DISTRIBUTED SECURITY ENHANCEMENT FROM A DATA MANAGEMENT PERSPECTIVE: THEORETICAL FOUNDATIONS AND PRACTICAL APPLICATIONS

YUE FU

PhD

The Hong Kong Polytechnic University

2024

The Hong Kong Polytechnic University
Department of Electrical and Electronic Engineering

Distributed Security Enhancement from A Data Management Perspective: Theoretical Foundations and Practical Applications

Yue Fu

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
April 2024

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Yue Fu

Abstract

This thesis explores the critical need for enhanced security measures in distributed systems, addressing vulnerabilities exposed by sophisticated cyber-attacks such as DDoS. It delves into advanced cryptographic techniques and data management strategies designed to secure data outsourcing, enable searchable encryption, and protect data integrity against online manipulations. Motivated by imperative security goals in the current landscape of distributed systems, this thesis develops three innovative privacy-preserving schemes designed to enhance security measures and ensure robust data protection. The first scheme addresses vulnerabilities in password authentication systems, employing a novel cryptographic approach that minimizes computational overhead and enhances resistance against brute-force attacks. This methodology secures password transactions and efficiently manages large volumes of authentication requests. The second scheme focuses on secure indexing, particularly for bi-attribute datasets common in advanced data analytics and AI applications. It introduces an optimized searchable encryption technique that allows for secure and efficient querying of encrypted databases without compromising data privacy. This scheme leverages modified probabilistic data structures to enhance performance and reduce potential privacy leaks through advanced inference attacks. The third scheme tackles the challenge of maintaining data integrity in the face of sophisticated online manipulation attacks. It integrates a game-theoretical model that dynamically adapts to evolving threats, effectively countering malicious attempts to alter data. This model uses a combination of real-time threat analysis and strategic response mechanisms to ensure

that data remains accurate and reliable, thus preserving the integrity essential for critical decision-making processes.

Publications Arising from the Thesis

1. Y. Fu, Q. Ye, R. Du, and H. Hu. "Interactive Trimming against Evasive Online Data Manipulation Attacks: A Game-Theoretic Approach", *IEEE International Conference on Data Engineering (ICDE)*, May 13-17, 2024, Utrecht.

2. Y. Fu, Q. Ye, R. Du, and H. Hu. "Collecting Multi-type and Correlation-Constrained Streaming Sensor Data with Local Differential Privacy.", *ACM Transactions on Sensor Networks*, September 2023, https://doi.org/10.1145/3623637.

3. Y. Fu, M. H. Au, R. Du, H. Hu, and D. Li. "Cloud Password Shield: A Secure Cloud-based Firewall against DDoS on Authentication Servers", *Proc. of 40th IEEE International Conference on Distributed Computing Systems (ICDCS '20)* , July 8 – 10, 2020, Singapore, pp. 1209-1210, doi: 10.1109/ICDCS47774.2020.00154.

4. Y. Fu, Q. Ye, R. Du, and H. Hu. "Unified Proof of Work: Delegating and Solving Customized Computationally Bounded Problems in A Privacy-preserving Way", *The 6th APWeb-WAIM International Joint Conference on Web and Big Data (APWeb-WAIM)*, 2022, Lecture Notes in Computer Science, vol 13423. Springer, Cham. https://doi.org/10.1007/978-3-031-25201-3_24.

5. Y. Fu, Q. Ye, R. Du, and H. Hu. "Secure Bi-attribute Index: Batch Membership Tests over the Non-sensitive Attribute", submitted to *Computers & Security.*

6. R. Du, Q. Ye, Y. Fu, and H. Hu. "Collecting High-Dimensional and Correlation-Constrained Data with Local Differential Privacy", *Proc. of 18th IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2021, 19373-19386.

7. R. Du, Q. Ye, Y. Fu, H. Hu, J. Li, C. Fang, and J. Shi. "Differential Aggregation against General Colluding Attackers" *IEEE International Conference on Data Engineering (ICDE)*, 2023.

8. R. Du, Q. Ye, Y. Fu, and H. Hu. "Distribution estimation under LDP against arbitrarily distributed attacks", submitted to *IEEE Transactions on Knowledge and Data Engineering*.

9. R. Du, Q. Ye, Y. Fu, H. Hu and K. Huang. "Top-k Discovery under Local Differential Privacy: An Adaptive Sampling Approach", submitted to *IEEE Transactions on Dependable and Secure Computing*.

10. R. Du, Q. Ye, Y. Fu, and H. Hu. "Leveraging Historical Perturbation for Data Stream Publication under Local Differential Privacy", submitted to *VLDB*.

Acknowledgments

As I stand on the threshold of completing my doctoral journey, I am filled with gratitude and would like to extend my deepest appreciation to several individuals whose guidance, support, and influence have been pivotal to my achievements.

First and foremost, my heartfelt thanks go to my supervisor, Prof. Haibo Hu, and my co-supervisor, Dr. Qingqing Ye. At a particularly challenging time in my life before beginning my PhD, their recognition and the opportunity they provided were instrumental in launching my academic career. I am especially grateful for their financial support, academic guidance, and the respect they have shown for my personal research interests and aspirations.

I am also profoundly grateful to every member of the ASTAPLE group. A special mention goes to my wife, Rong Du, who has not only been my companion but also an integral collaborator throughout my seven years of academic life. Each of my papers bears the imprint of your substantial contributions. My thanks also extend to The Hong Kong Polytechnic University, which provided invaluable resources and opportunities that greatly enriched my studies.

I owe a debt of gratitude to my parents and all my relatives for their selfless support, often given without full understanding, but always with unconditional love. I appreciate all the reviewers who accepted my papers for enriching my publication list; I am equally thankful to those who rejected my papers, as your insightful comments helped enhance my work.

Finally, I am grateful for the abundance of knowledge available in this world that provides us with endless learning opportunities, and for the even greater mysteries that remain unsolved, giving us reasons to continue exploring and discovering in the future.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Backgrounds

In today's digital age, the ubiquity and continuous generation of data have become a staple of our everyday lives, especially with the advent of the Internet of Things (IoT). This connectivity, while beneficial, also presents significant security challenges. The proliferation of IoT devices, often with inadequate security measures, has exacerbated the frequency and intensity of Distributed Denial of Service (DDoS) attacks, which not only disrupt service but also exploit devices to spread malware, turning them into bots for large-scale network assaults. Despite advancements in network defenses like cloud and network firewalls, DDoS attacks have evolved, targeting application layers and overwhelming services like user authentication systems with brute-force attacks. This scenario underscores the pressing need for robust, distributed security enhancements in data management systems. Parallel to these concerns, the field of cryptography has seen significant advances, enabling secure, privacy-preserving data outsourcing to untrusted third parties such as cloud servers. Searchable encryption techniques, for example, allow for keyword searches over encrypted data without needing decryption, addressing the dual needs of confidentiality and usability. This is

particularly relevant as the volume of data escalates, necessitating efficient and cost-effective storage and processing solutions like secure indices which utilize probabilistic data structures for managing searchable encryptions. The integrity and security of data are paramount, given the potential for malicious entities to manipulate data and impacting decision-making processes. The integrity of data is continuously at risk from attacks designed to alter or fabricate data to influence outcomes. To mitigate these risks, data management strategies need to incorporate dynamic defense mechanisms, such as evasion-aware strategies using game theory, to maintain data integrity and counteract sophisticated online data manipulation attacks.

In today's digital landscape, the proliferation of interconnected devices and systems has highlighted significant security challenges. The primary goal of this thesis is to develop comprehensive security solutions for distributed systems from a data management perspective. This goal addresses the urgent need for enhanced security measures to protect data integrity, confidentiality, and availability in the face of evolving cyber threats. The motivation for this research stems from the increasing vulnerabilities in distributed systems, especially with the rise of IoT devices prone to sophisticated attacks such as DDoS. The rationale for breaking down the research goal into three key tasks is to methodically tackle different aspects of security: password authentication, secure indexing, and data integrity. These tasks are interrelated, as they collectively contribute to a holistic approach to distributed security enhancement.

Password Authentication. This task focuses on developing a robust authentication system that minimizes vulnerabilities in password management. By employing innovative cryptographic techniques, the system aims to resist brute-force attacks and manage high volumes of authentication requests typical in IoT environments.

Secure Indexing. This task addresses the challenge of querying encrypted data efficiently without compromising privacy. By optimizing searchable encryption techniques, the system ensures secure and efficient access to encrypted databases, crucial for data analytics and AI applications.

Data Integrity against Online Attacks. This task aims to protect data from online manipulation attacks. By integrating game-theoretical models, the system dynamically adapts to threats, ensuring data accuracy and reliability for critical decision-making processes.

1.2 Contributions and Thesis Organization

The contributions of this thesis are threefold:

- Development of a novel cryptographic approach for secure password authentication.
- Introduction of an optimized searchable encryption technique for secure indexing.
- Creation of a game-theoretical model for maintaining data integrity against manipulation attacks.

Followed is a roadmap of thesis organization in each chapter:

Chapter 2: This chapter offers a thorough review of relevant literature and studies and the basic preliminaries applied throughout the thesis, encompassing DDoS attack countermeasures, probabilistic structures, local differential privacy protocols, and literature pertaining to data poisoning attacks.

Chapter 3: This chapter presents a cloud-based firewall based on Bloom filters to pre-screen and reject bad requests with wrong password before they arrive at the authentication server. The main challenge is the security of the passwords. We consider the firewall might be compromised and the Bloom filters are retrieved by an adversary to conduct offline brute-force attacks and

restore (hashed) passwords. To ensure the security, we start with a trusted firewall provider and design a key-based semantic secure Bloom filter (KSSBF) to achieve the best efficiency. We then design a generically secure Bloom filter (GSBF) for non-trusted firewall providers, whose security is strictly provable and key-independent. Through theoretical and empirical analysis, we show both of them can mitigate malicious traffics arrive at the back end while not degrading the security of passwords.

Chapter 4: This chapter presents a secure bi-attribute indexing solution, illustrated with a case study on searchable encryption for time-series data. We introduce two variants of matrix Bloom filters tailored for different workloads and implement a concept of bounded privacy loss via noise infusion from the randomized response technique. The outcome adheres to locally differential privacy principles, offering a provable privacy guarantee for sensitive attribute items.

Chapter 5: This chapter presents an interactive game-theoretical model to defend online data manipulation attacks using the trimming strategy. Our model accommodates a complete strategy space, making it applicable to strong evasive and colluding adversaries. Leveraging the principle of least action and the Euler-Lagrange equation from theoretical physics, we derive an analytical model for the game-theoretic process. To demonstrate its practical usage, we present a case study in a privacy-preserving data collection system under local differential privacy where a non-deterministic utility function is adopted. Two strategies are devised from this analytical model, namely, Tit-for-tat and Elastic. We conduct extensive experiments on real-world datasets, which showcase the effectiveness and accuracy of these two strategies.

Chapter 6: We conclude the outcomes of this thesis and propose new directions for future work.

Chapter 2

Literature Review and Preliminaries

This chapter provides an overview of the key concepts, technologies, and related work that form the foundation for the research presented in this thesis. We begin with a broad introduction to distributed systems security, followed by specific background on the main topics addressed in our work, and conclude with some preliminary knowledge on variational methods, which is utilized in later chapters.

2.1 Distributed Systems Security

2.1.1 DDos Countermeasures

Rate limitation has long been an important methodology of DDoS defence mechanism to provide bounded accessability for a single node. For instance, Dwork C. and Naor M. proposed a computational technique for controlling access to a shared resource, i.e. combating junk mail [28]. To prevent frivolous use, users are asked to compute a moderately hard function in order to gain access to the resource. Von Ahn L. et

al. introduced CAPTCHA, using unsolved AI problems to distinguish computer programs from human-being [80].

DDoS defence mechanism can be employed in the network level. There exists various IP traceback mechanisms to trace back the forged IP packets to their true sources, rather than spoofed ones [19, 43]. Victims can identify the path of attack traffic and distinguish them from legitimate ones [1, 19, 74]. There also exists packet marking and filtering mechanisms, which aims to mark legitimate packets at each router along their path to the destination, so that edge routers can filter the attack traffic. Peng et al. proposed a history-based IP filtering mechanism, which enable victims to filter bandwidth attack traffic according to the history that were maintained at ordinary times. Wang et al. proposed a Hop-count filtering mechanism, which records information about a source IP (as well as its corresponding hops) into a table for the victim's query. Yaar et al. proposed a Path identifier (Pi) to embed a path fingerprint in each packet, which enables victims to identify packets by traversing the same paths. There are also route-based packet filtering mechanisms [66, 67], malicious router detection mechanisms [57], detecting and stopping attack traffic inside the router.

DDoS defence mechanism can also be employed in the application level. Ranjan et al. proposed DDoS-Shield, which employs a rate limiting pre-defence mechanism using a statistical model [70, 71]. Srivatsa et al. proposed an admission control protocol to limit the number of users connected to a network service concurrently [76]. There are also mechanisms that try to distinguish malicious bots from legitimate users. Kandula et al. employed a completely automated public Turing test to differenate bots from humans [44]. Oikonomou et al. proposed defences against flash-crowd attacks, which distinguishes bots from humans baed on three uniqueness of human behaviours [61]. To the best of the author's knowledge, the ultimate goal of DDoS defence mechanism is to detect malicious nodes/traffics, and filter them as soon as possible.

2.1.2 Local Differential Privacy

Differential privacy (DP) [26, 27, 54] is a mathematical foundation of quantizing privacy protection, usually by appropriately using Laplacian [27], Gaussian [52] or Geometric distribution [33] to randomise the results of statistical queries in interactive query-response systems. Many researches have studied it from various fields, including theory analyses [6, 24, 45], data publication [17, 90, 91, 99], machine learining [5], and system [10]. However, DP works on the assumption of the existence of a trusted third-party server, which is regarded impractical in the privacy-aware crowdsourced systems. To deal with this problem, local differential privacy (LDP) [18,24,45] model was recently proposed to provide a stringent privacy guarantee for crowd-sourced systems where data providers trust no one but themselves.

Local differential privacy (LDP) [27] is proposed to provide a stringent privacy guarantee for users in a non-trusted data-collecting system. We say a perturbation mechanism $\mathcal{M}$, a non-deterministic algorithm mapping an input to some output with a certain probability, satisfies with ϵ-LDP ($\epsilon \geq 0$), if and only if for any two data records S_1, S_2 and any possible output $T \in \text{Range}(\mathcal{M})$, the following condition holds:

$$\frac{P[\mathcal{M}(S_1) = T]}{P[\mathcal{M}(S_2) = T]} \leq e^{\epsilon} \tag{2.1}$$

This is a formal definition of LDP, and the set of all possible outputs is called the value range. Since $P[\mathcal{M}(S_1) = T]$ is very close to $P[\mathcal{M}(S_2) = T]$, it is hard to tell any individual's true answer from observing the outcome. The most straightforward perturbation algorithm to guarantee LDP is the Randomized Response (RR) [84], which has been widely used in the "Yes or No" sensitive problem. Users will flip a biased coin and send true answers with probability p and $1 - p$ for false answers. A widely used RR for binary cases is given as follows:

$$b' = \begin{cases} b, & w.p.\ p \\ 1 - b & w.p.\ 1 - p \end{cases}$$

The privacy guarantee LDP [24,45] employed in our paper is initially proposed for protecting data privacy in an untrusted environment, where each user perturbs her values before sending them to the data collector. Some typical scenarios for applications of LDP include frequency estimation [6], mean estimation [24], and high-dimensional data collections [22]. Besides, LDP has been widely applied to real-life applications, such as Chorme, iOS, Win 10, and Samsung.

A fundamental goal of LDP functionality is frequency estimation. RAPPOR [30], which was proposed and well-employed into Chrome by Google, encodes users' data into a Bloom filter and then performs RR on each bit of Bloom filter, which enables the decoding result more accurate. However, the false positive rate of Bloom filter shall be restricted, thus the Bloom filter is necessary to be sparse, which renders the communication cost unsatisfactory. Bassily and Smith [6] proposed a 1-bit protocol for frequency estimation to optimize the communication cost. However, the data utility is still unsatisfactory. The parameter results are further optimized in the Optimized Unary Encoding (OUE) [30] protocol, which achieves significantly better accuracy. This literature also designed an OLH protocol, which provides much better accuracy, but still requires a $O(logn)$ communication cost. Note that all of the above methods focus on LDP on a single attribute, while some recent attempts discuss the multi-attribute case, such as spatiotemprial data [92] and high-dimensional data [22, 23]. Another interesting problem in LDP is mean estimation over numerical data, which have been widely studied in literature [24, 25].

LDPMiner [5] is a frequency publishing method, which is targeted at heavy hitter queries. Firstly, the data collector collects data and determines the Heavy Hitter set, and returns it to the user. Then, the user sends data corresponding to some of the items in the set to the data collector. Ren et al. develop the Lopub, [72] which is focused on high-dimensional crowdsourced data publication. However, the communication cost of this approach is very high, since the transmission of every attribute is the size of a Bloom filter. PM and HM mechanisms [82] perturb the

input value into a probability density function to get a better result accuracy.

2.1.3 Data Poisoning Attacks

In recent years, data poisoning attacks and their countermeasures have gained considerable attention, with numerous studies exploring various aspects, particularly in machine learning. Chen et al. [16] devise a black-box attack method capable of bypassing defenses against data poisoning, emphasizing the need for more robust countermeasures. Biggio et al. [9] examine poisoning attacks targeting SVM and introduced an effective attack strategy, highlighting the necessity for robust defenses in SVMs. Liu et al. [53] explore the transferability of adversarial examples, which are relevant to data poisoning attacks, and develop a black-box attack method based on this property. Steinhardt et al. [77] introduce certification, offering guarantees on a model's robustness against data poisoning attacks, and present a certified training algorithm. Jagielski et al. [42] propose an optimal attack strategy for poisoning regression models and develop effective countermeasures to minimize the attacker's impact. Mei and Zhu [56] employ a machine teaching approach to identify the most damaging data poisoning attacks on machine learning models, providing insights into both attack and defense strategies. Several other works also address data poisoning attacks and their countermeasures [58, 59, 86].

Manipulation attacks are more eminent in privacy-preserving scenarios (such as LDP [12, 40, 78]) where the perturbations can amplify the effect of poison values, as the honest output follows a distribution, but the injected poison values may locate anywhere. Thus, for a single malicious user, the aggregated value will be larger than it should be, and it may even exceed the upper bound of the input domain. This implies that a small fraction of malicious users can obscure the distribution of honest user's inputs, and this can be extremely fatal when the privacy level is high, or the domain is large. A recent work shows how poor the performance of an LDP protocol can be

under a malicious model [20]. This work proposes a general manipulation attack in which Byzantine users can freely choose to report any poison values in the domain without following a distribution imposed by the LDP perturbation.

