_G}/\mathrm{MSE_{Seg+E}}$ over 100 seeds. A ratio below 1 means that Exact Seg+E yields a larger MSE than Greedy. Across all 3,000 cases (6 datasets $\times$ 5 poisoning percentages $\times$ 100 seeds), the ratio never exceeds 1, i.e., Exact Seg+E is never worse than the Greedy. For the Uniform and Amzn datasets, the two methods sometimes differ noticeably; the ratio $\mathrm{MSE_G}/\mathrm{MSE_{Seg+E}}$ can drop to below 0.86, which means that Seg+E yields up to about 1.16 times larger MSE than Greedy.

Figure 9 shows the distribution of $\mathrm{MSE_{Seg+E(H)}}/\mathrm{MSE_{Seg+E}}$ over 100 seeds. Over the entire set of 3,000 measurements, we observed that this ratio is always greater than 0.99996. This indicates that the Heuristic Seg+E attack is very close to the Exact Seg+E attack. Although $\mathrm{MSE_{Seg+E(H)}}$ is rarely worse than $\mathrm{MSE_G}$, the maximum observed value of the ratio $\mathrm{MSE_G}/\mathrm{MSE_{Seg+E(H)}}$ is only 1.00004. In other words, Heuristic Seg+E provides an excellent trade-off between computational efficiency and attack effectiveness.

### 6.4 Running time Evaluation

We measure the running time of our three upper-bounding methods, whose time complexities are $O(T(n+\lambda))$, $O(T(n+\lambda))$, and $O((n+\lambda)\log(n+\lambda))$, respectively. We also report the running time of our Seg+E algorithms (Exact: $O(n\lambda^3)$, Heuristic: $O(n\lambda)$, and Exact in the relaxed setting: $O(n\lambda)$), as well as the running time of the greedy poisoning attack [28], which runs in $O((n+\lambda)\lambda)$ time.

Figure 10 shows the relationship between poisoning percentage and their running time for each algorithm. Since $\lambda$ is the product of $n$ and the poisoning percentage, $\lambda$ is proportional to the poisoning percentage. We set $n = 1{,}000$ and $R = 100{,}000$. We see that the running time of the greedy poisoning increases linearly with poisoning percentage, aligning with the complexity $O((n+\lambda)\lambda)$ (note that $\lambda < n$ in our setting). In contrast, the proposed upper-bound algorithms (Golden, Binary,

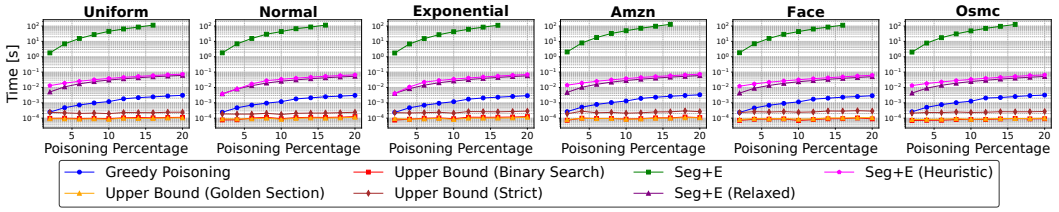



Fig. 10. Running time vs poisoning percentage. We set $n = 1{,}000$ and $R = 100{,}000$. The running time of the greedy poisoning method grows linearly with the poisoning percentage (consistent with $O((n+\lambda)\lambda)$ time complexity), whereas our upper-bound methods (Golden, Binary, Strict) are nearly flat (consistent with $O(T(n+\lambda))$ or $O((n+\lambda)\log(n+\lambda))$ time complexity).

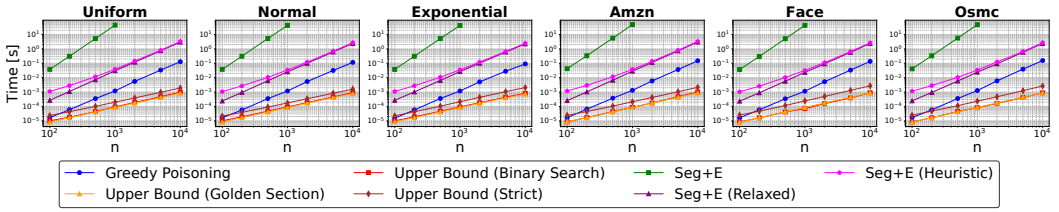



Fig. 11. Running time vs $n$. We set the poisoning percentage to 10% and $R = 100{,}000$. The running time of the greedy poisoning method grows quadratically with $n$ (consistent with $O((n+\lambda)\lambda)$ time complexity), whereas our upper-bound methods (Golden, Binary, Strict) grow linearly with $n$ (consistent with $O(T(n+\lambda))$ or $O((n+\lambda)\log(n+\lambda))$ time complexity).

Strict) show minimal change in running time. This is because, when $\lambda < n$, the time complexity of the proposed upper-bound algorithms is $O(n)$ or $O(n \log n)$.

As expected, the $O(n\lambda^3)$-time Exact Seg+E algorithm is significantly slower than the other algorithms, taking more than 10,000 times longer than the greedy algorithm. In contrast, the Exact Seg+E algorithm in the relaxed setting and the Heuristic Seg+E algorithm, both of which run in $O(n\lambda)$ time, are much faster. However, they are still about 20 times slower than the greedy algorithm despite having the same asymptotic time complexity. This is because the Seg+E algorithms involve heavier constant-time operations, such as solving cubic equations, resulting in a larger constant factor.