A special case of the general manipulation attack is the input manipulation attack, in which adversaries counterfeit some poison values before perturbation and strictly follow the LDP perturbation protocol. This can be treated as a special case of general manipulation attack with strong evasion, as it provides deniability for malicious users. If it is not possible to question individual users, the poison values are also indistinguishable from honest ones. While this evasion makes the poison values harder to detect, it also degrades the strength of the attack compared to general manipulations. This issue has received much attention, and some attempts have been made towards this problem in recent years. [12] proposes a robust defense against poisoning attacks in federated learning systems by leveraging Byzantine-resilient aggregation methods. The authors focus on developing a method to detect and mitigate the impact of attackers who send poisoned model updates during the aggregation process. [21] is an attempt to defend against general colluding attackers in LDP data collection. By exploiting the differences in behavior between attackers and normal users, a maximum likelihood estimation can be utilized to recover an attack distribution based on the collected data. However, this approach has a limitation, as it cannot address situations where attackers intentionally mimic the behavior of normal users. [51] investigates the problem of data poisoning attacks in the context of graph neural networks. While they study the robustness of graph neural networks under poisoning attacks and propose a novel defense mechanism called robust graph convolutional networks, the evasive adversaries are seldom studied and modeled in a comprehensive framework.

2.2 Probabilistic Structures

2.2.1 Bloom Filter

A Bloom filter is a space-efficient data structure used for single set membership testing. It was first proposed by Bloom [11] in 1970 and has found applications in many fields. It is an array of m bits indexed from 1 to m whose values are initially set to 0. For each element x mapped in, a sequence of k uniform, independent hash functions are set to hash the element into an integral value between 1 and m. This value is chosen to be the index number of the Bloom filter's array and the corresponding positions are set to 1. To query whether an element A is in a Bloom filter, one can simply check if all indices are set to 1. If so, outputs "positive", otherwise outputs "negative".

One-side-errors may occur in a Bloom filter, that is, consider an item A appears to be "positive" during a query, which was actually not inserted before. When the set of bits representing A is actually occupied by other elements respectively, the **false positive** occurs. The probability of a false positive happening is described by false positive rate f, which holds the following relationship between f, m, n and k:

$$f = (1 - e^{-\frac{nk}{m}})^k \tag{2.2}$$

Oppositely, no false negative occurs in a Bloom filter, since there is absolutely no element indexed to bits that remain 0.

The Bloom filter has emerged as a pivotal technology in the realm of searchable encryption, facilitating the search of sensitive data in ciphertext. The inaugural endeavor in this field, known as secure indexes [36], has since been followed by further studies [7,65]. Despite their advances, these methodologies fail to distinguish between sensitive and non-sensitive data. Moreover, they do not facilitate the search of non-sensitive data in plaintext, whilst simultaneously offering secure solutions for sensitive

counterparts. There have been several pragmatic attempts to overcome these limitations, with a prominent example being the hybrid cloud approach [62, 63], which solely outsources non-sensitive data. Furthermore, the work of Mehrotra et al. probes into partitioned data security for outsourced data, dividing attributes into sensitive and non-sensitive groups [55]. Although these works introduce partitioned data security and explore deterministic methods, our research concentrates on establishing an efficient probabilistic structure.

There have also been previous efforts to extend the Bloom filter into a matrix form. Geravand et al. proposed a bit matrix to detect copy-paste content in a literature library [35]. Wang et al. introduced a Bloom filter in matrix form, representing a new type of Bloom filter [81]. It employs s rows of Bloom filters with k hashes and an additional special hash to decide which row of the Bloom filter to insert into. The multi-set problem also aligns closely with the bi-attribute membership tests we study. Yu et al. proposed vBF [95] within the framework of BF, creating v BFs for v sets. For a query element a, vBF performs a series of queries across all the v BFs. If BF_i reports true, a is considered to be in set i, otherwise it is not. There are other BF-based solutions to the multi-set problem as well, such as the coded BF [14], the Bloomier filter [15], and kBF [87].

2.2.2 Secure Index

The idea of a secure index [36] is to generate a trapdoor with pseudo-random functions, which are defined later, and insert the trapdoors into a per-document Bloom filter. When a client wants to know which document contains a value he wants, he generates a trapdoor with the value, sends it to the third party where the document is stored and asks it to perform the searching on the trapdoor. Such an index scheme consists of the following four algorithms:

- $Keygen(\lambda)$: Given a security parameter λ, outputs the private key K_{priv}.

- $Trapdoor(K_{priv}, v)$: Given the master key K_{priv} and value v, outputs the trapdoor T_v for v.

- $BuildIndex(D, K_{priv})$: Given a document D and the private key K_{priv}, outputs the index ID.

- $SearchIndex(T_v, ID)$: Given the trapdoor T_v for value v and the index ID for document D, outputs 1 if $v \in D$, otherwise 0.

Pseudo-random functions. A pseudo-random function $f_k(\cdot) : \{0,1\}^n \times \{0,1\}^s \rightarrow \{0,1\}^m$ is said to be efficiently computable, if for any input $x \in \{0,1\}^n$, there exists a key $k \in \{0,1\}^s$, such that:

- With knowledge of k, $f_k(\cdot)$ has a succinct representation and shall be computationally light.

- Without knowledge of k, $f_k(\cdot)$ shall be computationally hard.

2.2.3 Locality Sensitive Hash

The location-sensitive hashing (LSH), proposed by Indyk and Motwani, is a technology mapping similar items into the same hash buckets with high probability [41] and is usually used for the similarity search. Given a similarity search query request, the query item q will be mapped to multiple buckets in the hash table, and we consider items that appear in these buckets are similar points to q. All these similar items will be considered as a group. Generally, the hash function of items close to each other is mapped to the same hash bucket with a higher probability. Let S denote the domain of input items and $||*||$ the distance metric between two items. For any items p and q, LSH function family, i.e., $H = \{h : S \rightarrow U\}$ is called (R, cR, P_1, P_2)-sensitive for

distance function $||*||$ if

$$||p,q|| \leq R \; then \; Pr_H[h(p) = h(q)] \geq \; P_1$$
$$||p,q|| > cR \; then \; Pr_H[h(p) = h(q)] \leq \; P_2.$$

We set $c > 1$ and $P_1 > P_2$ to better do similarity search. The gap between P_1 and P_2 is usually enlarged by using multiple hash functions. Given an d-dimension input item v, different $||*||$ denotes different LSH families which allow the hash function $h_{a,b}$ $R^d \rightarrow Z$ to map v into a hash bucket number. The hash function $h_{a,b}$ in H can be written as:

$$h_{a,b}(v) = \lfloor \frac{av+b}{w} \rfloor,$$

where a is a d-dimension vector randomly chosen from a s-stable distribution, b is a random number chosen from domain $[0, w)$ and w is a large constant.

2.3 Variational Method

2.3.1 Generalized Coordinates

In classical mechanics, the state of a physical system is often described by specifying its position and velocity. However, when dealing with complex systems or those subject to constraints, it's beneficial to use generalized coordinates [48], which are parameters that can describe the configuration of a system using the minimum number of independent variables.

When we refer to a system with s degrees of freedom, we mean the system can move in s independent ways. For example, a simple pendulum has one degree of freedom: the angle from the vertical axis. A double pendulum, however, has two degrees of freedom: the angle of each arm.

The use of generalized coordinates $(q_1, q_2, ..., q_s)$ allows us to express the system's configuration independently of the choice of a coordinate system. This is particularly useful in dealing with systems where Cartesian coordinates are not convenient. The velocities $(\dot{q}_1, \dot{q}_2, ..., \dot{q}_s)$ are the time derivatives of the generalized coordinates and represent how fast each coordinate is changing.

The trajectory of a system in its configuration space is a path that describes how the generalized coordinates change over time. It is analogous to a storyline of the system's motion, and describes where it is and how it moves at every moment in time. The system can be further described using the Lagrangian $\mathcal{L}(q_1, q_2, ..., q_s, \dot{q}_1, \dot{q}_2, ..., \dot{q}_s, t)$. The action, a function of $\mathcal{L}$, is the integral of the trajectory, i.e., $S = \int_{t_1}^{t_2} \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)dt$.

2.3.2 The Least Action Principle

The motion laws of a mechanical system can be expressed using the least action principle. It states that out of all possible paths that a system can take, the actual path is the one that minimizes the action. This requires that the first-order variation of the action with respect to the generalized coordinates and velocities equals zero:

$$\delta S = \delta \int_{t_1}^{t_2} \mathcal{L}(q_1, q_2, ..., q_s, \dot{q}_1, \dot{q}_2, ..., \dot{q}_s, t)dt = 0, \tag{2.3}$$

where δ is the variation of S, i.e., an infinitely small incremental change to a function. The solution to equation 2.3 yields the Euler-Lagrange equation, whose proof can be found in any classical mechanics textbook:

Lemma 1. *(Euler-Lagrange Equation). A necessary condition for $\delta S = 0$ is:*

$$\frac{\partial \mathcal{L}}{\partial q_i} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) = 0, \quad i = 1, 2, ...s. \tag{2.4}$$

The Euler-Lagrange equation is a set of s second-order ordinary differential equations, which govern the motion of the system. The equation describes how the system evolves over time, given the initial conditions of the generalized coordinates and velocities.

Chapter 3

Cloud Password Shield

With the coming era of IOT, more and more devices are tightly connected to the Internet in our day-to-day life. However, the rapid growth on the number of insecure devices also aggravates the power and scale of DDoS attacks, which stop legitimate users from accessing certain network services. These devices are vulnerable to malwares (e.g., backdoors and Trojans), which infect them and turn them into bots. The Mirai DDoS attack on Dyn in 2016 demonstrated the scale and severity of such attacks. Although network and cloud firewalls (e.g., Cloudflare) can mitigate TCP or IP flooding, DDoS attacks can occur in the application layer, for example, by flooding requests to the user login interfaces to DDoS the authentication service. Recently, two Russian banks became the victim of such DDoS attacks using Medusa, a brute-force web login toolkit.[1].

The vulnerability roots from the fact that most password-based authentication service, e.g., a RADIUS server, is powered by a relational database of users' credentials. Such a centralized system cannot scale well in a large system (e.g., Internet and IoT). When coordinated attackers deliberately send incorrect login requests to server at the

[1]http://www.infosecisland.com/blogview/24848-Medusa-DDoS-Botnet-Slams-Russian-Banks.html

same time, the database can exhaust its I/O resources to process all these queries and thus fail to respond to legitimate login requests.

Unfortunately, existing DDoS mitigation services (e.g., Cloudflare) can only limit the access rate of the login page at TCP/IP layer, which also affects the legitimate users. In this chapter, we propose to install Bloom filter, a structure with $O(1)$ time complexity for existence query, on cloud servers (i.e., firewalls) in front of the authentication server as a pre-screener to drop requests with wrong passwords before they reach the latter. The main challenge is that if such server is compromised and the Bloom filters are retrieved by an adversary, he/she might conduct offline brute-force attacks and restore (hashed) passwords. We set out to design the pre-screening mechanisms in two different security contexts, based on whether the firewall provider can be trusted with a shared key from the authentication server. In both contexts, our proposed pre-screener mechanisms mitigate malicious requests reaching the authentication server while not degrading the security of passwords against offline brute-force attackers.

The contributions of this chapter are listed as follows:

- **Bloom filter as a pre-screener.** We present a password pre-screener using a global Bloom filter to provide DDoS resistance for password-based authentication servers. The pre-screener supports all operations of an authentication server, i.e. setup, insertion, query and deletion. Upon these fundamental settings, the security issue is still to be developed, to differentiate whether the firewall provider is trusted or not.
- **KSSBF: key-based secure solution for trusted firewall providers.** For those firewall providers who can be trusted with a shared key for encryption/decryption, we design a key-based pre-screening Bloom filter which achieves semantic security against chosen password attacks.
- **GSBF: generically secure solution for non-trusted firewall providers.**

For those firewall providers who cannot be trusted, we design a Bloom filter that is unprofitable to anyone, thus appears to generically secure without relying on a private key. We show that false positives in a Bloom filter, which is commonly believed undesirable, can be work against offline brute-force attackers who have access to the Bloom filter after compromising the cloud server. We prove that in the context of indistinguishability, the gain of any attacker exponentially drops to zero with respect to the number of passwords inserted to this Bloom filter.

The rest of this chapter is organized as follows. In chapter 3.1, an overview of the system model is given to show some fundamental settings of the Bloom filter. In chapter 3.2 and chapter 3.3, security models of the Bloom filter are constructed, namely KSSBF and GSBF, depending on whether the firewall provider is trusted or not. In chapter 3.4, the performance of our firewall model is evaluated experimentally. Finally, the conclusion is drawn in chapter 3.5.

3.1 System model: an overview

In this section, we will discuss the possibility to tolerate DDoS against authentication services by introducing a Bloom-filter-based layer between the back end and the front end, which behaves like a firewall for password verification, as shown in Fig. 4.2. When dealing with login attempts, this layer is expected to effectively reduce malicious traffic to the back end, essentially thwarting attempts to overload the server with candidate passwords generated randomly. The firewall can be run by either individuals, or an enterprise who offers cloud services to a series of password authentication servers, hence is to be discussed whether it is trusted or not. Let's take an overview of the system model in this section, and remain the detailed discussions to the following two sections.

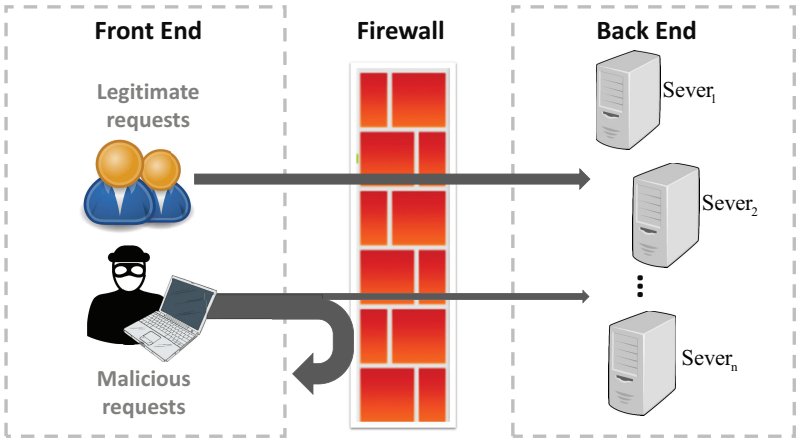


Figure 3.1: A Bloom filter-based firewall, alleviating workload of protected back ends.

3.1.1 Bloom filter as a pre-screener

3.1.1.1 Why Bloom filters

To achieve the motivations mentioned, the following properties are expected to be hold:

- Small storage cost.
- Computationally light for insertion/query.
- Negative results should be correctly confirmed, whereas positive results are tolerated to be falsely judged, as eligible users shall pass the filter accurately.
- Even if the new layer is exposed to an off-line attacker, the security level of the password-hashing-based back end doesn't degrade.

Intuitively, if both f and the Bloom filter runtime are set small enough, using a Bloom filter can easily meet the first three conditions:

- A Bloom filter is space-efficient.

- A Bloom filter index can be computationally light.
- One-side-error appears in Bloom filters, which only permits occurrence of false positives.

As for security issues, the case is rather complicated, since the new layer should be carefully designed, to guarantee its ability to block off most of the malicious requests, without being exploited by off-line attackers who try to gain the password list. Let's remain them to the latter sections, where security models will be defined, and provable security will be provided in detail.

3.1.2 Bloom filter settings

Informally speaking, our general idea is to record the registered passwords into a global Bloom filter, and to query them quickly before accessing the back end. Negative results shall be immediately rejected, whereas positive ones are to be sent to the back end and in turn complete the full authentication process, as shown in Algorithm 1.

Algorithm 1 Login response of a server with a Bloom filter

```
1: Input password pw;
2: if BF_pw=false
3:     then Return false
4:     elseif Backend_pw=false
5:         then Return false
6:     else Return true
```

To quantify our demand of the Bloom filter, let's define bad request rate λ to describe the bad request proportion of an incoming login request flow[2]. Apparently, when

[2]Note that, even for a group of legitimate users, λ should still expected to be non-zero, for it ought to tolerate a certain degree of bad requests, as there may be typos for regular users, or they may desire to recall forgotten passwords by retrying.

malicious traffics are involved, λ begins to increase, and raises an alarm for a launched DDoS attack when rising to some certain level. Upon this definition, our goal is to design a Bloom filter, such that:

- In the view of a server, when λ is large, the average response time of determining an item in the back end shall be lowered, so is possible to survive the heavy workload under a DDoS attack.
- In the view of the entirety of (server, Bloom filter), when λ is small, the overall average time of determining an item shall acceptably increase (preferably none), so limited overhead is involved during regular cases.
- The additional Bloom filter shall not compromise the security level of the (password hashing-based) back end.

For the convenience of discussion, let's further define the variables that are to be used. Denote the average time of querying an item in the Bloom filter by t_{BF} and that in the back end by t_{BE}. Let t_a be the overall average time of determining an item, t_b be the average time of querying an item in the back end. The time caused by an item should be counted in, even if the item was filtered out by the Bloom filter and cause a t_{BF}.

Runtime in average. The runtime of query in the Bloom filter shall be much less than that of the back end to ensure a quick response. Here, let's derive a series of generic details on the runtime of Bloom filters and that of the back end.

Let's firstly determine t_a. Clearly, for negative results, the cost is simply t_{BF}. For positive results, since they are to be checked into the back end again, the cost turns out to be $t_{BF} + t_{BE}$. Notice that, if λ is given, in the Bloom filter, the "return true rate" shall be $1 - \lambda + f\lambda$, since it returns true for the $(1 - \lambda)$ who are actually true, but there still exist a false positive rate f on the remaining λ, which results in an additional $f\lambda$ on the "return true rate". Oppositely, the "return false rate" turns out

to be $1-(1-\lambda+f\lambda)=(1-f)\lambda$. Then, the expectation value of t_a holds:

$$\begin{aligned} t_a =& (1-f)\lambda t_{BF} + (1+f\lambda-\lambda)(t_{BF}+t_{BE}) \\ =& t_{BF} + (1+f\lambda-\lambda)t_{BE} \end{aligned} \tag{3.1}$$

with a clear, manifest interpretation: For each incoming request, whether it is a true one or not, it shall be checked into a Bloom filter firstly and consume a runtime of t_{BF}. Then, only a part of them (return true ones) arrives at the back end, which results in release of computation of the back end. Accordingly, the expectation value of t_b holds:

$$t_b = (1+f\lambda-\lambda)t_{BE} \tag{3.2}$$

which is strictly less than t_{BE} and strictly decrease with respect to λ.

Password capacity. Then, let's decide the relationship between the approximate passwords capacity and the Bloom filter size. As mentioned, the Bloom filter starts at a certain state, and is filled up as users register and input passwords. With the Bloom filter getting full (or full enough to start giving out many false positives), the false positive rate rises and the performance of Bloom filter tends to worsen. Assume the initial false positive rate is f_1, and the worst acceptable false positive rate is f_2. Hence, the Bloom filter is expected to work between f_1 and f_2. When both m and k are pre-given, according to eq. 2.2, the following formula describing the relationship between n and f holds:

$$n = -\frac{m}{k} ln(1-e^{\frac{lnf}{k}})$$

Then we have the capacity:

$$N = n_2 - n_1 = -\frac{m}{k} ln \frac{1-e^{\frac{lnf_2}{k}}}{1-e^{\frac{lnf_1}{k}}} \tag{3.3}$$

Password revision/deletion. Finally, let's discuss what happens when a user changes his password, particularly in a scenario where an organization requires all its users to change passwords periodically as per policy. When deleting a record from the back end, it is straightforward to delete the corresponding record inside the

pre-screener concurrently. However, Bloom filters do not support deletions, with the exception of counting Bloom filters. But, even using counting Bloom filters does not solve this problem, as false negatives may occur after deletion: If overflow happens in some counter, the actual number of elements recorded turns out to be greater than what the upper bound is. When deleting elements, non-zero counters may be deleted to zero much earlier than it should happen, which results in possible invalidity of correct passwords.

Actually, we don't have to face such an intractable problem, because false positives are initially tolerable in our mechanism. The password after revision can be directly written into the Bloom filter, as if a new-comer who does a first-time registration, which is unnecessarily to be distinguished from the former in the pre-screener layer. The initial record inside the Bloom filter need not to be erased, if enough bits are initially allocated for the Bloom filter. Furthermore, this could be further improved by using adaptive sizing mechanisms or layered Bloom filters, which involves implementing a system of layered Bloom filters, where each layer represents passwords changed within a specific time frame and older layers are gradually phased out. In the end, instead of trying to delete individual entries, the entire Bloom filter could be regenerated periodically, perhaps aligning with the password change cycle.

In fact, in the latter section of GSBF, it can be seen that the security is guaranteed by a moderately high false positive rate. Hence, f_1 is required to be set to a certain value by an obfuscation operation. In this process, a great deal of incorrect passwords are to be uniformly inserted into the Bloom filter, which is also unnecessarily to be distinguished from the "rubbishes" remained by the revised passwords. In the end of the beginning, the notations used in this chapter is listed in Table 3.1 for the ease of understanding.

Notation	Description
f	False positive rate of the Bloom filter
t_{BF}	Average time of querying an item in the Bloom filter
t_{BE}	Average time of querying an item in the back end
t_a	Overall average time of determining an item
t_b	Average time of querying an item in the back end (considering the Bloom filter)
N	Password capacity in the Bloom filter
λ	Bad request rate, describing the proportion of bad requests
K_{priv}	Administrative private key
pw	Password input by the user
BF_{hash_i}	Hash function used in the Bloom filter
$trapdoor$	Encrypted representation of a password in the Bloom filter
s	Security seed used for generating the private key
f_1	Initial false positive rate
f_2	Maximum acceptable false positive rate
m	Size of the Bloom filter in bits
k	Number of hash functions used in the Bloom filter
b_i	A specific bit position in the Bloom filter

Table 3.1: Table of Notations

3.2 KSSBF: A key-based secure solution for trusted firewall providers

In this part, we assume the firewall provider is deemed trusted. This trust is crucial because it allows the firewall to securely manage private keys necessary for encryption and decryption operations. Typically, the firewall provider and the protected server belong to the same organization or trusted partnership, ensuring that sensi-

tive data handling remains secure. This enables the implementation of a key-based pre-screening Bloom filter without compromising security.

As mentioned, to defend against a DDoS attack, the pre-screener is required to be computationally light in the view of the server. However, to defend against an off-line attacker who breaks in the server and gains everything from the screener, it is required to be computationally hard in the view of any attacker. To achieve this goal, a straightforward consideration is to encrypt the password plaintexts into trapdoors that are indistinguishable to those without knowledge of the private key. The Bloom filter insertions/queries are processed on the trapdoors, to ensure the two-sided computational hardness.

Upon encryption, we are able to fight against an attacker who holds unbounded ability to access the Bloom filter, that is, he is able to insert/query anything he wants unlimitedly from the publicly given algorithms (To describe it formally, we need to define a semantic security against chosen password attack in this scene, which is to be discussed in details later in this section). Note that, in this design, the Bloom filter reveals nothing without knowledge of a private key, hence can be distributed to a non-trusted environment (say, a plug-in attached to the front end) and to be updated timely[3].

In this section, we will discuss the case when the firewall provider is deemed trusted, so is permitted to preserve the private key. In practical use, it can be true when the firewall provider and the protected server are of the same party, e.g. assume the server-side owns two servers, one of which is regarded absolutely secure and deemed appropriate to safeguard the private key. Indeed, in this model, the firewall layer plays a role of trapdoor generator, as calculation of trapdoors always invoke the private key. Here, the runtime of en/decryption is assumed to be negligible, though a delay of respondence, approximately equals to t_{BE}, shall be employed to avoid abuse

[3]It leads to synchronization of the front end and the back end, but is tolerable if the registrations/password revisions are not frequent.

of this mechanism. An overall framework of the system architecture is shown in Fig. 3.2, which is to be described in details in this section.

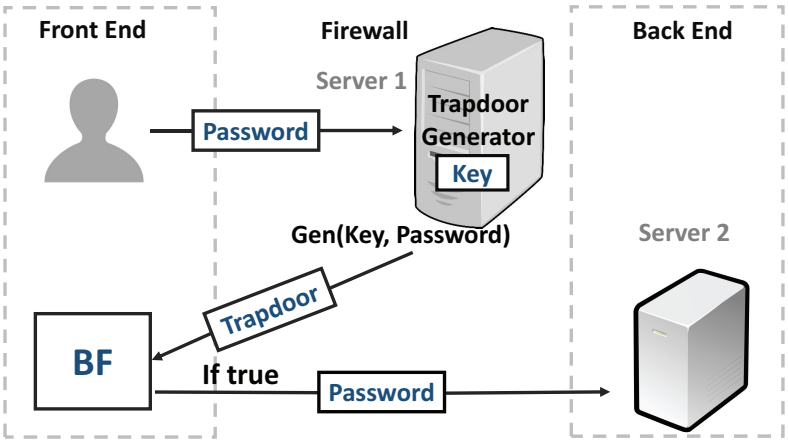


Figure 3.2: An overview of a password-hashing authentication service with KSSBF.

3.2.1 Construction

KSSBF can be constructed from the following four algorithms:

- $Keygen(s)$: Given a security seed s, output the administrative private key K_{priv} with respect to s.
- $BF_{insertion}$: Given an item, hash it into the corresponding bits and set them to 1.
- $Trapdoor(K_{priv}, password)$: Choose a pseudo-random function $f_K(\cdot)$. Given an initialized Bloom filter, a valid password, and the administrative private key K_{priv}, output the trapdoor as $T = f_K(password, K_{priv})$.
- BF_{query}: Given an item, hash it into the corresponding bits. Return positive if all of them are 1, otherwise return negative.

Algorithm 2 KSSBF registration

1: User inputs password pw and send it to Server 1
2: Server 1 generates a security seed s
3: $K_{priv} \leftarrow Keygen(s)$
4: $trapdoor \leftarrow Trapdoor(K_{priv}, pw)$
5: Server 1 sends $trapdoor$ to Front end
6: **for** i=1 to k
7: $\quad BF_{hash_i}(trapdoor) \leftarrow 1$
8: **end**

At ordinary times, when users register their passwords into the back end, the KSSBF shall also update. Concurrently, the pre-screener also outputs a trapdoor, and records it into the Bloom filter, following Algorithm 6. When a login request comes, users send their passwords to Server 1. Server 1 also outputs a trapdoor according to the password and its private key, delay a t_{BE}, then send it back to the Bloom filter at the front end. The Bloom filter queries the trapdoor, if true, returns the password to Server 2, otherwise rejects it immediately. This process is described by Algorithm 7.

Algorithm 3 Login response of a server with KSSBF

1: User inputs password pw and send it to Server 1
2: Server 1 generates a seed s
3: $K_{priv} \leftarrow Keygen(s)$
4: $trapdoor \leftarrow Trapdoor(K_{priv}, pw)$
5: Delay t_{BE}
6: Server 1 sends $trapdoor$ to Front end
7: **if** $BF_{query}(trapdoor)$=false
8: $\quad$ **then** Return false
9: $\quad$ **else** Send pw to Server 2

This simple but powerful solution can be made highly ideal, by choosing a t_{BF} (and a corresponding f) as small as possible. If both of them are negligible, according to

eq.3.1 and eq.3.2, both t_a and t_b turns out to be approximately $(1-\lambda)t_{BE}$, which implies:

- Almost no overhead in regular cases: If $\lambda \approx 0$, $t_a \approx t_{BE}$.
- Highly resistant against DDoS attacks: If $\lambda \approx 1$, $t_b \approx 0$.

3.2.2 Security model

3.2.2.1 Pseudo-random functions

Loosely speaking, a pseudo-random function, mapping n bit inputs to n bit outputs, is one that can not be computationally distinguished from a truly random function. More precisely, a function ensemble $\mathbb{F}^n$ is said to be pseudo-random, if for every probabilistic polynomial-time oracle machine D, every polynomial $p(\cdot)$ and all sufficiently large n's, the following formula is always satisfied:

$$|Pr[D^{\mathbb{F}^n}(1^n) = 1]| - |Pr[D^{\mathbb{H}^n}(1^n) = 1]| < \frac{1}{p(n)}$$

where $\mathbb{H}^n$ denotes truly random function.
A pseudo-random function $f_k(\cdot) : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ is said to be **efficiently computable**, if for any input $x \in \{0,1\}^n$, there exists a key $k \in \{0,1\}^s$, such that:

- With knowledge of k, $f_k(\cdot)$ has a succinct representation and shall be computationally light.
- Without knowledge of k, $f_k(\cdot)$ shall be computationally hard.

3.2.2.2 Semantic security against chosen password attack

In this security model, we consider the chosen password attack as security threats as it addresses scenarios where adversaries attempt to exploit the system by submitting

specific passwords and analyzing the responses. By designing the system to withstand such attacks, we ensure that attackers gain no meaningful advantage, even when they can choose passwords to test. This enhances the robustness of the Bloom filter pre-screener, providing additional security to the authentication process.

Intuitively, our goal is to illustrate and describe the concept that an adversary $\mathcal{A}$ can not deduce a registered (user-name, password) pair from the Bloom filter (which has a lower runtime than that of the back end), even if some of the historical results are given. Loosely speaking, suppose the adversary $\mathcal{A}$ gives the challenger $\mathcal{C}$ two randomly picked passwords, say, (b_1, b_2). $\mathcal{C}$ chooses one from b_1 and b_2 in equal-probability, say, b', then encode b' into a initialized Bloom filter according to some public insertion algorithms, and send the result to $\mathcal{A}$. $\mathcal{A}$'s challenge is to guess whether $b' = b_1$ or $b' = b_2$. If the mission of determining the Bloom filter inserting b_1 and b_2 is computationally hard, the Bloom filter reveals nothing about the inserted contents, hence is unprofitable in the view of an off-line attacker. In this case where $\mathcal{A}$ is not able to determine which Bloom filter holds b_1 or b_2, the probability of $b' = b_1$ and $b' = b_2$ shall be non-negligibly from $\frac{1}{2}$, hence they are computationally indistinguishable in the view of $\mathcal{A}$.
More precisely, we use the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ to define semantic security against a chosen password attack:

- **Setup.** The challenger $\mathcal{C}$ builds an initialized Bloom filter, then chooses a set of valid passwords and give them to the adversary $\mathcal{A}$. $\mathcal{A}$ chooses 2 of them, say, b_1 and b_2, and return them to $\mathcal{C}$. $\mathcal{C}$ choose one from them in equal-probability, insert it into the Bloom filter, and return the Bloom filter to $\mathcal{A}$.

- **Challenge.** $\mathcal{A}$ is permitted to make any insertion, query inside the Bloom filter, according to a pre-given, public insertion/query algorithm.

- **Response.** $\mathcal{A}$ eventually outputs a result b', representing its guess for b. The advantage of $\mathcal{A}$ winning this game is defined as $Adv_{\mathcal{A}} = |Pr[b = b'] - \frac{1}{2}|$.

If $Adv_{\mathcal{A}} < \epsilon$ is satisfied for any possible adversary, where ϵ is almost negligible in polynomial time, we say the Bloom filter is **semantically secure** against chosen password attack. Upon usage of pseudo-random function with key, the semantic security can be achieved. The proof is trivial, and is omitted here.

3.3 GSBF: a generically secure solution for non-trusted firewall providers

Although the performance of KSSBF is regarded highly satisfactory, it relies on a trusted firewall provider to keep privacy of the key. Once a trusted environment to preserve the key no longer exists, or revealed to be insecure, the Bloom filter turns out to be exploitable to attackers and in turn breaks the security of this scenario.
In this section, we will discuss the case when firewall providers are deemed untrusted, i.e. run by a third-party enterprise. Apparently, it's not applicable to authorize the key (or the password list) to a non-trusted third-party. Our solution is to design a Bloom filter that is regarded unprofitable unconditionally, so is able to undertake the heavy workloads for the protected servers in an untrusted environment.

3.3.1 On false positives

In traditional applications of a Bloom filter, false positives, regarded as a negative factor, are deemed to be "the smaller, the better", likewise to the runtime t_{BF}. Indeed, in KSSBF, they are assumed to be small and omitted.
Here, let's start from a counterintuitive question: With a tolerable overhead, can large false positives (as well as a large t_{BF}) be helpful in the aspect of security? Note that, even if someone manage to find a password that is inside the Bloom filter, it could be a false positive. Roughly speaking, in the view of off-line attackers, the larger f is, the less profit they gain from this Bloom filter, as they have to pick out the true

positive ones in the back end. Once t_{BF} is also large enough, t_a can be even worse than directly checking in the back end.

It is reasonable to deduce, if t_{BF} and f are set large enough, it is possible to render any potentially existing attackers gain nothing from the Bloom filter. However, without a key, the computation is no longer two-sided, which also comes to be true in the view of the entirety of (pre-screener, server). In this case, the pre-screener no longer alleviates the overall workload. It blocks majority of malicious traffics and alleviates workload of those protected back ends, at the expense of a greater overall computation, most of which is undertaken by the firewall. Hence, t_{BF} can't be any large, or it will cause a large overhead that the system could hardly bear. Here, a trade-off between security, DDoS resistance, and the overhead during regular cases should be carefully considered.

3.3.2 Security model

Semantic security is strong enough to defend against attackers who try to benefit from the Bloom filter. However, it is regarded somewhat overqualified in the context of our pre-screener, since the Bloom filter is only required to "not degrade the security level of the back end", but not "more secure than it". Roughly speaking, in the view of any outside attacker, if the overall average runtime of testing an element with the additional Bloom filter is greater than not, he is not motivated to do so. Hence, it is reasonable to believe that the Bloom filter brings no extra security concerns, as if it doesn't exist. For the convenience of discussion, we strictly define the notion of nothing-to-gain as follow:

Definition 1. ***Nothing-to-gain.*** *For attackers who successfully break in the server and try to achieve the password plaintexts from organizing a brute-force attack, if it costs less time for them to use data or mechanisms from the back end directly, that is, in the view of an arbitrary attacker, the overall computation resource (measured*

by the runtime of the algorithm) consumed in the former is greater than that in the latter, we say it is ***nothing-to-gain*** *from the Bloom-filter.*

Obviously, if a Bloom filter is semantically secure in the view of an attacker, it also appears to be nothing-to-gain for him.
Upon the definition of nothing-to-gain, let's derive the boundary condition, where the Bloom filter can reject false request more quickly (possibly cause overhead during regular cases), but does not degrade the security against off-line attackers:

Theorem 1. *A Bloom-filter as a pre-screener appears to be nothing-to-gain for any potential off-line attacker, if the following condition is met:* $t_{BF} \geq (1-f)t_{BE}$

Proof. Obviously, for the original scheme without a Bloom filter, the average time of determining an item is t_{BE}. If this choice costs less time for potential attackers, they are not motivated trying to gain the Bloom filter. In this situation, the following condition must be satisfied:

$$t_a \geq t_{BE} \tag{3.4}$$

In the view of attackers who try to gain information from the Bloom filter by brute-force, that is, traversing all possible inputs by querying them into the Bloom filter, let's calculate t_a and compare it with t_{BE}. Since most of them are incorrect ones, this is in fact equivalent to testing it by a flow with $\lambda \approx 1$. In this case, we have the expectation value of t_a by substituting $\lambda = 1$ into eq.3.1:

$$t_a = (1-f)t_{BF} + f(t_{BF} + t_{BE}) = t_{BF} + ft_{BE} \tag{3.5}$$

Substitute by eq.3.4, we have:

$$t_{BF} \geq (1-f)t_{BE}$$

□

As discussed, increase on t_{BF} also enhance the overhead. To achieve the best overall performance, let

$$t_{BF} = (1 - f)t_{BE} \tag{3.6}$$

3.3.3 Construction

3.3.3.1 Runtime of GSBF

Let's look back and decide t_a and t_b. Substitute eq.3.6 into eq.3.1 and eq.3.2, we have:

$$t_a = [(2 - f) + (f - 1)\lambda]t_{BE}$$

$$t_b = [(1 + (f - 1)\lambda]t_{BE}$$

Notice that t_a and t_b are linearly related to λ, given $f = 0.1, 0.5, 0.9$, respectively, we show their tendency varying with respect to λ in Fig.3.3, where t_a is represented by full lines and t_b is represented by dotted lines. The vertical axis denotes the times of t_{BE}, in the scope of $(0, 2)$. As shown in the picture, both security and DDoS resistance could be achieved, if a overhead (equals to $t_a - t_{BE}$) on the overall performance could be tolerated. Apart from that, the greater f is, the less DDoS resistant it is, accordingly the less overhead it bears.

3.3.3.2 Initialization

Each GSBF shall be initialized before using. In this process, an additional algorithm $Obfuscation$ is needed. As mentioned in the context, apparently, if a moderately

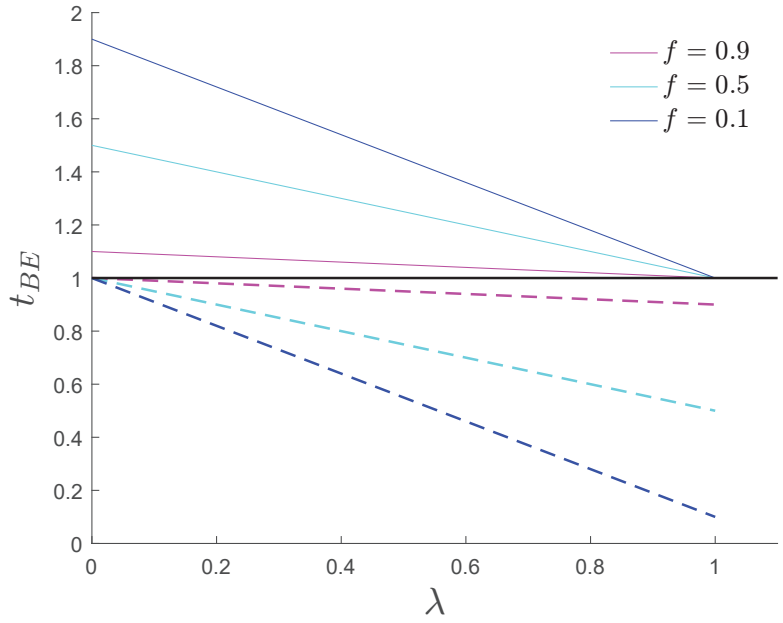


Figure 3.3: t_a and t_b varying with λ. f is set to 0.1, 0.5, 0.9, respectively.

high false positive rate f_1 is required at the beginning, the Bloom filter does not start out empty. To achieve an initial f_1, we choose a naive method, that is, obfuscating it by uniformly setting some of its bits to 1, until the overall false positive rate arrives at the requiring f_1, which is equivalent to insert a series of random passwords into the Bloom filter, but without any additional calculations. The initialization phase is described as Algorithm 4:

To illustrate the feasibility of this strategy, as a rough estimation, let f_1 be $ln1.44 \approx 0.36$, f_2 be $ln1.96 \approx 0.67$, $k = 2$ and $m = 1MB = 8,388,608bits$. Substitute them into eq.4.2.4.1, we have:

$$N = n_2 - n_1 = -\frac{m}{2} ln \frac{1 - 1.2}{1 - 1.4} = -\frac{m}{2} ln 0.5 \approx 2,900,000$$

which implies a rather small space is enough to support millions of password revisions/new registrations.