Figure 11 shows the relationship between $n$ and the running time of each algorithm. We set the poisoning percentage to 10% and $R = 100{,}000$. The running time of the greedy poisoning method increases quadratically with $n$, consistent with the time complexity $O((n+\lambda)\lambda)$. In contrast, the proposed upper-bound algorithms (Golden, Binary, Strict) scale as $O(n)$ or $O(n \log n)$ and are much faster than the greedy algorithm. In particular, the two approximation methods (Golden and Binary) are extremely fast, with running times no greater than 0.003 seconds.

These results show that our methods compute upper bounds in practical time even for large $n$ and $\lambda$. This efficiency enables downstream uses, such as lightweight attack-quality assessment and as a surrogate objective for higher-level attack-generation procedures; we discuss these use cases in Section 7.

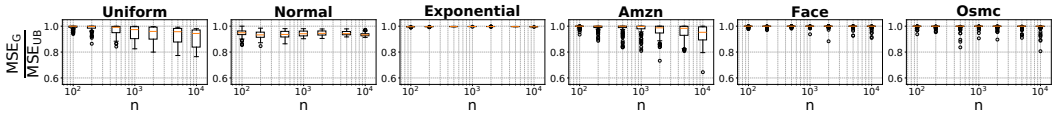



Fig. 12. $\mathrm{MSE_G}/\mathrm{MSE_{UB}}$ vs $n$. We set the poisoning percentage to 10% and $R = 100{,}000$. As $n$ increases, $\mathrm{MSE_G}/\mathrm{MSE_{UB}}$ shows a slight tendency to decrease.

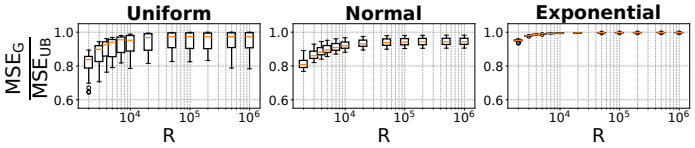



Fig. 13. $\mathrm{MSE_G}/\mathrm{MSE_{UB}}$ vs $R$ for synthetic data. We set the poisoning percentage to 10% and $n = 1{,}000$ for all datasets. When $R$ is quite small, the ratio tends to decrease.

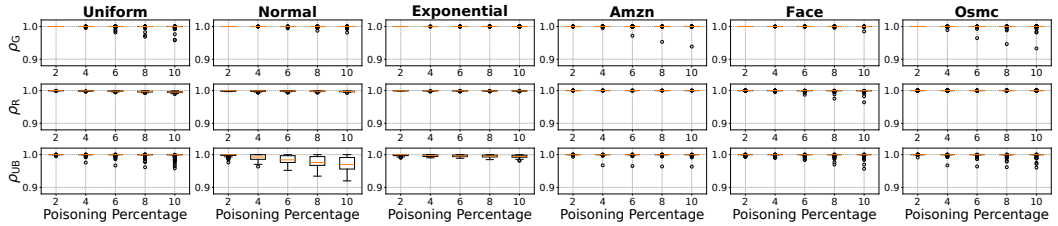



Fig. 14. Boxplots of the ratios $\rho_{\mathrm{G}}$, $\rho_{\mathrm{R}}$, and $\rho_{\mathrm{UB}}$, defined in Equation (15), over 100 seeds for each dataset and poisoning percentage. We set $n = 50$ and $R = 1{,}000$. All ratios remain close to 1 (minimum 0.92), with $\mathrm{MSE_{OPT}}/\mathrm{MSE_{ROPT}}$ particularly tight.

### 6.5 Ablation Study of $n$ and $R$

Here, we investigate how $\mathrm{MSE_G}/\mathrm{MSE_{UB}}$ changes with respect to $n$ and $R$. Figure 12 shows the relationship between $n$ and the ratio $\mathrm{MSE_G}/\mathrm{MSE_{UB}}$ for synthetic data. We set the poisoning percentage to 10% and $R = 100{,}000$ for synthetic data. As $n$ increases, we observe a slight decreasing trend in the ratio. That is, the ratio remains consistently tight: the minimum value is 0.64 (or 0.73 if excluding a single outlier seed from Amzn), while the average value is always above 0.91. This indicates that, regardless of $n$, our proposed upper bound tightly bounds the attack impact.

Figure 13 shows the relationship between $R$ and $\mathrm{MSE_G}/\mathrm{MSE_{UB}}$. We set the poisoning percentage to 10% and $n = 1{,}000$ for all datasets. We observe that the smaller the value of $R$, the smaller the ratio tends to be. This is expected because our upper bound is obtained by relaxing the constraint to allow duplicate keys, and the effect of this relaxation becomes more pronounced when $R$ is small. In particular, the ratio worsens when $R < 10^4$. However, note that $R = 10^4$ already corresponds to a fairly narrow key range (i.e., a dense distribution), where 10% of the integers contained in the range are legitimate. In the real datasets we used, the range spanned by $n = 1{,}000$ consecutive keys is at least $1.8 \times 10^{13}$ (Amzn), $2.8 \times 10^4$ (Face), and $2.3 \times 10^{10}$ (Osmc). Thus, the degradation observed for small $R$ is negligible in practice, and our upper bound remains tight under realistic settings.