Algorithm 4 Initialization of GSBF

1: Input average time of decision in the back end t_{BE}
2: password capacity N
3: initial false positive rate f_1
4: maximum false positive rate f_2
5: Set t_{BF} according to eq. 3.6
6: Set m according to eq. 4.2.4.1
7: **while** $f < f_1$
8: Select a random bit b_i in GSBF, $b_i \leftarrow 1$
9: **end**

3.3.3.3 Login response

It can be seen from Fig. 3.4, both the overhead and t_b strictly decrease with respect to λ, which implies it costs less time to use this mechanism when λ is large. To achieve the best profit, as described in Algorithm 5, it is wise to pre-determine an alarm threshold λ_0, such that the back end is able to work when $\lambda < \lambda_0$, and run it in an adaptive patten:

- When $\lambda < \lambda_0$, the workload of the back end is acceptable, inactivate the pre-screener and directly send the incoming login requests to the back end to avoid the heavy overhead
- When $\lambda > \lambda_0$, the workload of the back end starts to be heavy, activate the pre-screener to reduce the pressure of the back end.

3.3.4 Provable security of GSBF

In this subsection, we will provide provable security of GSBF in the sense of indistinguishability. Intuitively, a large f contributes to this kind of security: Consider a

Algorithm 5 Login response of a server with GSBF

```
Input password pw, DDoS alarm bound λ_0
if λ < λ_0 then
    Send pw to back end
    Return 0
else
    Send pw to GSBF
if BF_query(pw)=false
    then Return false
    else Send pw to back end
```

computationally unbounded challenger $\mathcal{C}$ in decoding a password from a Bloom filter, such that t_{BF} is always ignorable in the view of $\mathcal{C}$. Even if this is true, to finally decide which one is correct, he still has to compare the two results computed from the Bloom filter. However, as there exists false positives, if both of them appears to be positive, they are still indistinguishable in the view of $\mathcal{C}$. The higher f is, the less information the Bloom filter could offer, which is easy to be understood by an extreme case: Consider a Bloom filter that is entirely set to 1 and returns positive to everything, hence it provides no information, as if doesn't exist. Indeed, in the context of indistinguishability, this feature behaves like a background noise, which is relevant to nothing but f. To make it precise, we have the following definition and theorem:

Definition 2. *Adversary's advantage. Let $\frac{1}{2} + \epsilon'$ be the probability of an adversary's ability to judge b' from b according to any auxiliary knowledge he holds, where ϵ' is defined as the adversary's advantage.*

Theorem 2. ***f^2-background noise on indistinguishability.*** *Consider an initialized Bloom filter with false positive rate f. The semantic security requires that even if an adversary $\mathcal{A}$ holds a non-negligible advantage ϵ' from his ability of efficiently decoding the Bloom filter, the actual advantage shall be lowered to at least*

$(1 - f^2)\epsilon'$.

Proof. Let's look back to the game where semantic security was defined. Consider an adversary $\mathcal{A}$ who holds a non-negligible advantage ϵ' to judge b' from b, that is, in probability $\frac{1}{2} + \epsilon'$, it is computationally light for $\mathcal{A}$ to decode b_1 and b_2 in that Bloom filter and judge which one has been inserted in.
However, even if he is computationally able to do so, the result could be a false positive, and it is still indistinguishable for $\mathcal{A}$ if both of the results return positive (one or more of them appears to be false positive). The corresponding probability turns out to be f^2, which results in an actual appearance of advantage $(1 - f^2)\epsilon'$ □

Upon this theorem, we claim that the semantic security of KSSBF is in fact double-guaranteed by existing false positive rate f: Even if the pseudo-randomness of function $f(\cdot)$ is weakened by some prior knowledge, or the privacy of the private key K_{priv} is broken, the background noise still exists, rendering the security level "not that bad".
Here, we also prove a simple deduction from theorem 2 and eq.2.2, which illustrates the strong security of GSBF's working patten:

Theorem 3. *Due to the existence of f^2 background noise, for any adversary $\mathcal{A}$ who holds an non-zero advantage ϵ', ϵ' drops to 0 exponentially, with respect to n.*

Proof. Consider an initialized Bloom filter, where both m and k are given. With the Bloom filter getting full, f varies with respect to n. Substitute eq.2.2 into $(1 - f^2)$, we have:

$$(1 - f^2) = 1 - (1 - e^{-\frac{nk}{m}})^{2k}$$

When n becomes large, $e^{-\frac{nk}{m}}$ is close to 0, hence, $(1 - e^{-\frac{nk}{m}})^{2k} \approx 1 - 2ke^{-\frac{nk}{m}}$. Then, we have:

$$(1 - f^2) = 1 - (1 - e^{-\frac{k}{m}n})^{2k} \approx 2ke^{-\frac{k}{m}n}$$

which drops exponentially with respect to n, if m and k are pre-given. □

3.4 Performance evaluation & Experiments

3.4.1 Performance metrics

As shown in Table 3.2, there are four possible outcomes from the pre-screener, donating to true negative, false negative, false positive, true positive, respectively.

Table 3.2: Outcome of the pre-screener.

DDoS defense decision \ Desirable decision	Negative	Positive
Negative	A	B
Positive	C	D

Theoretically, according to the definition of λ, in our model, the desirable decision of positive and negative shall be $(1 - \lambda)$ and λ. However, due to the false positive of Bloom filters, there exists a $f\lambda$ of false positives among the λ negative ones, hence, $C = f\lambda$. Accordingly, $A = \lambda - f\lambda$. Obviously, $B = 0$, as no false negatives occurs in a Bloom filter. Then, we have $D = 1 - \lambda$. For the convenience of discussion, the results are listed in Table 3.3.

Six metrics constituted from elements in Table 3.2, which have been previously introduced in literature [96], can be employed to evaluate DDoS defence mechanisms quantitatively. Let's simply quote the existing results and substitute elements in Table 3.3 to them:

Table 3.3: Theoretic value of Table 3.2.

DDoS defense decision \ Desirable decision	Negative	Positive
Negative	$\lambda - f\lambda$	0
Positive	$f\lambda$	$1 - \lambda$

- Accuracy: $(A + D)/(A + B + C + D) = 1 - f\lambda$
- Sensitivity: $D/(B + D) = 1$
- Specificity: $A/A + C = 1 - f$
- Precision: $D/C + D = (1 - \lambda)/(1 - \lambda + f\lambda)$
- Reliability: $C/C + D = f\lambda/(1 - \lambda + f\lambda)$
- False negative rate: $B/(A + B) = 0$

3.4.2 Performance evaluation

Note that, the six metrics are constituted from elements of Table 3.3. Hence, the performance evaluation is essentially verification of elements in Table 3.3. We just run the experiment on a PC, and compare the results with Table 3.3.

3.4.2.1 Configurations

The CPU is Intel(R) Core(TM) i5-7500 @ 3.40 GHz, and the memory is 16.0 GB. Set the overall size m of the Bloom filter to $m = 1MB = 8388608bits$. Three commonly used uniform string hashes (BKDR, EK, PJW) are employed in the Bloom filter, hence, $k = 3$.

3.4.2.2 Experimental results of elements in Table 3.3

As shown in Table 3.4, the theoretical value vs. experimental value of elements in Table 3.3 are listed to make comparison. The case when $f = 0.005$ corresponds to the KSSBF, and the remaining three correspond to GSBF with different false positive rates. Since $B = 0, D = 1 - \lambda$ are constants that are irrelevant to the Bloom filter, they are not necessary to be tested. Hence, only results of A and C are shown in this Table.

3.4.3 Registration/password revision performance of KSSBF

In the system model section, we claimed that the Bloom filter works in the scope of false positive rate f_1 and f_2, and give out a study case in the GSBF section to illustrate the feasibility. In this section, we will verify the registration/password revision performance of KSSBF experimentally.
In this experiment, the Bloom size still remains 1MB. To keep the false positive rate in a relatively small area, let the number of passwords recorded into KSSBF be 100000, 200000, 300000, 400000, 500000, respectively, and evaluate if their performances are steady. We can see from Fig. 3.4, in the scope of a negligible false positive rate, the performance of KSSBF still holds, even if the number of the recorded passwords rise to 500000.

3.4.4 Scalability

A scalable DDoS defence mechanism shall effectively handle its duty, even if the amount of malicious traffic increases. In this part, we will verify the scalability of both KSSBF and GSBF to test their stress tolerance.
Assume the maximum possible workload of the back end is 500000 requests per second. With initialized Bloom filters who holds determined false positive rates (f=0.001

Table 3.4: Comparison between experimental and theoretical value.

Attribute \ λ	0.01	0.1	0.5	0.9
f=0.005				
A_{theo}	0.001	0.01	0.05	0.09
A_{exp}	0.0013	0.0136	0.0677	0.122
C_{theo}	0.009	0.09	0.45	0.81
C_{exp}	0.00863	0.0863	0.432	0.777
f=0.1				
A_{theo}	0.009	0.09	0.45	0.81
A_{exp}	0.00987	0.0985	0.492	0.886
C_{theo}	0.001	0.01	0.05	0.09
C_{exp}	0.000126	0.00148	0.00737	0.0132
f=0.5				
A_{theo}	0.005	0.05	0.25	0.45
A_{exp}	0.00503	0.0504	0.252	0.454
C_{theo}	0.005	0.05	0.25	0.45
C_{exp}	0.00496	0.0496	0.247	0.445
f=0.9				
A_{thro}	0.001	0.01	0.05	0.09
A_{exp}	0.00137	0.01368	0.06779	0.122
C_{thro}	0.009	0.09	0.45	0.81
C_{exp}	0.00863	0.0863	0.432	0.777

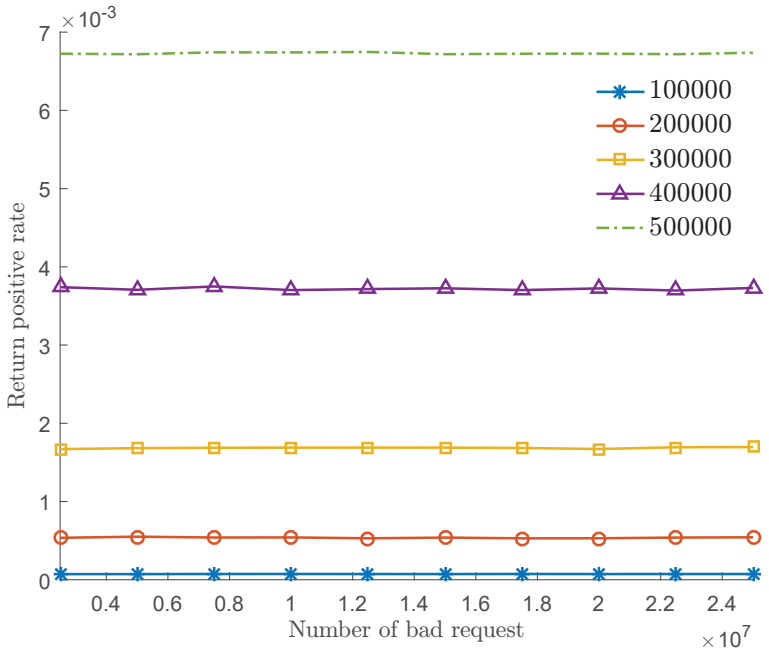


Figure 3.4: Performance of registration/password revision of KSSBF.

for KSSBF, others for GSBF of different f), we increase the number of bad request and show their performance in Fig. 4.4. From this picture, we can see the number of requests arrive at the back end increase linearly, with different slopes according to different f, even if the maximum possible workload is exceeded. This verifies scalability of the defence mechanism.

3.5 Chapter Summary

In this chapter, we design a firewall architecture using a Bloom filter as a pre-screener to protect password-based authentication servers from DDoS attacks. To differentiate whether the firewall is trusted (same-party) or untrusted (third-party), we design KSSBF and GSBF, both of which provide the two-folded provable security against DDoS attacks and off-line brute-force attacks.

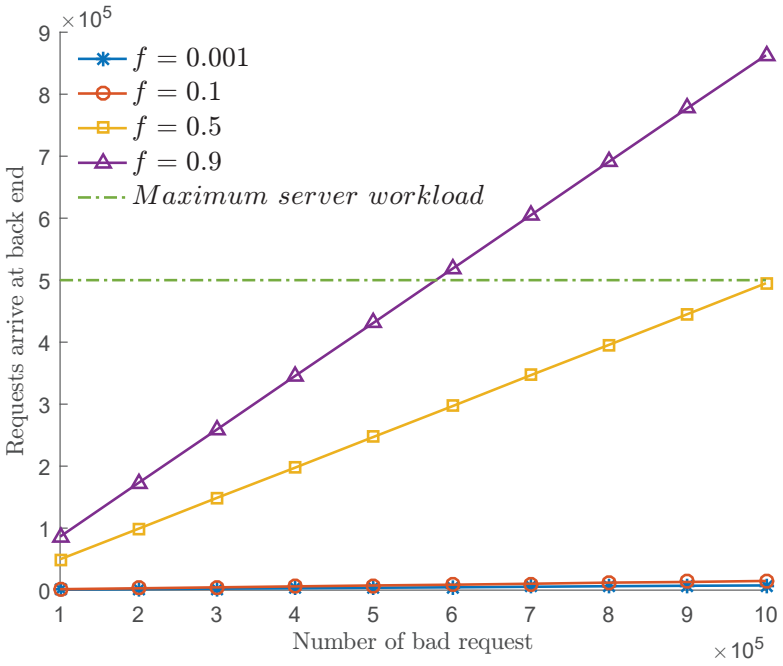


Figure 3.5: Scalability of KSSBF and GSBF pre-screening mechanism.

Chapter 4

Secure Indexing

Thanks to the booming cryptographic techniques [29, 34] in recent years, sensitive data can be outsourced to an untrusted third party (e.g., a cloud server) in a privacy-preserving way. However, robust security guarantees pose a challenge to the efficient retrieval of data, given the impracticality of decrypting all ciphertexts for a search operation. To overcome this obstacle, searchable encryption has emerged as a solution, enabling keyword search over encrypted data. However, the expanding volume of data, especially in large-scale networks [94], necessitates cost-effective storage and processing solutions. Probabilistic data structures for the approximate processing of searchable encryptions have gained popularity in this context, most notably the secure index [36]. It allows users to generate a "trapdoor" from a secret key for a keyword to test its presence in a Bloom filter (BF) [11]. Traditional approaches cater primarily to univariate data processing, while a vast range of AI and data mining applications require bi-attribute data handling, such as key-value pairs or spatiotemporal data. For instance, a log file in operating systems and web servers frequently record a timestamp alongside the log event, rendering each entry as a bi-attribute record (IP, timestamp). The repetition of the same IP necessitates numerous queries over the timestamp, creating an efficiency bottleneck.

Table 4.1: A web server log file.

Ground truth

Time	IP trapdoors
19:22:00	3f3c17b5cfb73e22
19:22:15	ced40518c3e32660
...	
19:23:19	e6c5d3422583e8e5
...	
19:25:55	e6c5d3422583e8e5

With false positives

Time	IP trapdoors
19:22:00	3f3c17b5cfb73e22
19:22:15	ced40518c3e32660
...	
19:23:19	e6c5d3422583e8e5
19:23:24	ced40518c3e32660
...	
19:25:55	e6c5d3422583e8e5
19:25:58	ced40518c3e32660

Table 4.2: Auxiliary knowledge from compressed session data.

Time slot	IP
19:21:55-19:22:05	158.132.150.83, 141.211.29.122
19:22:15-19:22:25	158.132.50.12, 70.108.10.124
19:22:40-19:22:50	124.197.24.255
19:23:15-19:23:25	158.132.150.83, 167.136.142.43
19:25:50-19:26:00	59.160.0.12, 167.136.142.43

Fortunately, in practice not all attributes are sensitive. Non-sensitive attributes in plaintext enable batch operations, such as prefix queries, within a single trapdoor generation. For instance, in a web server log file (Table 4.1), the IP address is sensitive, while the timestamp is not. Notice that the trapdoor 'e6c5d3422583e8e5' repeats twice and is likely to surface elsewhere in the log file. For a range query over the timestamp, for instance, an administrator may want to verify if a specific IP accessed the server between 19:00-20:00, a single trapdoor generation suffices. While this approach alleviates the cost of trapdoor generation, it is vulnerable to inference attacks where an adversary could exploit the plaintext outcome to deduce the sensitive at-

tribute using **auxiliary knowledge**. Adversaries with auxiliary knowledge, such as session data, can compromise privacy by correlating non-sensitive and sensitive attributes, as demonstrated in Table 4.2. For example, by comparing the plaintext of trapdoor 'e6c5d3422583e8e5' with the encrypted log file, it is highly possible that the IP address is '167.136.142.43', which is essentially a known plaintext attack. This demonstrates the outcome of the non-sensitive attribute can lead to privacy loss [50] for the sensitive attribute when the two are **correlated**. While the BF structure offers minimal privacy protection due to its probabilistic nature and tendency for false positives [8] (for instance, a false positive in red in Table 4.1 contradicts the auxiliary knowledge), its protection level is not equivalent to a cryptographic guarantee for every item.

Secure hybrid cloud solutions [62, 97] that safely outsource non-sensitive data have been developed, combining public cloud advantages with strong security. However, these solutions demand potentially unlimited local storage and may induce significant inter-cloud communication overhead [55]. This chapter aims to simultaneously address batch processing over non-sensitive attributes and inter-attribute privacy and create a lightweight, secure bi-attribute index deployable on public cloud platforms. We utilize the randomized response [84] to inject noise into the index, ensuring the outcomes align with local differential privacy [27]. This chapter's contributions are twofold:

Efficient Query Processing: We propose a matrix BF-based index, a 2-dimensional extension of the secure index supports batch operations over one attribute (e.g., IP) and customizable queries over two attributes (e.g., IP and timestamp). We illustrate this framework with a case study, enabling searchable encryptions and range queries over non-sensitive attributes within a single trapdoor generation. Moreover, we introduce two variants of the matrix BF, optimizing insertion and membership test efficiency for different workload patterns.

Bounded Privacy Loss: Given the correlation between the two attributes, achiev-

ing full utility of the non-sensitive attribute and zero privacy loss of the sensitive one concurrently is unattainable. We formally define the privacy loss bounded by a security parameter, ensured by adding noise through the Randomized Response technique [84]. We propose an initialization approach for the index, ensuring a controlled level of privacy loss in data processing, and theoretically derive the false positive rate of the matrix BF index and its maximal capability.

The rest of the chapter is structured as follows. Chapter 4.1 formalizes the problem, discusses baseline solutions, and delineates the threat model. Chapter 4.2 details our proposed approach, including a theoretical analysis and a case study. Chapter 4.3 introduces two matrix BF variants. Chapter 4.4 presents experimental setup and results. Chapter 4.5 concludes this chapter.

4.1 System Overview

4.1.1 Problem Definition

Generally, a bi-attribute index serves as an efficient strategy for executing queries over two attributes of a single data piece in a partitioned manner. This is refined into a membership test on the **co-existence** of both attributes.

Definition 3. ***Bi-Attribute Membership Test.*** *Given a set of bi-attribute data, where (x, y) symbolizes a piece of data from that universe, a bi-attribute membership test over (x, y) is a bivariate function $Q(x, y) = 1$ or 0. This returns true or false for the co-existence of (x, y) in the given set.*

This type of membership test must support prefix queries over one attribute. Consequently, when the suffix attribute is readily visible in plaintext, an efficient batch operation becomes feasible, leading to the concept of a bi-attribute batch membership test.

Definition 4. ***Bi-Attribute Batch Membership Test.***
Given a constant value x_1 of variable x (or y_1 of y), a bi-attribute membership test $Q(x, y)$ is deemed batchable if there exists a univariate membership test $\mathcal{Q}_y(x_1, y) = 1$ or 0 (or $\mathcal{Q}_x(x, y_1)$) for y, returning true or false for the co-existence of (x, y).

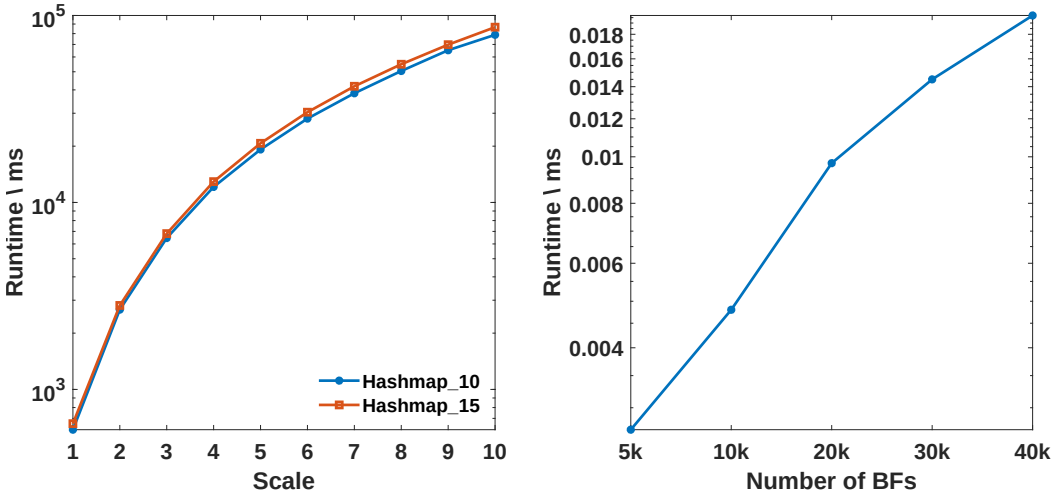
4.1.2 Baseline Solutions

This section discusses existing solutions and their efficacy for bi-attribute batch membership tests.

Single BF Naive Approach. One method uses a secure index to process bi-attribute data. However, it fails to support bi-attribute batch membership tests as it requires unique trapdoor generations for each query.

Hashmaps Utilization. Hashmaps, storing key-value pairs, can be space-consuming. Collision handling involves chaining, making large-scale searches costly. We demonstrate this in Fig. 4.1 (a), where we initialize hashmap lengths to 10 and 15 times the number of elements and reduce them to 1 and 1.5 times respectively at scale=10. Despite O(1) complexity for key search, it doesn't apply to values.

Use of Multiple BFs. Another approach employs multiple BFs, like the multi-dimensional BF (MDBF) for l-dimensional elements [38], or the combinational BF for multiple group membership queries [39]. However, these require many hash functions and memory accesses, and suffer high misclassification probabilities. Furthermore, multiple BFs function as separate entities, failing to establish member relationships within the set. This necessitates knowing where an attribute is stored, leading to a single attribute linear search, and then member searching from the corresponding BF. Bi-attribute batch membership tests require this to be an O(1) complexity process, regardless of the number of BFs used, as validated in Fig. 4.1 (b). Here, BF search runtime increases with the scale of BF sets, though each lookup in the corresponding BF has O(1) time complexity.



(a) Hashmap. (b) Multiple BFs.

Figure 4.1: The runtime of hashmap and multiple BFs with scale increasing.

4.1.3 Threat Model

In our setup, the sensitive attribute is encrypted, and the non-sensitive attribute is visible in plaintext. We assume an honest-but-curious adversary with extensive auxiliary knowledge (e.g., one in Table 4.2) of the inter-attribute correlation, who will attempt to infer the sensitive attribute by eavesdropping on the outcome of the non-sensitive one. An ideal but unattainable method would offer full utility of the non-sensitive attribute with zero privacy loss of the sensitive attribute simultaneously [50]. However, to make it practical, we only require that any published set of outcomes of the non-sensitive attribute results in a bounded privacy loss of the sensitive one:

Definition 5. ***Security Definition.*** *A bi-attribute batch membership test $\mathcal{Q}y(x_1, y)$ (over the non-sensitive attribute y) has a* ***bounded privacy loss*** *if and only if there exists a parameter λ that can converge to 0, such that the following formula holds for any trapdoor T_j and any pair of plaintexts P_{i1}, P_{i2} of attribute x:*

$$\begin{aligned} &|Pr(T_j \leftarrow Trap(P_{i1})|Ak, \mathbb{Y}, y_1) - \\ &Pr(T_j \leftarrow Trap(P_{i2})|Ak, \mathbb{Y})| \leq \lambda \end{aligned} \tag{4.1}$$

where Ak represents the auxiliary knowledge of inter-attribute correlations, $\mathbb{Y}$ is any

set of known outputs of y that the adversary is aware of, y_1 is any potential output of $\mathcal{Q}_y(x_1, y)$ over y, and $Pr(T_j \leftarrow Trap(P_i))$ is the probability that the adversary could ascertain that a trapdoor T_j is generated from plaintext P_i.

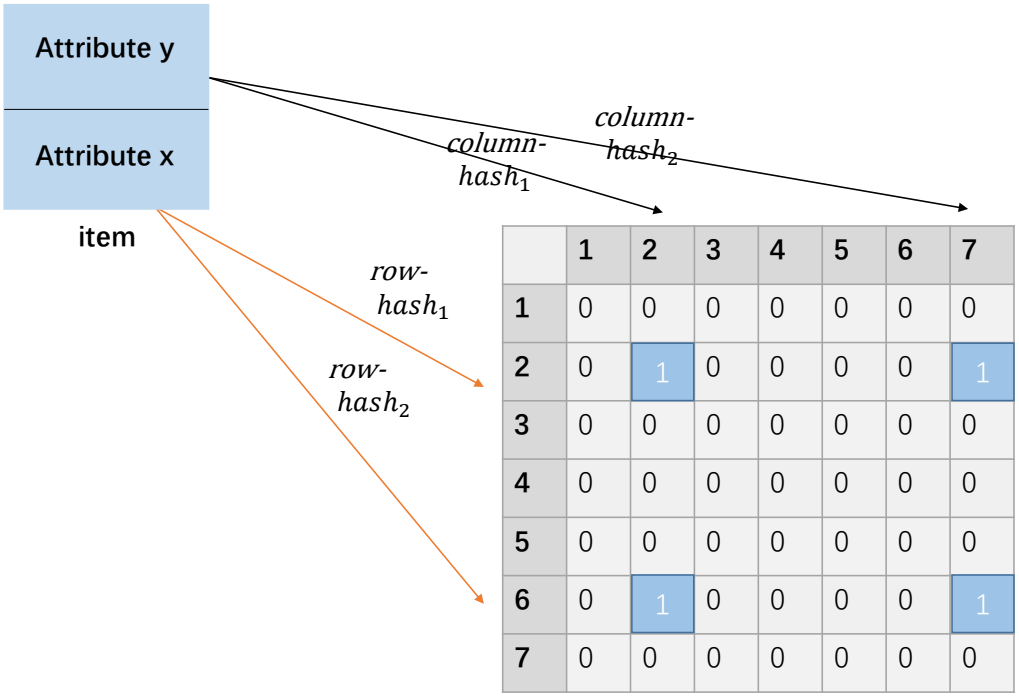
The parameter λ is only a logical concept employed to regulate the privacy loss within the security model. In the following section, we will elucidate how the LDP technique [27] endeavors to assure a bounded privacy loss, managed by the parameter ϵ. Notably, ϵ parallels the role of λ as it characterizes the capability to protect privacy.

4.2 The Matrix BF Index

The Bloom Filter (BF), a space-efficient probabilistic data structure, is an array of m bits using k hash functions for item insertion and querying [11]. It may produce false positives, indicating an item's presence incorrectly. The Matrix BF extends this concept to two dimensions, treating rows and columns as separate BFs with parameters (m_1, n_1, k_1) and (m_2, n_2, k_2) respectively, enabling advanced querying [83]. This work utilizes the Matrix BF for bi-attribute membership tests.

4.2.1 The Basic Structure

The bi-attribute membership test in a matrix BF is described in Algorithm 6. To insert a piece of data (x, y) into the matrix BF, each attribute is hashed separately. The first attribute, x, is hashed using k_1 different hash functions, each generating a row index. Similarly, the second attribute, y, is hashed using k_2 different hash functions, each yielding a column index. The intersection of these hashed rows and columns identifies the positions in the matrix that are set to 1. When querying whether a piece of bi-attribute data (x, y) exists in the set, we follow a similar process. We hash x and y to obtain k_1 and k_2 indices, respectively. We then check the corresponding



	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	1
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	1	0	0	0	0	1
7	0	0	0	0	0	0	0

Figure 4.2: The basic structure of a matrix BF index.

$k_1 \times k_2$ positions in the matrix. If all of these positions are set to 1, we infer that the queried data (x, y) is likely in the set and return "true". If any of these positions is not set to 1, we conclude that the queried data (x, y) is definitely not in the set and return "false". In this way, the matrix BF performs membership tests on two attributes in a partitioned manner. Different sets of hash functions are applied to each attribute, allowing the structure to maintain the co-existence of the two attributes within each data pair.

4.2.2 False Positive Rate

A false positive occurs when the matrix BF indicates that a queried item is part of the set when it is, in fact, not. We can calculate the probability of this happening, denoted as f:

Theorem 4. *The probability of a false positive in a matrix BF is given by* $(1 -$

Algorithm 6 Bi-attribute membership test in a matrix BF

Input: Item (x, y), matrix BF M

1: *ItemInsertion*
2: **for** $i = 1$ **to** k_1 **do**
3: $row\_array[i] \leftarrow rowhash_i(x)$;
4: **for** $j = 1$ **to** k_2 **do**
5: $column\_array[j] \leftarrow columnhash_j(y)$;
6: $M(row\_array[i], column\_array[j]) \leftarrow 1$;
7: **end for**
8: **end for**
9: *ItemLookup*
10: **for** $i = 1$ **to** k_1 **do**
11: $row\_array[i] \leftarrow rowhash_i(x)$;
12: **for** $j = 1$ **to** k_2 **do**
13: $column\_array[j] \leftarrow columnhash_j(y)$;
14: **if** $M(row\_array[i], column\_array[j]) == 0$ **then return** false
15: **end if**
16: **end for**
17: **end for**
18: **return** true

$e^{\frac{-nk_1k_2}{m_1m_2}})^{k_1k_2}$.

Proof. Suppose the hashes choose the positions equiprobably. Given an item that is **not** in the membership set,[1] for any bit the probability that it is not set to 1 in a single hash insertion turns out to be:

$$p_1 = 1 - \frac{1}{m_1m_2} \tag{4.2}$$

After all the k_1k_2 executions the probability that the bit is not set to 1 is $p_2 = (1 - \frac{1}{m_1m_2})^{k_1k_2}$. When n items are inserted, the probability that the bit is still 0 is $p_3 = (1 - \frac{1}{m_1m_2})^{nk_1k_2}$. Hence, the probability that the bit is set to 1 is $p_4 = 1 - (1 - \frac{1}{m_1m_2})^{nk_1k_2}$. The false positive result occurs when all the queried k_1k_2 bits are set to 1:

$$f = (1 - (1 - \frac{1}{m_1m_2})^{nk_1k_2})^{k_1k_2} \approx (1 - e^{\frac{-nk_1k_2}{m_1m_2}})^{k_1k_2}$$

□

Next, let us decide the optimal number of hash functions and the lowest false positive rate of a matrix BF:

Theorem 5. *The optimal condition of a matrix BF can be gained from substituting $k = k_1k_2$ and $m = m_1m_2$ into the formula $k = \frac{m}{n}ln2$, which is the theoretical optimal condition of a standard BF.*

Proof. Recall that

$$f = e^{k_1k_2ln(1-e^{\frac{-nk_1k_2}{m_1m_2}})}$$

[1] If we randomly choose an element from the entire space, and when the domain is large enough, the probability that it falls into a predefined membership set tends to be zero.

Let $p = e^{\frac{-nk_1k_2}{m_1m_2}}$, $g = k_1k_2ln(1-e^{\frac{-nk_1k_2}{m_1m_2}}) = \frac{-m_1m_2}{n}lnpln(1-p)$. Clearly, f arrives at its minimum when g reaches its minimum. Due to the symmetry of lnp and $ln(1-p)$, the following restriction holds:

$$p = e^{\frac{-nk_1k_2}{m_1m_2}} = \frac{1}{2} \tag{4.3}$$

It implies f arrives at its minimum when overall 50% bits are occupied with 1. Hence, we have the boundary condition:

$$k_1k_2 = \frac{m_1m_2}{n}ln2$$

As well as the minimal value of f:

$$p_{min} = (\frac{1}{2})^{k_1k_2} \tag{4.4}$$

Compare with the corresponding results of the standard BF:

$$p = \frac{1}{2}, k = \frac{m}{n}ln2, p_{min} = (\frac{1}{2})^k \tag{4.5}$$

Replace k and m in equation 4.5 by k_1k_2 and m_1m_2, then complete the proof. □

4.2.3 Partitioned Hashing Strategies

The structure of the matrix BF allows us to apply different hashing functions over different attributes. Specifically, we can use general hash functions for the trapdoor insertion/query for normal requests, while employing customized hash functions on the non-sensitive attribute to meet specific requirements. We illustrate this by using our index approach to partition the two attributes and then applying existing

Algorithm 7 Range searching with LSH

Input: Timestamp t_s, matrix BF index M, global BF B;

1: $FindBlock : (t_s, M, B)$
2: **for** $j = 1$; $j < k_2$; $j++$ **do**
3: $column\_array[j] \leftarrow columnhash_j(y)$
4: **for** $t = 1$; t in TimeBlock detect set; $t++$ **do**
5: **for** $i = 1$; $i < k_1$; $i++$ **do**
6: $row\_array[i] \leftarrow rowhash_i(t)$
7: **if** $M(row\_array[i], column\_array[j]) = 0$ **then** break
8: **end if**
9: **end for**
10: Goto $GlobalBF : (t, y, B)$
11: **end for**
12: **end for**
13: $GlobalBF : (t, y, B)$
14: **for** $s = 1$; $s < k_3$; $s++$ **do**
15: $location\_[j] \leftarrow columnhash_s(t + y)$
16: **if** $B(location\_[j]) == 1$ **then return** false
17: **end if**
18: **end for**
19: **return** true

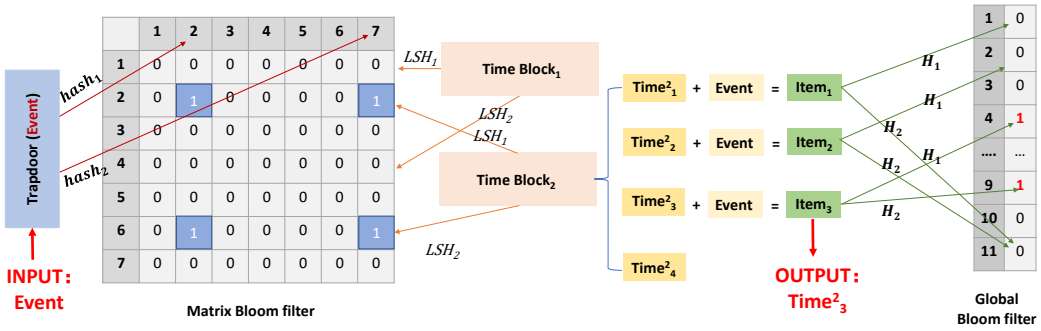


Figure 4.3: The framework of our study case. Timestamps are queried with LSH in the matrix BF first and are further checked in a global BF.

efficient hashing technologies (e.g., LSH [41]) to improve performance. We demonstrate this with a real-world application by answering the timestamped membership query (tmt-query) problem posed by Peng et al. [68]: "Given an element in the form of (event, timestamp), can we determine whether any event occurred within a time range (t_1, t_2)?" In our setup, the index supports searchable encryption over the sensitive "event" attribute and range searching over the non-sensitive "timestamp" attribute. Thus, trapdoors are generated only for events, while timestamps remain in plaintext.

Even though we now need only one trapdoor generation for multiple timestamps, traversing the entire history would still require a brute-force linear search. This is still expensive as the scale increases. However, we can further improve this by using LSH, as introduced in the preliminaries section, for searching over the timestamps. Figure 4.3 illustrates the framework of the case study. Our basic idea is to use a matrix BF index as a preprocessing step to reduce the search range over timestamps by utilizing the locality provided by LSH. Specifically, we set some probing points uniformly on the time domain, corresponding to a series of time blocks that completely cover the entire domain. The size of the time blocks is related to the locality of LSH. Therefore, any item membership test (on each probing point) is essentially a neighbourhood search to identify the time block that the timestamp should belong to. For example, in Figure

4.3, the query on time block 1 will return false, which means the timestamp does not fall into this time block. We then move to the next probing point, corresponding to time block 2, which returns true and should contain the objective timestamp. Finally, to improve the accuracy, we use a global BF that records the element (event, timestamp) as a whole to pinpoint the objective timestamp within the time block. The entire process is described in Algorithm 7.

4.2.4 Securing the index

4.2.4.1 Index Initialization

In light of Definition 4.1, the security requirement is that the release of any set of outcomes related to the non-sensitive attribute should contribute only a bounded privacy loss of the sensitive one. This essentially relates to the distributional similarity of items over the non-sensitive attribute. One extreme case would be if every pair of sensitive items had the same distribution of co-occurring non-sensitive values, meaning that the outcome of any non-sensitive value set contains no information about the sensitive item, even though they may be correlated.

In the context of the matrix BF, for any specified sensitive item, the set of co-existing non-sensitive items has a BF representation, which includes all the matrix BF columns that the sensitive item is hashed to. If the universe of all possible BF representations can be ensured by equation 2.1, satisfying the definition of LDP, this implies that any possible membership test subset also maintains the same level of privacy. Consequently, we can assert the following:

Corollary 1. *If the set of BF representations $\mathbb{B}_1, \mathbb{B}_2, ..., \mathbb{B}_i$ for all non-sensitive items complies with LDP, then any bi-attribute batch membership test over the non-sensitive attribute will have a privacy loss that is bounded (controlled by ϵ).*

As p approaches 0.5, it implies that ϵ approaches 0, leading to an infinite perturbation

that makes any inputs indistinguishable. In this scenario, the BF representation returns "yes" with a probability of 50% for any item and hence contains no information about the set of inserted values (and thereby the sensitive attribute).