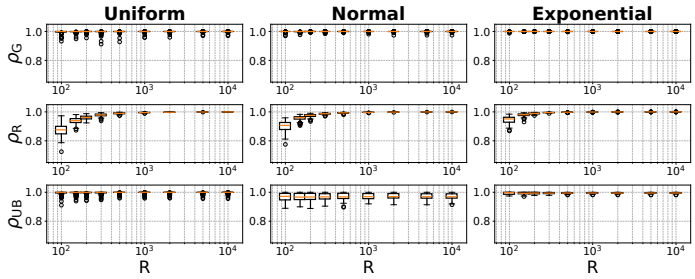



Fig. 15. The ratios ($\rho_G$, $\rho_R$, and $\rho_{UB}$) vs $R$ for synthetic data. We set the poisoning percentage to 10% and $n = 50$. $\rho_{UB}$ and $\rho_G$ remain stable as $R$ varies, whereas $\rho_R$ stays close to 1 for large $R$ but drops when $R$ is small.

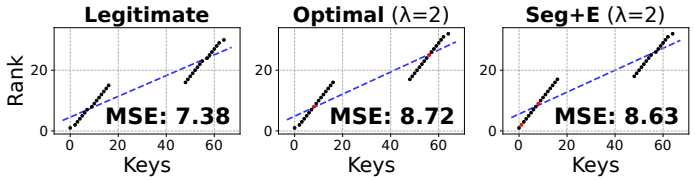



Fig. 16. A counter example where the Exact Seg+E solution is not globally optimal. The legitimate key set is $\mathcal{K} = \{0, 1, \ldots, 16, 48, 49, \ldots, 64\} \setminus \{1, 8, 56, 63\}$. The optimal solution is $\mathcal{P} = \{8, 56\}$, but the Exact Seg+E solution is $\mathcal{P} = \{1, 8\}$.

### 6.6 Detailed Analysis in Small-Scale Settings

Here, we present a detailed analysis in the small-scale setting, where our theory makes it possible to obtain the optimal solution within practical time. As explained in Section 5, Theorem 2 implies that the number of candidates for the optimal solution to the original problem is at most $\binom{2n-2+\lambda}{\lambda}$, allowing us to compute $\mathrm{MSE}_{\mathrm{OPT}}$ in $O\left((n+\lambda)\binom{2n-2+\lambda}{\lambda}\right)$ time. Similarly, Theorem 2 and Theorem 4 imply that there are at most $\binom{n+\lambda-1}{\lambda}$ candidates in the relaxed setting, allowing us to compute $\mathrm{MSE}_{\mathrm{ROPT}}$ in $O\left((n+\lambda)\binom{n+\lambda-1}{\lambda}\right)$ time. Hence, we can compute the global optimum for moderately small $n$ and $\lambda$; in our experiments, we set $n = 50$ and $\lambda \leq 5$.

First, we examine the gap between the greedy solution and the upper bound. Since our algorithm produces upper bounds by further upper-bounding the optimal solution of the relaxed problem, letting $\mathrm{MSE}_{\mathrm{ROPT}}$ as the MSE under the optimal solution of the relaxed problem, the inequalities $\mathrm{MSE}_{\mathrm{G}} \leq \mathrm{MSE}_{\mathrm{OPT}} \leq \mathrm{MSE}_{\mathrm{ROPT}} \leq \mathrm{MSE}_{\mathrm{UB}}$ hold. Accordingly, we evaluate the following three ratios:

$$\rho_G \coloneqq \frac{\mathrm{MSE}_{\mathrm{G}}}{\mathrm{MSE}_{\mathrm{OPT}}}, \quad \rho_R \coloneqq \frac{\mathrm{MSE}_{\mathrm{OPT}}}{\mathrm{MSE}_{\mathrm{ROPT}}}, \quad \rho_{UB} \coloneqq \frac{\mathrm{MSE}_{\mathrm{ROPT}}}{\mathrm{MSE}_{\mathrm{UB}}}. \tag{15}$$

All of these ratios are less than or equal to 1. Values of $\rho_G$ near 1 indicate that the greedy attack is close to optimal, values of $\rho_R$ near 1 indicate that the gap introduced by the relaxation is small, and values of $\rho_{UB}$ near 1 indicate that the max-min upper-bounding step (Equation (14)) is tight. Note that the product of these ratios equals the ratio examined in the previous sections, $\mathrm{MSE}_{\mathrm{G}}/\mathrm{MSE}_{\mathrm{UB}}$. By decomposing it in this manner, we can analyze in detail which component contributes to the observed deviation from 1.

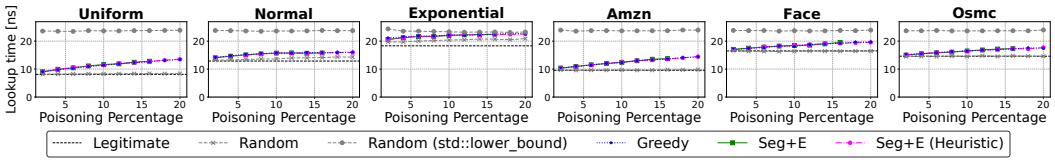



Fig. 17. Lookup time. The lookup time includes both regression inference and exponential search. We measure it 10 times for each key and report the average. We set $n = 1{,}000$ and $R = 100{,}000$. Poisoning increases the lookup time by up to 1.6 times.

Figure 14 reports the distributions of $\rho_{\mathrm{G}}$, $\rho_{\mathrm{R}}$, and $\rho_{\mathrm{UB}}$ over 100 seeds; we zoom the $y$-axis to a narrow neighborhood of 1 for readability. Across datasets and settings, all three ratios are consistently close to 1 (all exceed 0.92) and exhibit a mild decreasing trend as $\lambda$ increases. We observe that $\rho_{\mathrm{G}}$ tends to be smaller (i.e., the gap between the greedy and the optimal solutions is larger) on Uniform, Amzn, and Osmc. We also observe that $\rho_{\mathrm{R}}$ is generally closer to 1 than $\rho_{\mathrm{UB}}$, suggesting that the main source of looseness comes from the max-min inequality rather than from permitting duplicate poisons. We also note that $\rho_{\mathrm{UB}}$ is most frequently smaller on the Normal dataset, whereas $\rho_{\mathrm{R}}$ tends to be smaller on the Face dataset.

Figure 15 studies how the ratios vary with $R$ on synthetic data (with $n = 50$ and a 10% poisoning percentage). We find that both $\rho_{\mathrm{UB}}$ and $\rho_{\mathrm{G}}$ are largely insensitive to $R$. In contrast, $\rho_{\mathrm{R}}$ is close to 1 for moderately large $R$ (e.g., $R \geq 1{,}000$) but can be somewhat smaller when $R$ is small. Taken together with the ablation in Section 6.5, these findings reinforce that the dominant contributor to the overall gap at small $R$ is the duplicate-allowing relaxation, whereas for larger $R$ the relaxation is tight and the remaining looseness mainly stems from the max-min upper-bounding step.

Next, we compare Seg+E with the optimal solution. We set $n = 50$ and $R = 1{,}000$. The summary statistics of $\mathrm{MSE}_{\mathrm{Seg+E}}/\mathrm{MSE}_{\mathrm{OPT}}$ for both the original and relaxed settings are reported in Table 1 as *Empirical Perf. Ratio*. We observed that, under both the original and relaxed settings, the Exact Seg+E solution always matched the global optimal solution in all 3,000 cases (6 datasets, 5 poisoning percentages, 100 seeds). However, in the original setting, we have already found corner cases where the Exact Seg+E solution is not globally optimal. Figure 16 (along with additional corner cases in Appendix F of the full version [46]) shows one such example. Here, the optimal solution is $\mathcal{P} = \{8, 56\}$, consisting of two isolated points, which is not Seg+E and has larger MSE than the Exact Seg+E solution. Thus, while there exist rare corner cases where Seg+E is suboptimal, it almost always coincides with the global optimum in practice.

### 6.7 Impact of Poisoning on Lookup Time

To quantify the practical slowdown caused by poisoning in linear models of learned indexes, we measure its impact on the *lookup time*. We define lookup time as the time from receiving a query key to obtaining its position in the sorted array, i.e., the total time for linear-regression inference and an exponential search around the predicted position. We query all legitimate keys, repeat the measurement 10 times per key, and report the average. We set $n = 1{,}000$ and $R = 100{,}000$. As a reference, we include a random-poisoning baseline, where poisons are sampled uniformly from the key domain, and we also report the time of `std::lower_bound`.

Figure 17 reports lookup time before and after poisoning. We find that poisoning increases lookup time by up to 1.6× at 20% poisoning. By contrast, randomly chosen poisons barely affect lookup time, indicating that poisons generated by Greedy [28] or our Seg+E algorithm increase the lookup time effectively. These results quantify how poisoning can slow down the leaf-model lookup

routine by degrading the model accuracy, suggesting that poisoning can measurably increase the end-to-end lookup time in learned indexes.

## 7 Discussion

*On Defending Against Poisoning Attacks.* Defending linear regression over the CDF against poisoning attacks is challenging, as also discussed in [28]. In our experiments, adding only 4% poison points increased the MSE by up to 10×, suggesting that simply tracking the cardinality of the key set is unlikely to be sufficient for reliable poisoning detection. Furthermore, poisons are not outliers in this setting: they lie within the domain and are placed adjacent to legitimate keys, making them hard to detect or mitigate with robust regression methods such as Huber regression [23] and RANSAC [18]. Developing defenses tailored to regression over the CDF remains an important direction for future work.

*On Using the Upper Bound for Defense.* Our upper bound provides a worst-case guarantee that the poisoning impact never exceeds the bound. For example, if an index must operate within a tolerance of at most a 10× increase in MSE, a defender can estimate how many additional keys can be inserted while ensuring that the MSE provably remains within this range.

*On Using the Upper Bound for Attack.* Computing the upper bound is substantially faster than the greedy attack (Section 6.4), enabling several downstream uses. First, an attacker can use the bound to cheaply assess candidate solutions: one can run the upper-bound method in parallel with the greedy attack and stop early when the gap is small; otherwise, switch to a more sophisticated (and potentially more expensive) attack algorithm to approach the optimum. Second, the bound can serve as a proxy for the post-attack MSE in higher-level attack-generation procedures. For example, the RMI attack of Kornaropoulos et al. [28] performs hill climbing by repeatedly evaluating the post-attack MSE of each linear model; replacing these evaluations with our bound can substantially reduce the computational cost.