The initialization of a matrix BF index is described in Algorithm 8. Each item from the membership set I into the matrix M (line 2), and the algorithm starts two nested loops. The outer loop iterates over all the indices i where the sum of the ith row in the array is not zero. The inner loop iterates over all the indices j (lines 3-4). For each pair of indices (i, j), the algorithm stores the value of the cell at the ith row and jth column of the matrix M in a temporary variable $Temp$ (line 5). Finally, the algorithm generates a random number between 0 and 1. If this number is less than the perturbation parameter p, the value of the cell at the ith row and jth column of the matrix M remains the same. Otherwise, the value of this cell is switched to its complement, i.e., 1 becomes 0 and vice versa (line 6-9).

The user determines the privacy budget ϵ to choose (which corresponds to the perturbation parameter p), and the initialized matrix BF will have the lowest false positive rate f_1 as the **privacy bound**. The user also determines the highest acceptable false positive rate f_2 as the **utility bound**. Thus, the index starts at a specific state and is filled up as the user inserts new items. As the BF becomes full (or full enough to start producing many false positives), the false positive rate increases and the utility of the BF tends to worsen. Therefore, the BF is expected to operate effectively between f_1 and f_2. The user should reconstruct a new index when the false positive rate exceeds f_2.

Index Update. Conventional BFs do not support deletions, with Counting Bloom Filters [31] as an exception. However, they can still lead to false negatives post deletion [37]. In our framework, we treat deletions as if they were part of the initial perturbation, inserting fake items. So, we don't need to delete an item from the index. Revisions, which involve deleting an old item and inserting a new one, are simplified to just an item insertion.

Algorithm 8 Index initialization with RR

Input: Membership item set I, Perturbation parameter p, matrix BF M;

1: $Insert(item \in I) \rightarrow M$
2: **for** All i when $\sum row\_array[i]! = 0$ **do**
3: **for** All j **do**
4: $Temp = M(row\_array[i], column\_array[j])$
5: **if** $rand(1) < p$ **then**
6: $M(row\_array[i], column\_array[j]) = Temp$
7: **else**
8: $M(row\_array[i], column\_array[j]) = 1 - Temp$
9: **end if**
10: **end for**
11: **end for**

Capability. We can now derive the capacity of an initialized index, i.e., the number of items that can be added. Once the values of m and k for the Bloom filter are fixed, the following formula relating n (the number of items in the filter) and f (the false positive rate) holds:

$$n = -\frac{m}{k} \ln(1 - e^{\frac{\ln f}{k}})$$

We can then determine the number of items that can be added upon initialization as follows:

$$N = n_2 - n_1 = -\frac{m}{k} \ln \frac{1 - e^{\frac{\ln f_2}{k}}}{1 - e^{\frac{\ln f_1}{k}}}$$

where n_2 and n_1 are the numbers of items in the BF when the false positive rates are f_2 and f_1.

4.3 Handling Inter-attribute Correlations

We have discussed the matrix BF assuming independent attributes, which contradicts our problem scenario involving inter-attribute correlations. Practically, the matrix's size and shape should correspond to pre-known dataset characteristics. This section proposes two matrix BF variants leveraging prior knowledge. The maximum adaptive approach is universally applicable, while minimum storage approach utilizes precise repetition information of each key, requiring prior knowledge. Most real-world situations lie somewhere between these extremes, which will be illustrated later.

4.3.1 Maximum Adaptive Matrix

The number of items on the two attributes and their combination patterns co-determine the shape and size of the matrix BF. We should consider the worst case when having no prior knowledge about the dataset, or the index should be frequently renewed and updated by adding unknown items. That is, for any given n_1, n_2, we pre-allocate large enough spaces for any given $\mathbb{S}$ with parameters n_1, n_2, where the maximum possible combinations is $n_1 n_2$. To be adaptive to the most general case, given m_1 and m_2, treat the rows and columns as dedicated standard BFs, both of which hold the lowest possible false positive rate. Hence, we have

$$k_1 = \frac{m_1}{n_1} ln2, k_2 = \frac{m_2}{n_2} ln2$$

Load factor. The maximum adaptive matrix is relatively empty. About 50% of rows/columns is set to 0, hence the load factor is approximately 25%. In the worst case where $n = n_1 n_2$, since the queries in row and columns satisfy equation 4.3, the load factor turns out to be approximately $50\% \times 50\% = 25\%$. In general cases, n is in fact less than $n_1 n_2$, thus the load factor is always less than 25%.

False positive rate. When queries in both rows and columns turn out to be false

positive, the overall false positive result occurs. Hence,

$$fpr_{mam} = f_1 \times f_2 = (\frac{1}{2})^{k_1+k_2}$$

Storage overhead. As mentioned, the maximum adaptive matrix is relatively empty. It achieves a better performance of adaptive batch queries at the expense of storage overhead. Let us derive the maximum possible storage cost of the maximum adaptive matrix, and compare it with a standard BF.

Assume a standard BF is allocated to insert the same set of $n_1 n_2$ elements, where the same false positive rate $fpr_{mam} = (\frac{1}{2})^{k_1+k_2}$ is achieved. Let the size of the standard BF be m_0, hence, the corresponding number of hash functions in the standard BF turns out to be $k_0 = \frac{m_0}{n_1 n_2} ln2$. Let $k_0 = k_1 + k_2$, we have:

$$(\frac{m_1}{n_1} + \frac{m_2}{n_2})ln2 = (\frac{n_2 m_1 + n_1 m_2}{n_1 n_2})ln2 = (\frac{m_0}{n_1 n_2})ln2$$

That is,

$$n_1 m_2 + n_2 m_1 = m_0$$

Due to the mean value inequality, we have:

$$2\sqrt{n_1 n_2} \cdot \sqrt{m_1 m_2} \leq m_0$$

Hence,

$$m = m_1 m_2 \leq \frac{(m_0)^2}{4 n_1 n_2} \tag{4.6}$$

Complexity. For each membership test, there are overall $k_1 k_2$ hashing/comparisons, therefore the complexity is simply $O(k_1 k_2)$. Since

$$k_1 k_2 = (\frac{m_1 m_2}{n_1 n_2})ln^2 2$$

Substitute by equation 4.6, we have:

$$k_1 k_2 = (\frac{m_1 m_2}{n_1 n_2})ln^2 2 \leq \frac{(m_0)^2}{4(n_1 n_2)^2} ln^2 2 = \frac{k_0^2}{4}$$

However, in the case of batch membership tests, the complexity can be further reduced. Consider the best case of key-value pairs, where a series of varying values correspond to the same key. For this special dataset, the key is to be hashed only once, while the values are to be hashed at most $max(k_1, k_2)$ times. Therefore, the best complexity can be lowered to $O(max(k_1, k_2))$.

4.3.2 Minimum Storage Matrix

While the maximum adaptive matrix is adaptive to the most general case without prior knowledge, it bears an extensive overhead on the storage cost and hashing numbers. However, in practice, we may be aware of some statistical features of the dataset, with which we can construct a lower storage matrix applying in read-only (or not frequently updated) scenarios, e.g., a dictionary. In this part, we will discuss some typical datasets with strong background knowledge and construct the corresponding matrices according to our theory.

Case A. Let us start at a special case where there exists a bijection between $\mathbb{S}_1$ and $\mathbb{S}_2$, i.e., for any two different elements, there is no repeat on both of the two components. This is typical for a time-series data, e.g., a non-overlapped trajectory of the movement of a point. Clearly, in this case $n_1 = n_2 = n$, which means the two sets of different components contain the same number of elements. In this special case, the answer is simple. In this case, positions of (k_1, m_1) and (k_2, m_2) are symmetric, since both (k_1, k_2) and (m_1, m_2) are commutable. Naturally, we employ a square matrix for insertion of the n elements, where $k_1 = k_2 = k, m_1 = m_2 = m$.

Case B. In this case, let us consider a more general situation where elements in $\mathbb{S}$ can be represented as a weak combination from $\mathbb{S}_1$ and $\mathbb{S}_2$. For the convenience of discussion, let $n_1 > n_2$. In this case, each element in $\mathbb{S}_2$ combines with elements in $\mathbb{S}_1$ at most j times. Elements in $\mathbb{S}_1$ are not allowed to repeat. Hence, $n_1 = jn_2$. This is typical for any lexicographical ordered data set where $j = 26$.

Element insertion/query. Let us start at a special case where $n_1 = 2n_2$. We adopt a special hash to classify components in $\mathbb{S}_1$ into 2 types, each belonging to either $\mathbb{S}_{11}$ or $\mathbb{S}_{12}$. In each set, there are $n_2 = \frac{n_1}{2}$ elements. For each set, we employ a square matrix as described in the context of Case A.

For any element to insert, we first employ the special hash to find out which square matrix it belongs to. Then, in the determined square matrix, find out the corresponding rows for components in $\mathbb{S}_1$ as well as columns for components in $\mathbb{S}_2$.

For the two square matrices, components in $\mathbb{S}_2$ are always mapped into the same columns. Hence, it equals merging the two matrices from left to right to get a $2m_2 \times m_2$ matrix. For the general case, when $n_1 = jn_2$, similarly, the special hash should choose which of the j sets an element belongs to. Then for each set, a square matrix is adopted, and the j square matrices are stuck together to build a $jm \times m$ j-matrix. See Fig. 4.4 as an example. When any elements are queried, similar rules are executed to find out if the k_1k_2 bits are 1.

False positive rate. Let us discuss the special case where $n_1 = 2n_2$ first. Suppose overall $2m^2$ bits are employed to build this matrix. Now let us prove the false positive rate in this scenario equals using a standard BF, where the same number of bits ($2m^2$) is allocated to insert the n_1 elements.

Obviously, the false positive rate of that standard BF is $(\frac{1}{2})^{\frac{2m^2}{n_1}ln2}$. In the matrix, since the two square matrices share the same false positive rate, only one of them needs to be calculated. The false positive rate turns out to be:

$$fpr_{2-matrix} = (\frac{1}{2})^{\frac{m^2}{n_2}ln2} = (\frac{1}{2})^{\frac{2m^2}{n_1}ln2}$$

Extending to the j-matrix case, we have the following theorem:

Theorem 6. *When jm^2 bits are adopted to build a general $jm \times m$ j-matrix case, the false positive rate equals that of a standard BF where the same number of bits (jm^2) is used for insertion of the same number of elements. (Say, $n_1 = jn_2$)*

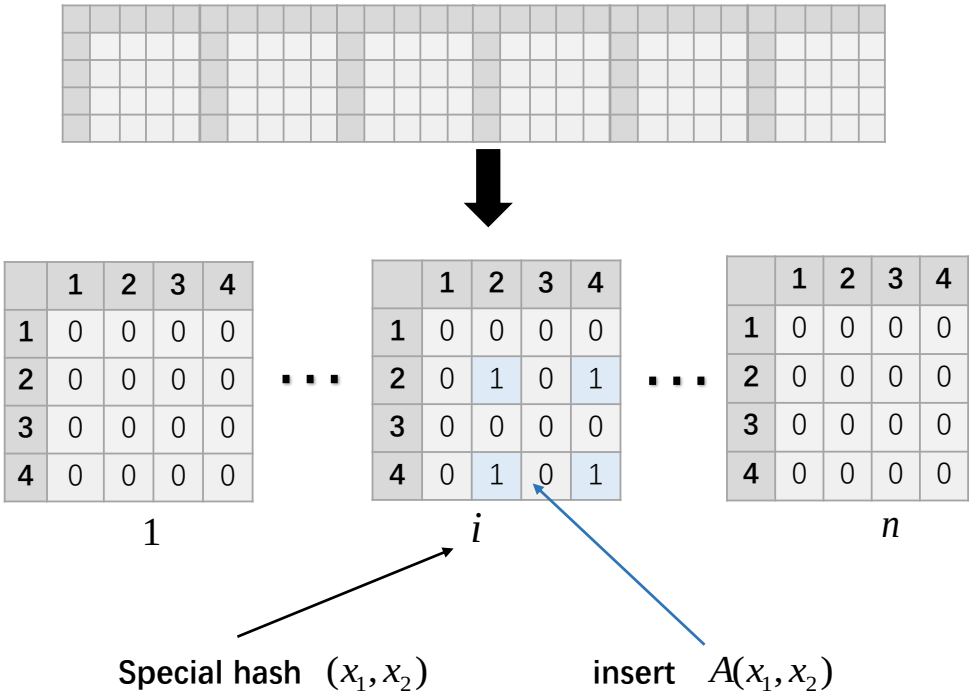


Figure 4.4: Element insertion/query in a j-matrix.

Proof. The false positive rate of that standard BF is $(\frac{1}{2})^{\frac{jm^2}{n_1}ln2}$. Since the j square matrices share the same false positive rate, just one of them needs to be considered. The false positive rate turns out to be:

$$fpr_{j-matrix} = (\frac{1}{2})^{\frac{m^2}{n_2}ln2} = (\frac{1}{2})^{\frac{jm^2}{n_1}ln2}$$

□

4.4 Experiments

The experiments are conducted using MATLAB R2021a on a PC equipped with an Intel i7-10700K RTX 3090 eight-core processor, 128GB RAM, and Windows 10 OS. To implement the BFs, we opt for universal hash functions [13] to map elements into BFs. For any item X represented as $X =< x_1, x_2, ..., x_b >$ in b-bits, the ith hash

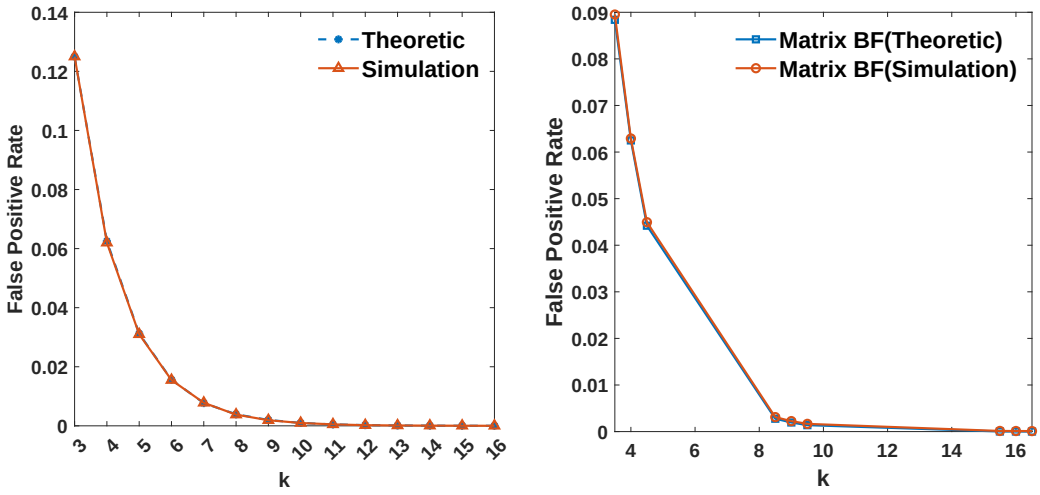
function over X $h_i(x)$ is calculated as $h_i(x) = (d_{i1} \times x_1) \oplus (d_{i2} \times x_2) \oplus ... \oplus (d_{i3} \times x_3)$, where $\times$ signifies the bitwise AND operator and $\oplus$ represents the bitwise XOR operator.

4.4.1 False Positive Rate

We use two real-world datasets from the Bag of Words database, which consists of five text collections. We chose **KOS** (with 353160 key-value pairs from 3430 keys and 5851 values) and **NIPS** (with 746316 key-value pairs from 1500 keys and 12375 values). Moreover, we create a synthetic bi-attribute dataset, **SYN_FD**, which is fully-duplicated. This dataset is formed from the Euclidean cross product of two scalar datasets where elements are random numbers, which gives the bi-attribute data a duplication feature on both attributes. Both scalar datasets are fixed with 1000 numbers, resulting in a total of 1000000 bi-attribute data in the dataset.

Verification of Theorem 4. We use **SYN_FD** for this part. For the standard BF, we insert a piece of bi-attribute data as a whole. Let k be different integers, and allocate an appropriate value of m using the formula $k = \frac{m}{n} ln2$. For the matrix BF, we allocate the same number of bits and items where $k = k_1 k_2$, as a comparison to the standard BF. The insertion rules of a single row/column for the matrix BF are the same as for a standard BF.

The left-hand part of Fig.4.5 shows the tendency of false positive rates of the standard BF varying with respect to k, and the right-hand part is that of the matrix BF. Notably, the points in the matrix BF are more concentrated because m_1 and m_2 are commutable, which results in a square matrix BF. However, since $k = k_1 k_2$, if both k_1 and k_2 are required to be strict integers, there are fewer choices for k. Therefore, we approximate some points near $k = 8$ and $k = 16$ where the number of hash functions is fixed to integers, while the corresponding m_1 and m_2 are calculated from the non-integer values of k. As seen from Fig.4.5, the experimental results align well with the



(a) Standard BF. (b) Matrix BF.

Figure 4.5: Comparison of theoretical/experimental value of the optimal false positive rate in a standard/matrix BF.

theoretical values. Thus, Theorem 4 is verified, which implies the performance of our matrix BF is equivalent to the standard BF.

On maximum adaptive matrix. The performance of the maximum adaptive matrix is evaluated under various parameter settings on the aforementioned three datasets. We allow $k = k_1 + k_2$ to take on different integer values. The parameters m_i are adjusted to appropriate values using the formula $k_i = \frac{m_i}{n_i} ln2, i = 1, 2$, respectively. The results are illustrated in Figs. 4.6 (a) (c) (e), which indicate that the evaluated false positive rate (fpr) aligns with the theoretical value when the matrix is fully populated, as evaluated on **SYN_FD**. However, as the two real-world datasets are not as duplicated as the synthetic one, the matrices are less loaded, leading to a lower false positive rate.

Figs. 4.6 (b) (d) (f) assess the false positives of different proportions of $\frac{m_1}{m_2}$ as m_0 varies. Similarly, the evaluated values are well-aligned with the theoretical ones when the matrix is fully populated, i.e., as evaluated on **SYN_FD**, and are considerably lower for real-world datasets. This suggests that the maximum adaptive matrix re-

quires a significant amount of storage for typical cases, a trade-off for the generality it offers.

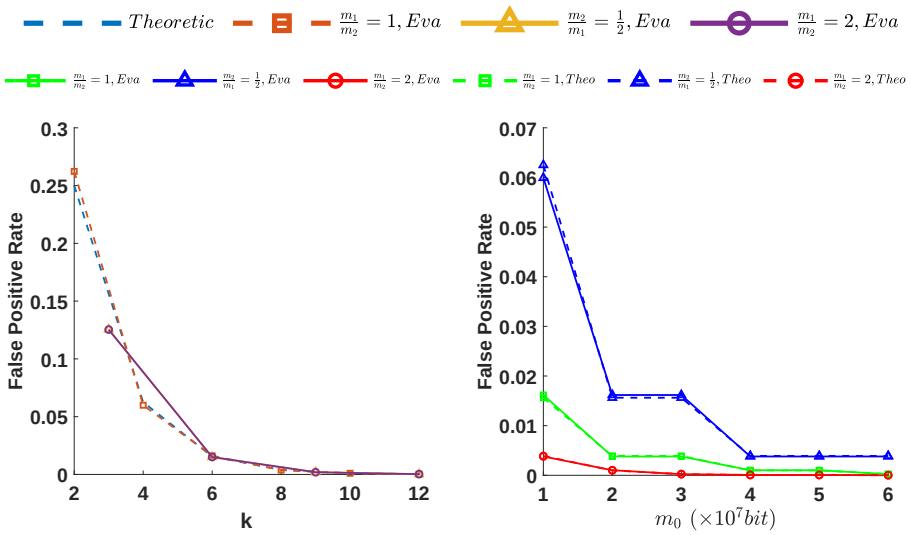
In our final set of experiments, we examine the load factor of our matrix with respect to the proportion of inserted elements. The parameters selected for this are identical to those in previous experiments. Table 4.3 illustrates that the load factors for different values of $\frac{m_1}{m_2}$ are nearly identical when the same proportion of elements is inserted. In particular, when the matrix is fully loaded, the load factor is very close to the predicted 25%.