*On Empirical Observations and Conjectures.* Based on extensive experiments and analyses, we report several empirical observations and formulate key conjectures (see Appendix F of the full version [46] for details). First, we observe that optimal poisons tend to concentrate near the endpoints or near locations where the regression line intersects the point sequence induced by the legitimate keys (Observation 3); Appendix F of the full version [46] also provides an intuitive mathematical explanation for this phenomenon. As observed in Section 6.6, Seg+E matches the optimum in many cases, while Figure 16 shows intentionally constructed corner cases where Seg+E is not optimal (Observation 4). Moreover, in every instance we tested, Seg+E is optimal in the relaxed setting; we leave as a conjecture that this holds in general (Conjecture 1). We also propose an auxiliary conjecture (Conjecture 2) which, if proven, would imply Conjecture 1.

*On Non-Integer Keys.* Although we state the problem for integer keys, the same analyses can be extended to non-integer keys. For ordered non-integer keys (e.g., strings), we can apply an order-preserving bijection to map them to a contiguous integer range. For multi-dimensional keys, as in prior spatial learned indexes [55, 56], we can map them to one-dimensional integer keys via space-filling curves. Training and poisoning are performed on the resulting integers, and all of our analyses apply directly.

*On the Implications for Learned Indexes.* Directly applying our analysis to learned indexes (e.g., RMI [29]) is not feasible because they are hierarchical and nonlinear. However, typical learned indexes rely on a large number of linear regression models. In such cases, our analysis can be applied to obtain (near-)optimal attacks or upper bounds for each linear regression model. Moreover,

motivated by the performance of our Seg+E technique, we conjecture that for some models, a global worst-case attack influences only a small part of the model.

*On Dynamic Settings.* Our contributions may also offer insights into dynamic settings, where the key set evolves through updates and an attacker injects poisons sequentially to maximize cumulative loss. A key challenge is to quantify the gap between online attacks (which choose poisons without knowing future updates) and offline attacks (which plan poisons with full knowledge of the update sequence), and to understand how defenses should adapt. Our upper bound enables lightweight evaluation of the worst-case impact at each update step. Moreover, the structure of our Seg+E approach may inspire simple heuristics for online poison selection, and it may help defenders design triggers for poisoning detection and retraining.

## 8 Limitations and Future Work

One limitation of our theoretical guarantees is that we were unable to prove the convexity of the loss function in single-point poisoning attacks. While experimental results consistently suggest that this convexity holds, we could not provide a formal proof. Although our current findings are sufficient to ensure that the single-point poisoning attack can efficiently lead to the optimal solution, proving this convexity would be a valuable theoretical insight.

Additionally, developing methods to compute tighter bounds remains an important direction for future research. We observed that the gap between our upper bound and the greedy solution can sometimes be relatively large. Part of this gap reflects the difference between the greedy method and the true optimum, but some of it stems from the looseness of our bound itself (as discussed in Section 6.6). This looseness stems from two approximations made in the derivation of the bound: (i) allowing duplicate poison keys, and (ii) swapping the order of $\min_{w}$ and $\max_{\boldsymbol{d}}$ in Equation (14). Avoiding these approximations and designing methods to obtain tighter bounds remain important topics for future work.

Another important future direction is to extend our analysis beyond linear regression. We deliberately focused on linear regression, as it is the simplest and most widely used leaf model in learned indexes. This restriction enabled a clean theoretical foundation and clear insights into adversarial robustness, but it also limits generality. A natural next step is to broaden the framework to nonlinear regression models (e.g., higher-order polynomials or neural networks) and to multi-stage learned indexes.

## 9 Conclusion

In this work, we provided the first theoretical advancements on single-point and multi-point poisoning attacks against linear regression models trained on CDFs. For single-point attacks, we proved that a previously proposed heuristic—whose optimality had remained unclear—is indeed guaranteed to yield the optimal solution. For multi-point attacks, we derived the key structural properties of the optimal attack, proposed methods to guarantee an upper bound on the achievable loss, and proposed algorithms to find exact and heuristic Seg+E solutions efficiently. Through experiments, we confirmed that the greedy poisoning method achieves near-optimal attacks, and at the same time, our methods tightly upper-bound the actual attack impact. Overall, our contributions advance the theoretical understanding of linear regression models on CDFs, providing a foundation for analyzing both the vulnerabilities and robustness of learned indexes under adversarial conditions.

## Acknowledgments

This work was supported by the JSPS Bilateral Program (Grant Number JPJSBP120259915) and by JSPS KAKENHI (Grant Number 25KJ0754).

## References


[1] Abdullah Al-Mamun, Hao Wu, Qiyang He, Jianguo Wang, and Walid G. Aref. 2025. A Survey of Learned Indexes for the Multi-dimensional Space. *ACM Comput. Surv.* 58 (2025), 37 pages.

[2] Kasun Amarasinghe, Farhana Choudhury, Jianzhong Qi, and James Bailey. 2025. Learned Indexes with Distribution Smoothing via Virtual Points. In *Proceedings of the International Conference on Extending Database Technology*. OpenProceedings, Barcelona, Spain, 668–680.

[3] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2020. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems* 87 (2020), 13 pages.

[4] Martin Aumüller and Matteo Ceccarello. 2023. Recent Approaches and Trends in Approximate Nearest Neighbor Search, with Remarks on Benchmarking. *IEEE Data Eng. Bull.* 47 (2023), 89–105.

[5] Rudolf Bayer and Edward M. McCreight. 1972. Organization and maintenance of large ordered indexes. *Acta Inf.* 1 (1972), 17 pages.

[6] Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13 (1970), 422–426.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, 3rd Edition*. MIT Press, Cambridge, MA.

[8] Luis Alberto Croquevielle, Guang Yang, Liang Liang, Ali Hadian, and Thomas Heinis. 2025. Beyond Logarithmic Bounds: Querying in Constant Expected Time with Learned Indexes. In *International Conference on Database Theory*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 19:1–19:21.

[9] Yifan Dai, Yien Xu, Aishwarya Ganesan, Ramnatthan Alagappan, Brian Kroth, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau. 2020. From WiscKey to Bourbon: A Learned Index for Log-Structured Merge Trees. In *USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association, Berkeley, CA, USA, 155–171.

[10] Zhenwei Dai and Anshumali Shrivastava. 2020. Adaptive learned bloom filter (ada-bf): Efficient utilization of the classifier with application to real-time information filtering on the web. *Advances in Neural Information Processing Systems* 33 (2020), 11700–11710.

[11] Jialin Ding, Umar Farooq Minhas, Jia Yu, Chi Wang, Jaeyoung Do, Yinan Li, Hantian Zhang, Badrish Chandramouli, Johannes Gehrke, Donald Kossmann, David Lomet, and Tim Kraska. 2020. ALEX: An Updatable Adaptive Learned Index. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, New York, NY, USA, 969–984.

[12] Jialin Ding, Vikram Nathan, Mohammad Alizadeh, and Tim Kraska. 2020. Tsunami: a learned multi-dimensional index for correlated data and skewed workloads. *Proc. VLDB Endow.* 14 (2020), 74–86.

[13] Emanuele Dolera, Stefano Favaro, and Stefano Peluchetti. 2023. Learning-augmented count-min sketches via Bayesian nonparametrics. *Journal of Machine Learning Research* 24 (2023), 60 pages.

[14] Fangming Dong, Pinghui Wang, Rundong Li, Xueyao Cui, Junzhou Zhao, Jing Tao, Chen Zhang, and Xiaohong Guan. 2025. Poisoning Attacks and Defenses to Learned Bloom Filters for Malicious URL Detection. *IEEE Transactions on Dependable and Secure Computing* 22 (2025), 3275–3288.

[15] Harman Singh Farwah, Gagandeep Singh, and Cheng Tan. 2024. Exploiting Time Channel Vulnerability of Learned Bloom Filters. In *ICLR Tiny Papers Track*. OpenReview.net, Vienna, Austria, 5 pages.

[16] Paolo Ferragina, Fabrizio Lillo, and Giorgio Vinciguerra. 2020. Why Are Learned Indexes So Effective?. In *Proceedings of the International Conference on Machine Learning*. PMLR, Virtual, 3123–3132.

[17] Paolo Ferragina and Giorgio Vinciguerra. 2020. The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds. *Proc. VLDB Endow.* 13 (2020), 14 pages.

[18] Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24 (1981), 381–395.

[19] Alex Galakatos, Michael Markovitch, Carsten Binnig, Rodrigo Fonseca, and Tim Kraska. 2019. FITing-Tree: A Data-aware Index Structure. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, Amsterdam, The Netherlands, 1189–1206.

[20] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. *SIGOPS Oper. Syst. Rev.* 37 (2003), 15 pages.

[21] Fuma Hidaka and Yusuke Matsui. 2024. FlexFlood: Efficiently Updatable Learned Multi-dimensional Index. In *NeurIPS Workshop on Machine Learning for Systems*. NeurIPS Workshop on Machine Learning for Systems, Vancouver, BC, Canada, 13 pages.

[22] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. 2019. Learning-Based Frequency Estimation Algorithms. In *International Conference on Learning Representations*. OpenReview.net, New Orleans, LA, USA, 20 pages.

[23] Peter J Huber. 1964. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*. Springer, New York, NY, 492–518.

[24] Xuyang Jing, Xiaojun Cheng, Zheng Yan, and Xian Li. 2022. Deceiving Learning-based Sketches to Cause Inaccurate Frequency Estimation. In *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*.