Table 4.3: Load factor & proportion of elements of different $\frac{m_1}{m_2}$'s

Load factor $\frac{m_1}{m_2}$ / Proportion	$\frac{1}{4}$	$\frac{1}{2}$	1
20%	0.063681451	0.065117817	0.065625635
40%	0.121001445	0.120029788	0.122250358
60%	0.170028163	0.167312289	0.175074846
80%	0.213055668	0.210467359	0.217504935
100%	0.249106352	0.24721041	0.256099335

On minimum storage matrix. This part evaluates the robustness and storage efficiency of the minimum storage matrix. We set $k = \frac{k_1}{j} = k_2$ and $n_1 = jn_2$ with $n_2 = 144$, and allow k to take different integer values. The value of m_2 is adjusted accordingly, based on the formula $k^2 = \frac{m_2^2}{n_2} ln2$.

We plot the false positive rates versus k^2 for $j = 2, 10, 40, 100$ on the left part of Fig. 4.7. To make an approximation, we take some points nearby non-integer k^2 values while fixing the number of hash functions to integers. The corresponding number of bits is then calculated from the non-integer values of k. Our results indicate that all four j-matrices perform similarly and remain almost unchanged regardless of the

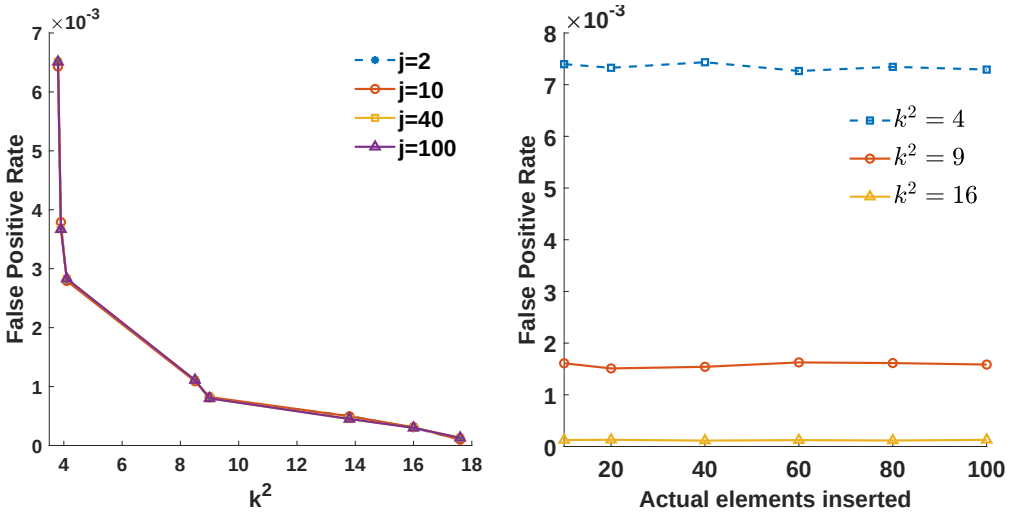


(a) **SYN_FD.** (b) **SYN_FD.**

(c) **NIPS.** (d) **NIPS.**

(e) **KOS.** (f) **KOS.**

Figure 4.6: False positive rate of maximum adaptive matrix with respect to k and m_0.



(a) fpr with respect to k^2.

(b) Robustness.

Figure 4.7: The false positive rate and robustness of minimum storage matrix.

value of j.

Next, we fix k^2 at $4, 9, 16$ and plot the false positive rates with respect to j. As illustrated in the right-hand part of Fig. 4.7, the false positive rate remains stable as j increases, indicating the high robustness of our proposed structure.

Finally, we compare the storage cost for initializing a minimum storage matrix and a maximum adaptive matrix at the same theoretical false positive rate, as shown in Table 4.4. The results demonstrate that when the dataset duplication level is low, the storage can be reduced by several orders of magnitude if we have sufficient prior knowledge about the datasets. This highlights the potential for significant storage efficiency gains with the minimum storage matrix, particularly in read-only or infrequently updated scenarios where dataset characteristics can be accurately predicted.

Table 4.4: Storage Comparison

Method		Key	Value	fpr	k_1	k_2	$m_0(MB)$
Maximum adaptive matrix		10^4	10^6	$\frac{1}{2}^4$	2	2	2.9×10^4
				$\frac{1}{2}^{12}$	6	6	8.9×10^4
Minimum storage matrix	j=50			$\frac{1}{2}^4$	2	2	34.4
	j=50			$\frac{1}{2}^{12}$	4	4	103.19
	j=100			$\frac{1}{2}^4$	2	2	68.79

4.4.2 Batch Performance

In this part, we assess the search time performance of our proposed structure. We utilize the following datasets:

Solar: Extracted from the Solar Power Data for Integration Studies, this dataset consists of 1-year worth of solar power data and hourly day-ahead forecasts for around 6000 simulated PV plants in Alabama. We use a subset of this data for our experiment, treating solar power as events and time as values. The dataset consists of 62 events and 287 values (in the form of hour and minute), resulting in 105120 combinations, many of which are duplicated.

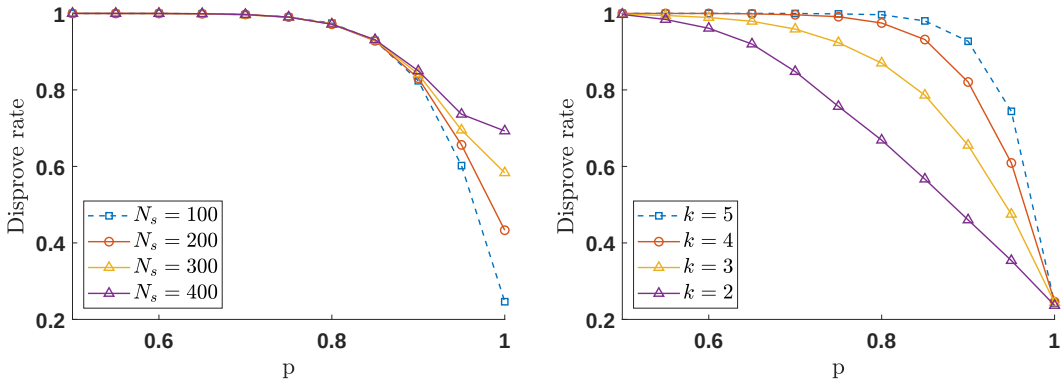
Electricity: This dataset contains electricity consumption data from 321 clients, recorded every 15 minutes from 2012 to 2014. Here, electricity consumption is treated as the event, and the time is considered the value. The dataset has 32 events and 95 values (in the form of hour and minute), leading to 105120 combinations with numerous duplications.

SYN_ND: This is a synthetic, non-duplicated dataset generated from 20 events and 3599 values. Each pair from the two scalar sets is unique, resulting in 71980 non-duplicated pairs. The form of the value is (minute, second).

We evaluate the performance by comparing the total hashing generation times concerning different w values, the locality of the Locality-Sensitive Hashing (LSH) used,

Table 4.5: Times of searching and trapdoor generation.

Data-Set	Comparsion methods	Comparsion times									
		w=1	w=2	w=3	w=4	w=5	w=6	w=7	w=8	w=9	w=10
Solar	Brute force	149.67									
	LSH	149.67	127.82	119.71	111.31	106.27	99.59	92.03	84.50	82.20	75.05
Electricity	Brute force	52.50									
	LSH	52.50	45.32	43.07	41.11	38.71	37.99	35.12	32.99	31.31	27.56
SYN	Brute force	1860.5									
	LSH	1860.5	838.17	615.10	474.73	342.2	299.5	237.85	205.25	190.85	181.57
Trapdoor		0	1	1.5	2	2.5	3	3.5	4	4.5	5



(a) Number of sensitive attributes.

(b) Number of hashes.

Figure 4.8: The disprove rate of auxiliary knowledge.

and the trapdoor generation times. The results are listed in Table 4.5. From these results, it is evident that LSH substantially reduces the hashing time for timestamp searching, with a reasonable cost of trapdoor generation.

The LSH comparison time decreases as w increases for two reasons. First, LSH allows nearby data to be checked only once, thus reducing redundant operations. Second, as the locality w increases, each time block contains more data, reducing the overall time required for comparisons.

Our synthetic data yields better results in terms of reducing search times because it is non-duplicated. Real datasets contain some non-existing and duplicated values, which slightly degrade the performance. However, the results are still significantly better than using a brute force approach. In summary, an LSH-based index can effectively decrease the hashing times during brute force searches over timestamps.

4.4.3 Privacy Guarantee

This part evaluates the privacy guarantees provided by the initialized matrix BF index. We configure the matrix size to 5000×5000 and insert some bi-attribute data, including a piece of target data for which we possess auxiliary knowledge to infer. We

then perturb the matrix BF using RR for all non-zero rows. If more than one data record is discovered within the time range valid for the auxiliary knowledge, we deem the perturbation to have successfully disproves the auxiliary knowledge.

Fig. 4.8 presents the disprove rates. When $p = 1$, no perturbation is applied. N_s in the left-hand part denotes the number of inserted sensitive attributes, with the hash number fixed at 4. The disprove rate for $p = 1$ originates from the inherent false positives of the BF, which is referred to as the "better than nothing" privacy. This rate increases with the number of inserted items. Regardless of the initial state, all rates quickly rise and converge to 1 in relation to $1 - p$, taking advantage of the RR perturbation mechanism.

The right-hand part of the figure keeps the number of items constant while varying the hash numbers, which influences the slopes of the curves. A higher hash number yields more privacy for the same p value. This demonstrates the ability of our approach to provide privacy guarantees, validating its effectiveness in preserving the privacy of sensitive attributes.

4.5 Chapter Summary

This chapter presents a secure bi-attribute index, enabling batch operations over one attribute. Efficiency-wise, the proposed matrix BF structure partitions query processing over two attributes, one of which can be non-sensitive and processed in plaintext. The structure keeps the attribute co-existence and supports any existing efficient hashing technique to further enhance the performance under certain scenarios. Privacy-wise, the proposed initiation approach with RR protects the BF representation of the plaintext outcome and achieves a bound privacy loss.

Chapter 5

Interactive Trimming

In the era of big data and AI, the sheer volume and ubiquity of data have profound impact on our daily life and the world at large. As such, the integrity of data stands as a cornerstone for high-quality data analysis and decision making. Unfortunately, data integrity is under perpetual threat — malicious entities frequently engage in data manipulation, fabricating falsified values to skew outcomes in their favor.

The issue of online data manipulation has been a focal point in the data management community. For example, an e-commerce platform uses a collaborative filtering algorithm to recommend products to users. A malicious vendor aims to artificially boost the visibility of their products. They create multiple fake accounts and manipulate their interaction data (e.g., ratings, purchase history) in real-time to influence the recommender system. These also apply to the field of knowledge graph [4, 91, 100], federated recommendation systems [73,75,85], and countermeasures [88,89,93]. Data manipulation attacks also pose immediate threats to the training process of machine learning systems. Given these high stakes, it is imperative for data collectors to take safeguard measures to detect and neutralize data poisoning attacks, while retaining good quality of the rest (benign) data.

To reduce the impact of data manipulation attacks, one approach is to sanitize the in-

put dataset. A classic method is distance-based sanitization, also known as trimming, where the defender calculates the distance d_i for each data point i and removes any point with $d_i > \theta_d$, a threshold chosen by the defender [46]. Popular distance-based defenses against data manipulation attacks include [47, 49], by optimizing a designated objective function. However, such strategies are static and neglect the evasive nature of adversaries, that is, they always manage to circumvent these defensive measures [64]. Therefore, an evasion-aware defense strategy must consider potential evasion strategies employed by these adversaries. Game theory is a common tool to find a dynamic balance between evasive attackers and defenders, known as Nash equilibrium. Recently, a few game-theoretical models [64, 98] have been proposed for **static** data poisoning attacks, where data are collected in a single round. However, in many real-life data collection systems, data are frequently updated or streaming, and the collection process is continual or in multi-round. As such, a static defensive strategy is insufficient, as adversaries can adapt their strategies in each round. Due to the immense complexity of potential strategies a dynamic attacker might deploy, it has rarely been explored in the field of **online data manipulation attacks**.

In this chapter, our objective is to derive a feasible Stackelberg equilibrium within a **complete trimming strategy space** to defend against data manipulation attacks, specifically in the context of online data poisoning. Our game-theoretic model is anchored in the simplicity of the trimming strategy and is shown to achieve a genuine equilibrium within its complete strategy space. The findings are validated using real-world machine learning data across widely-used algorithms, including k-means, SVM, and SOM classification. We illustrate how the threshold and poison values are determined and elucidate the impact of each scheme on the system's final outcome. Additionally, through empirical studies, we illustrate that **attackers who behave irrationally and diverge from rational strategies will merely gain less utility from poison values**. The key contributions of this chapter can be encapsulated as follows:

· We propose an interactive game-theoretic model for online data poisoning attacks and defenses using the trimming mechanism. This model streamlines the formulation process, accommodates a complete strategy spectrum, and simplifies the derivation of Stackelberg equilibrium, even against evasive and colluding attackers with diversified poisoning strategies.

· We utilize the principle of least action and the Euler-Lagrange equation in theoretical physics to build an analytical model for the game-theoretic process. The model is in the form of the Lagrangian that governs the system in both equilibrium and non-equilibrium states.

· We present a case study in a privacy-preserving data collection system under local differential privacy (LDP) [24, 32, 92] where a non-deterministic utility function is adopted. Two strategies are devised from this analytical model, namely, Tit-for-tat and Elastic, based on which we apply the Euler-Lagrange equation to derive the system's steady-state solution.

· We conduct extensive experiments across varied scenarios using diverse real-world datasets to validate the effectiveness and accuracy of our proposed method.

The rest of this chapter is organized as follows. Chapter 5.1 presents the game-theoretical model of the data collection game. Chapter 5.2 constructs the analytical model of the infinite collection game. Chapter 5.3 discusses the scenario where the system has a non-deterministic utility function. Chapter 5.4 shows the experimental results. Finally, we conclude this chapter in chapter 5.5.

5.1 Game-Theoretic Model Formulation

5.1.1 Threat Model

Attack Model. We assume that the attacker possesses an equivalent level of information as the data collector. This implies that the attacker has full knowledge of the strategy employed by the data collector in the previous round, for example, the data collector's trimming positions. The attacker is also in agreement with the data quality standards set by the data collector, and they are acutely aware of how the poison values they send are treated. In other words, we adopt a **white-box attack** as our attack model, which corresponds to a game with complete information. Conversely, should the attacker lack the capability to ascertain the data collector's strategy and data quality standards from the previous round, it would result in an asymmetric information scenario between the attacker and the data collector. This scenario is aligned with a black-box attack model and a game of incomplete information, which is beyond the scope of this paper.

Defensive Goal. Our game-theoretic model aims to counteract a general malicious threat model where attackers are colluding, opportunistic, and **evasive**. The term "colluding" refers to Sybil attacks, in which attackers can coordinate and share strategies to orchestrate their poison values. This situation is plausible, as these attackers may originate from a single botnet launched by one adversary. "Opportunistic" describes attackers whose goal is to maximize the deviation of estimated statistics from the ground truth, manipulating poison values to their advantage. "Evasive" pertains to attackers who are consistently rational, knowledgeable, and skilled enough to evade existing countermeasures by manipulating the poison value distribution [20]. We believe this threat model is more comprehensive (and thus more realistic) than all existing models that commonly restrict their attacking strategies or assume that the collector has any apriori knowledge of these strategies.

5.1.2 Payoff Functions

Assuming a publicly recognized data quality standard denoted by $Quality\_Evaluation()$, we establish payoff functions for both parties within the context of data manipulation attacks. Equipped with this standard, the collector can assess the intensity of poison values based on the data provided by the adversary and further determine the subsequent strategy. The existence of this metric is necessary for building up a game-theoretic model. Using this standard, let P denote the payoffs for poisoning and T for trimming. The game between the collector and the adversary is a zero-sum game where any gain for the adversary implies a loss for the collector and vice versa, i.e., $P_{collector} = -P_{adversary}$. However, the collector also incurs loss of accuracy due to incorrectly trimming honest values, denoted by $-T$. Hence, the collector's payoff function is $(-P - T)$.

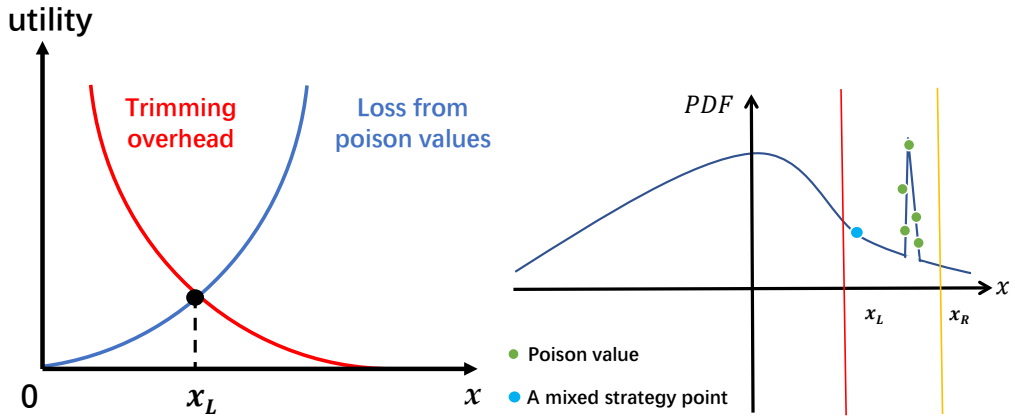
5.1.3 Strategy Space

5.1.3.1 Single Poison Value Case

This subsection discusses the strategy space for both parties. In the single value case, where the adversary injects only one poison value, their strategy is denoted by the injection point. Similarly, the collector's strategy is determined by a trimming point in the input domain. Thus, the strategy space is represented by a pair of values $(x_{\text{adversary}}, x_{\text{collector}})$ in the input domain.

Rational players do not randomly choose strategy points from the entire space. Trimming incurs loss of utility by removing benign values, while the loss from poison values increases with more malicious data injected. However, the trimming overhead decreases as more data points are removed, making the collector more cautious when trimming.