IEEE, Wuhan, China, 209–216.
[25] Minsu Kim, Jinwoo Hwang, Guseul Heo, Seiyeon Cho, Divya Mahajan, and Jongse Park. 2024. Accelerating String-Key Learned Index Structures via Memoization-Based Incremental Training. *Proc. VLDB Endow.* 17 (2024), 14 pages.
[26] Andreas Kipf, Ryan Marcus, Alexander van Renen, Mihail Stoian, Alfons Kemper, Tim Kraska, and Thomas Neumann. 2020. RadixSpline: a single-pass learned index. In *International Workshop on Exploiting Artificial Intelligence Techniques for Data Management.* Association for Computing Machinery, New York, NY, USA, 5 pages.
[27] Donald E. Knuth. 1998. *The Art of Computer Programming, Volume 3: Sorting and Searching* (2nd ed.). Addison-Wesley, Reading, Massachusetts.
[28] Evgenios M Kornaropoulos, Silei Ren, and Roberto Tamassia. 2022. The price of tailoring the index to your data: Poisoning attacks on learned index structures. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.* ACM, Philadelphia, PA, USA, 1331–1344.
[29] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The Case for Learned Index Structures. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.* Association for Computing Machinery, New York, NY, USA, 489–504.
[30] Pengfei Li, Yu Hua, Jingnan Jia, and Pengfei Zuo. 2021. FINEdex: a fine-grained learned index scheme for scalable and concurrent memory systems. *Proc. VLDB Endow.* 15 (2021), 321–334.
[31] Pengfei Li, Yu Hua, Pengfei Zuo, and Jingnan Jia. 2019. A scalable learned index scheme in storage systems. *CoRR* abs/1905.06256 (2019), 13 pages.
[32] Pengfei Li, Hua Lu, Qian Zheng, Long Yang, and Gang Pan. 2020. LISA: A learned index structure for spatial data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.* Association for Computing Machinery, New York, NY, USA, 2119–2133.
[33] Qiyu Liu, Siyuan Han, Yanlin Qi, Jingshu Peng, Jin Li, Longlong Lin, and Lei Chen. 2025. Why Are Learned Indexes So Effective but Sometimes Ineffective? *Proc. VLDB Endow.* 18 (2025), 13 pages.
[34] Baotong Lu, Jialin Ding, Eric Lo, Umar Farooq Minhas, and Tianzheng Wang. 2021. APEX: a high-performance learned index on persistent memory. *Proc. VLDB Endow.* 15 (2021), 597–610.
[35] Kai Lu, Nannan Zhao, Jiguang Wan, Changhong Fei, Wei Zhao, and Tongliang Deng. 2022. TridentKV: A Read-Optimized LSM-Tree Based KV Store via Adaptive Indexing and Space-Efficient Partitioning. *IEEE Transactions on Parallel and Distributed Systems* 33 (2022), 1953–1966.
[36] Ryan Marcus, Andreas Kipf, Alexander van Renen, Mihail Stoian, Sanchit Misra, Alfons Kemper, Thomas Neumann, and Tim Kraska. 2020. Benchmarking Learned Indexes. *Proc. VLDB Endow.* 14 (2020), 1–13.
[37] Ryan Marcus, Emily Zhang, and Tim Kraska. 2020. Cdfshop: Exploring and optimizing learned index structures. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.* Association for Computing Machinery, New York, NY, USA, 2789–2792.
[38] Michael Mitzenmacher. 2018. A model for learned bloom filters and optimizing by sandwiching. *Advances in Neural Information Processing Systems* 31 (2018), 464–473.
[39] Dingheng Mo, Fanchao Chen, Siqiang Luo, and Caihua Shan. 2023. Learning to Optimize LSM-trees: Towards A Reinforcement Learning based Key-Value Store for Dynamic Workloads. *Proc. ACM Manag. Data* 1 (2023), 25 pages.
[40] Vikram Nathan, Jialin Ding, Mohammad Alizadeh, and Tim Kraska. 2020. Learning Multi-Dimensional Indexes. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.* Association for Computing Machinery, New York, NY, USA, 985–1000.
[41] Kyosuke Nishishita, Atsuki Sato, and Yusuke Matsui. 2025. Optimized Learned Count-Min Sketch. In *NeurIPS Workshop on Machine Learning for Systems.* NeurIPS Workshop, San Diego, CA, USA, 13 pages.
[42] William H Press. 2007. *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge University Press, New York, NY, USA.
[43] Jianzhong Qi, Guanli Liu, Christian S. Jensen, and Lars Kulik. 2020. Effectively learning spatial indices. *Proc. VLDB Endow.* 13 (2020), 2341–2354.
[44] Raghu Ramakrishnan and Johannes Gehrke. 2002. *Database Management Systems.* McGraw-Hill, Inc., United States.
[45] Pedro Reviriego, José Alberto Hernández, Zhenwei Dai, and Anshumali Shrivastava. 2021. Learned bloom filters in adversarial environments: A malicious URL detection use-case. In *IEEE International Conference on High Performance Switching and Routing.* IEEE, Paris, France, 1–6.
[46] Atsuki Sato, Martin Aumüller, and Yusuke Matsui. 2026. Mathematical Foundations of Poisoning Attacks on Linear Regression over Cumulative Distribution Functions. *CoRR* abs/2603.00537 (2026), 26 pages.
[47] Atsuki Sato and Yusuke Matsui. 2023. Fast Partitioned Learned Bloom Filter. *Advances in Neural Information Processing Systems* 36 (2023), 28 pages.
[48] Roei Schuster, Jin Peng Zhou, Thorsten Eisenhofer, Paul Grubbs, and Nicolas Papernot. 2025. Learned-Database Systems Security. *Transactions on Machine Learning Research* 2025 (2025), 23 pages.

[49] Hinrich Schütze, Christopher D. Manning, and Prabhakar Raghavan. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, England.
[50] Micha Sharir. 1988. Davenport-Schinzel sequences and their geometric applications. In *Theoretical Foundations of Computer Graphics and CAD*. Springer, Berlin, 253–278.
[51] Benjamin Spector, Andreas Kipf, Kapil Vaidya, Chi Wang, Umar Farooq Minhas, and Tim Kraska. 2021. Bounding the last mile: Efficient learned string indexing. *CoRR* abs/2111.14905 (2021), 5 pages.
[52] Zhaoyan Sun, Xuanhe Zhou, and Guoliang Li. 2023. Learned index: A comprehensive experimental evaluation. *Proc. VLDB Endow.* 16 (2023), 1992–2004.
[53] Chuzhe Tang, Youyun Wang, Zhiyuan Dong, Gansen Hu, Zhaoguo Wang, Minjie Wang, and Haibo Chen. 2020. XIndex: a scalable learned index for multicore data storage. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. Association for Computing Machinery, New York, NY, USA, 308–320.
[54] Kapil Vaidya, Eric Knorr, Tim Kraska, and Michael Mitzenmacher. 2021. Partitioned Learned Bloom Filters. In *International Conference on Learning Representations*. OpenReview.net, Online, 17 pages.
[55] Haixin Wang, Xiaoyi Fu, Jianliang Xu, and Hua Lu. 2019. Learned Index for Spatial Queries. In *IEEE International Conference on Mobile Data Management*. IEEE, Hong Kong, Hong Kong, 569–574.
[56] Ning Wang and Jianqiu Xu. 2020. Spatial queries based on learned index. In *International Conference on Spatial Data and Intelligence*. Springer-Verlag, Berlin, Heidelberg, 245–257.
[57] Taiyi Wang, Liang Liang, Guang Yang, Thomas Heinis, and Eiko Yoneki. 2025. A New Paradigm in Tuning Learned Indexes: A Reinforcement Learning Enhanced Approach. *Proc. ACM Manag. Data* 3 (2025), 26 pages.
[58] Youyun Wang, Chuzhe Tang, Zhaoguo Wang, and Haibo Chen. 2020. SIndex: a scalable learned index for string keys. In *Proceedings of the ACM SIGOPS Asia-Pacific Workshop on Systems*. Association for Computing Machinery, New York, NY, USA, 8 pages.
[59] Jiacheng Wu, Yong Zhang, Shimin Chen, Yu Chen, Jin Wang, and Chunxiao Xing. 2021. Updatable Learned Index with Precise Positions. *Proc. VLDB Endow.* 14 (2021), 13 pages.
[60] Shangyu Wu, Yufei Cui, Jinghuan Yu, Xuan Sun, Tei-Wei Kuo, and Chun Jason Xue. 2022. NFL: robust learned index via distribution transformation. *Proc. VLDB Endow.* 15 (2022), 2188–2200.
[61] Lei Yang, Hong Wu, Tieying Zhang, Xuntao Cheng, Feifei Li, Lei Zou, Yujie Wang, Rongyao Chen, Jianying Wang, and Gui Huang. 2020. Leaper: a learned prefetcher for cache invalidation in LSM-tree based storage engines. *Proc. VLDB Endow.* 13 (2020), 14 pages.
[62] Rui Yang, Evgenios M. Kornaropoulos, and Yue Cheng. 2023. Algorithmic Complexity Attacks on Dynamic Learned Indexes. *Proc. VLDB Endow.* 17 (2023), 780–793.
[63] Yifan Yang and Shimin Chen. 2024. LITS: An Optimized Learned Index for Strings. *Proc. VLDB Endow.* 17 (2024), 3415–3427.
[64] Sepanta Zeighami and Cyrus Shahabi. 2023. On Distribution Dependent Sub-Logarithmic Query Time of Learned Indexing. In *Proceedings of the International Conference on Machine Learning*. PMLR, Honolulu, Hawaii, USA, 12 pages.
[65] Sepanta Zeighami and Cyrus Shahabi. 2024. Theoretical Analysis of Learned Database Operations under Distribution Shift through Distribution Learnability. In *Proceedings of the International Conference on Machine Learning*. JMLR.org, Vienna, Austria, 23 pages.
[66] Jintao Zhang, Chao Zhang, Guoliang Li, and Chengliang Chai. 2024. PACE: Poisoning attacks on learned cardinality estimation. *Proceedings of the ACM on Management of Data* 2 (2024), 1–27.
[67] Meifan Zhang, Hongzhi Wang, Jianzhong Li, and Hong Gao. 2020. Learned Sketches for Frequency Estimation. *Information Sciences* 507 (2020), 21 pages.
[68] Yihang Zheng, Chen Lin, Xian Lyu, Xuanhe Zhou, Guoliang Li, and Tianqing Wang. 2024. Robustness of Updatable Learning-based Index Advisors against Poisoning Attack. *Proceedings of the ACM on Management of Data* 2 (2024), 1–26.
[69] Wei Zhou, Chen Lin, Xuanhe Zhou, Guoliang Li, and Tianqing Wang. 2024. TRAP: Tailored Robustness Assessment for Index Advisors via Adversarial Perturbation. In *IEEE International Conference on Data Engineering*. IEEE, Utrecht, The Netherlands, 42–55.
[70] Weihong Zhou and Shiyu Yang. 2024. SLIPP: a space-efficient learned index for string keys. In *Proceedings of the International Conference on Big-Data Service and Intelligent Computation*. Association for Computing Machinery, New York, NY, USA, 69–77.

Received October 2025; revised January 2026; accepted February 2026