As shown in Fig. 5.1 (a), there exists a tradeoff between the loss caused by poison



(a) x_L at the balance point (b) a mixed strategy point

Figure 5.1: The definition of x_L, and arbitrary poison value distributions represented by a mixed strategy point

values and the overhead caused by trimming. A balance point, denoted by x_L, is present, such that $P(x_L) = T(x_L)$. This balance point is where the payoff for the collector and the adversary is equal, and below which the collector is not motivated to trim the data any further. In other words, a rational collector would only trim the data up to a certain point where the benefits of trimming outweigh the costs, and below that point, she would accept the risk of data poisoning to retain the accuracy. In contrast, the collector evaluates the largest acceptable value, beyond which she will definitely remove any values so that any rational adversary will not inject poison values outside of that point. As shown in Fig. 5.2, let x_R denote the maximum value that, according to the collector's belief, the adversary is willing to inject. Therefore, we have:

Definition 6. *Let $[x_L, x_R]$ be the domain of poison values. We say the adversary plays soft if he injects poison values near x_L and gains $\underline{P}$, and he plays hard if he injects poison values near x_R and gains $\overline{P}$. Conversely, the collector plays soft if she trims near x_R and gains $-\underline{T}$, and she plays hard if she trims near x_L and gains $-\overline{T}$.*

Let the (x_c, x_a) pair denote the strategies chosen by both parties, where x_c denotes

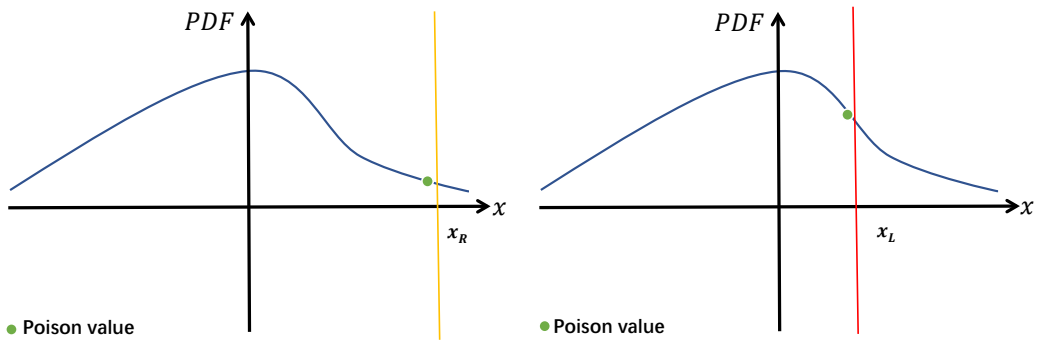


Figure 5.2: Definition of x_L and x_R for a single poison value

the trimming point, and x_a denotes the point at which the adversary decides to inject poison values, and both x_c and x_a fall in the domain $[x_L, x_R]$. It is important to note that the strategy space is complete for both the collector and the adversary, i.e., any strategy in the domain can be chosen by both parties.

5.1.3.2 General Case

Now we discuss the general case where any poison value distribution can be deployed. Without loss of generality, any point x_p in the domain $[x_L, x_R]$ can be represented as a linear combination of x_L and x_R, i.e., there exists p_L and p_R such that $x_p = p_L x_L + p_R x_R$. This can also be viewed as a **mixed strategy** in the sense of game theory, that is, the player chooses to play x_L with probability p_L, and x_R with probability p_R. As such, any single poison value injection strategy over $[x_L, x_R]$ can be reduced to a mixed strategy represented by $p_L x_L + p_R x_R$.

Since all factors in this linear combination of x_L and x_R are additive, any poison value can be reduced to a single point in the strategy space. As illustrated in Fig. 5.1 (b), we assert that any poison value distribution defined on $[x_L, x_R]$ can be reduced to a mixed strategy of a single poison value. As such, the strategy space for both the collector and the adversary is complete in this general case.

Table 5.1: The payoff matrix of the ultimatum game, $\overline{P} > \overline{T} >> \underline{P} > \underline{T} > 0$

		Adversary	
		Soft	Hard
Collector	Soft	$(-\underline{P} - \underline{T}, \underline{P})$	$(\overline{P} - \underline{T}), -\overline{P}$
	Hard	$(-\overline{T}, 0)$	$(-\overline{T}, 0)$

5.1.4 Sequential Moves

In a scenario where the data collection process consists of a single round, it embodies a strategic game. Here, both the attacker and the defender simultaneously select their strategies, resulting in a straightforward and uncomplicated Nash equilibrium. As depicted in Table 5.1, this situation mirrors the prisoner's dilemma, culminating in a unique equilibrium wherein both the adversary and the player opt for a tough stance, despite a gentler approach being mutually beneficial.

However, in real-world applications, data collection tends to be a continuous, multi-round process. This can be represented as a Stackelberg game, characterized by sequential moves where one participant's actions follow those of the other. This structure fosters cooperation, given that players can retaliate against defection, particularly when the number of rounds is indefinite or unknown. By opting for a gentler approach rather than a tough one, players can achieve a globally optimal state. The intricacies of this infinite game will be examined and modeled in detail in the following section.

5.2 Infinite Collection Game

5.2.1 Overview

In the previous section, we emphasized the utmost importance of transforming the collection process into an infinite, roundwise repeated game to foster cooperation between the collector and the adversaries. When dealing with a limited-round scenario, wherein the game is confined to a specific number of rounds, denoted as N, adversaries may be tempted to defect in the final round, triggering a domino effect of defections from the second-to-last round backwards. To address this critical issue, the game must be ingeniously designed to encompass an infinite number of rounds, thereby ensuring continuous data collection.

Fig. 5.3 overviews such infinite game, wherein a data collector gathers data from the data stream (step ③), and an adversary attempts to inject poison values into the collected data along with normal users (step ②). A public board, accessible to the adversary, enables the collector to record the untrimmed data (step ①, ⑥). In each round, the collector collects and trims the same amount of data (step ④), and then determines the trimming threshold in the next round ⑤.

Under this framework, each round becomes an invaluable opportunity to build trust, foster cooperation, learn from past experiences, and adapt strategies accordingly. The infinite nature of this repeated game actively promotes cooperation among game-theoretically rational players, paving the way for the emergence of trust and culminating in mutually beneficial outcomes in the long run.

5.2.2 Analytical Model

To formalize the infinite collection game, we employ the principle of least action from analytical mechanics. As the game involves an infinite number of samples collected

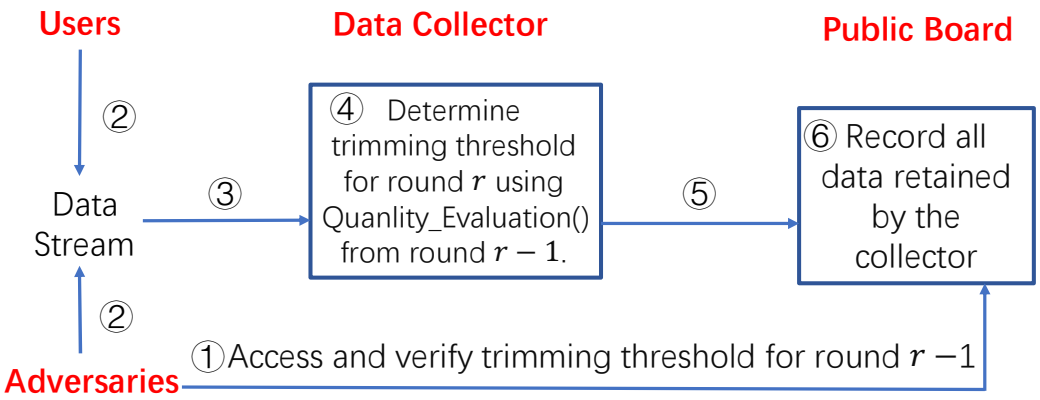


Figure 5.3: An overview of the infinite game

in an infinite number of rounds, it can be viewed as a streaming process with a fixed number of samples gathered in each round. Consequently, the parameter r can be regarded as a continuum, functioning as a timer within our system, analogous to the role of time t in classical mechanics.

The utility functions of the adversary and the collector, u_a and u_c, respectively, are natural coordinates that uniquely determine the state of the system. These functions are cumulative effects over r and can be treated as continuous and differentiable functions of r. With this setting, the evolution of the system shares the same spatiotemporal structure as classical mechanics, where the general coordinate is replaced by the utility functions of both parties, and time t is replaced by round r. We then have the fundamental principle of the infinite collection game:

Axiom 1. *The state of the infinite collection game is determined by the least action principle, which is similar to equation 2.3:*

$$\delta S = \delta \int_{r_1}^{r_2} \mathcal{L}(u_a(r), u_c(r), \dot{u_a}(r), \dot{u_c}(r), r)dr = 0, \tag{5.1}$$

where $\dot{u_a} = \frac{du_a}{dr}$ *and* $\dot{u_c} = \frac{du_c}{dr}$ *are generalized velocities, and* $\mathcal{L}$ *is the Lagrangian.*

And we have:

Lemma 2. *The Euler-Lagrange equation of equation 5.1 is given by:*

$$\frac{\partial \mathcal{L}}{\partial u_a} - \frac{d}{dr}\left(\frac{\partial \mathcal{L}}{\partial \dot{u_a}}\right) = 0, \frac{\partial \mathcal{L}}{\partial u_c} - \frac{d}{dr}\left(\frac{\partial \mathcal{L}}{\partial \dot{u_c}}\right) = 0 \tag{5.2}$$

5.2.3 Equilibrium State

From equation 5.2, we can derive some immediate results regarding the Stackelberg equilibrium and the behavior of the collector and the adversary in the infinite collection game. When we reach a Stackelberg equilibrium, it is already a convergence state that occurs after infinite iterations of responding to each other's actions. As such, if such a convergence exists, there is no interaction between the collector and the adversary, as if they are evolving independently. Therefore, we have:

Lemma 3. *The Lagrangian of the system is additive to that of u_a and u_c, that is, $\mathcal{L} = \mathcal{L}(u_a) + \mathcal{L}(u_c)$.*

Since the Lagrangian should keep the form unvaried with respect to translation of r and u, that is, we should arrive at the same Stackelberg equilibrium for any outset we choose for r and u. From this, we have:

Theorem 7. *$\mathcal{L} = \mathcal{L}(\dot{u}^2)$ and $\dot{u} =$ constant for any Stackelberg equilibrium state.*

Proof. As $\mathcal{L}$ is only a function of the magnitude of $\dot{u}$ and is independent of the direction of $\dot{u}$, we have $\mathcal{L} = \mathcal{L}(\dot{u}^2)$. Since the Lagrangian is uniform with respect to both r and u, it can only be an explicit function of $\dot{u}$, i.e., $\mathcal{L} = \mathcal{L}(\dot{u})$. Substitute this into equation 5.2, we have $\frac{\partial \mathcal{L}}{\partial u_i} = 0$. The Euler-Lagrange function can then be written as $\frac{d}{dr}\left(\frac{\partial \mathcal{L}}{\partial \dot{u_i}}\right) = 0$. As $\frac{\partial \mathcal{L}}{\partial \dot{u_i}}$ is only a function of u, we have $\dot{u} = constant$. This completes the proof. □

Finally, we attain the form of the Lagrangian for the Stackelberg equilibrium state:

Theorem 8. *The Lagrangian of any Stackelberg equilibrium state can be written as* $\mathcal{L} = m_a \dot{u_a}^2 + m_c \dot{u_c}^2$*, where* m_a *and* m_c *are two factors regarding the adversary and the collector.*

Proof. Consider an infinitesimal increment $\delta\dot{u}$ of $\dot{u}$ in the Lagrangian $\mathcal{L}$. According to Theorem 7, it corresponds to a Lagrangian of the form

$$\mathcal{L}' = \mathcal{L}((\dot{u} + \delta\dot{u})^2). \tag{5.3}$$

Expanding it as a power series in terms of $\delta\dot{u}$ and neglecting higher order terms, considering Lemma 3, we have:

$$\mathcal{L}((\dot{u} + \delta\dot{u})^2) = \mathcal{L}(\dot{u}^2) + 2\frac{\partial \mathcal{L}}{\partial \dot{u}^2}\dot{u}\delta\dot{u}. \tag{5.4}$$

According to Theorem 7, at the Stackelberg equilibrium state, $\dot{u}$ is constant. Choosing different values of $\dot{u}$ as the origin yields Euler-Lagrange equations with the same form. This means that the difference in their Lagrangians, $2\frac{\partial \mathcal{L}}{\partial \dot{u}^2}\dot{u}\delta\dot{u}$, must be a total derivative with respect to r. Therefore, when substituted into equation 5.1, $\frac{d}{dr}2\frac{\partial \mathcal{L}}{\partial \dot{u}^2}\dot{u}\delta\dot{u}$ can be eliminated in $\delta S = 0$, resulting in the same Euler-Lagrange equation. Hence, $2\frac{\partial \mathcal{L}}{\partial \dot{u}^2}\dot{u}\delta\dot{u}$ and $\dot{u}$ are linearly dependent. It follows that $\frac{\partial \mathcal{L}}{\partial \dot{u}^2}$ is independent of velocity, therefore we have:

$$\mathcal{L} = m\dot{u}^2/2, \tag{5.5}$$

where m is a proportionality constant related to the intrinsic properties of the system. Since there are two parties, according to Lemma 3, we can express the overall Lagrangian as:

$$\mathcal{L} = m_a \dot{u_a}^2/2 + m_c \dot{u_c}^2/2. \tag{5.6}$$

This completes the proof. □

Referring to the intrinsic factors associated with the attributes of both parties, it is important to acknowledge that m_a and m_c serve merely as two logical concepts employed to depict the system's converged state. Remarkably, they are not necessary

for the determination of our strategy, specifically the trimming threshold, as will be evident in the forthcoming derivations.

5.2.4 Non-equilibrium State

A system is in a non-equilibrium state if there is a permanent non-zero interaction between the collector and the adversary, where they continuously respond to each other's last response. To mathematically describe this interaction, a term $U(u_a, u_c)$ is added to the Lagrangian, which is a function of the positions u_a and u_c of the collector and the adversary, respectively. Therefore, where interaction exists between the collector and the adversary, the Lagrangian can be written as:

$$\mathcal{L} = m_a \dot{u_a}^2 + m_c \dot{u_c}^2 + U(u_a, u_c) \tag{5.7}$$

The interaction term $U(u_a, u_c)$ objectively reflects the response strength of a particular strategy in relation to deviations in data quality within the practical context. It quantifies the interaction effect between the user's action and the counteraction, depending on the scenario in which it is applied. In the upcoming section, we will derive the differential equation of the infinite game according to a given form of U.

5.3 Non-deterministic Utility

The Tit-for-tat strategy in game theory mirrors an opponent's previous action in repeated games, fostering cooperation by rewarding cooperation and punishing defection. In Section 5.1, we assume a commonly acknowledged data quality norm for both parties. However, in some practical scenarios, the utility function of a data collection system may be non-deterministic, meaning the system's outcome cannot be predicted with certainty even with known inputs.[1] Directly applying Tit-for-tat

[1]This often occurs in privacy-preserving systems using LDP for data collection, where participants add random noise to their data before sharing. While protecting sensitive information, the noise

to data collection could inadvertently trigger early termination of data exchange due to the probabilistic nature of data quality assessment. This vulnerability is inherent when using Tit-for-tat in its pure form. To avoid early termination, we propose the Elastic strategy as a variant of Tit-for-tat tailored for systems with uncertain outcomes.

It should be noted that numerous variants of Tit-for-tat exist, such as Tits-for-two-tats [3] and Generous Tit-for-tat [60]. They can also be adapted through Elastic strategies for repeated games with uncertainty. For simplicity, this paper focuses the discussion on the original Tit-for-tat. The insights can be readily extended to other variants of Tit-for-tat.

5.3.1 Tit for Tat Strategy

The data collector selects the following parameters: Tth, the trimming threshold; *Quality_Evaluation()*, which measures the quality of the data X_i received in the i-th round; *Round_no*, which represents the number of data collection rounds; *Quality_Evaluation(X_0)*, the triggering condition; and *Red*, a redundancy to ensure that the termination round is not too small. The procedure of Titfortat is given in Algorithm 9.

It is easy to apply when utility is deterministic. As a trigger strategy requiring permanent termination of cooperation upon betrayal, we have the interaction term $U(u_a, u_c)$ for the Tit-for-tat strategy becomes $U(u_a, u_c) = 0$, if $u_a = u_c$ and otherwise $U(u_a, u_c) = \infty$. In non-deterministic utility scenarios, cooperation termination may be triggered by normal jitter even if both parties are cooperative. Intuitively, the collector should compromise their roundwise gain to preserve redundancy and maximize long-run benefit.

In the Stackelberg equilibrium, we assume that both the collector and the adversary

renders the system outcome probabilistic.

Algorithm 9 Titfortat Strategy

Input: *Quality_Evaluation()*, X_0, *Red*, $\underline{T}$, $\overline{T}$, *Round_no*
Output: *Round_terminate*
$Tth \leftarrow \underline{T}$
Round_terminate $\leftarrow$ *Round_no*
for $i \leftarrow 1$ **to** *Round_no* **do**
    if *Quality_Evaluation*$(X_i) <$ *Quality_Evaluation*$(X_0) +$ *Red* **then**
        $Tth \leftarrow \overline{T}$
        Round_terminate $\leftarrow i$
        break
    end if
end for
return *Round_terminate*

have a symmetric setting. This means that if u_a and u_c are symmetric, the solution should also be symmetric. Let $g_c = \overline{T} - \underline{P} - \underline{T}$ and $g_a = \overline{P}$ be the roundwise gain for both parties during cooperation (which is the payoff of compliance minus betrayal), and $g_{ac} = \frac{g_a + g_c}{2}$ due to the symmetry axiom. The collector now expects a gain of $g_0 = g_{ac} - \delta$, where δ is a compromise in data utility. If the adversary complies, the collector can observe compliance deterministically since the probability of the data utility being less than g_0 in the outcome is negligible. However, if the adversary defects at g_{ac}, the collector judges compliance with probability p and defects with probability $1 - p$ due to the perturbation's probabilistic nature. With these settings, we derive the following theorem concerning the Stackelberg equilibrium for the Tit-for-tat strategy:

Theorem 9. *The condition for the adversary choosing to comply in the Tit-for-tat game is* $\delta < \frac{d-dp}{1-dp} g_{ac}$*, where* d *denotes the roundwise discount rate of data utility acknowledged by both parties.*

Proof. From the adversary's perspective, their current-round gain expectation when choosing to comply is

$$g_{com} = g_0 + dg_{com}, \quad \text{or} \quad g_{com} = \frac{g_0}{1-d}. \tag{5.8}$$

However, if the adversary opts to defect, they will be assessed in the subsequent round as complying with probability p and defecting with probability $1-p$. As a result, their current-round gain expectation becomes

$$g_{def} = g_{ac} + dpg_{def}, \quad \text{or} \quad g_{def} = \frac{g_{ac}}{1-dp}. \tag{5.9}$$

The adversary will decide to comply if and only if $g_{com} > g_{def}$, which is equivalent to $g_0 > \frac{1-d}{1-dp}g_{ac}$, or $\delta < \frac{d-dp}{1-dp}g_{ac}$. This completes the proof. □

Should $p = 1$, implying that the adversary is never identified as defecting, they would always opt to defect given the lack of consequences. In contrast, as $p \to 0$, signifying an increased likelihood of the adversary being flagged as defecting, a substantial adjustment must be made to δ to cultivate trust. This analysis underscores the complexities and trade-offs inherent in managing non-deterministic utility situations within data collection systems. The delicate balance between cooperation, trust, and data utility is crucial to the system's sustained success. Hence, given $\overline{T}, \underline{T}, \overline{P}, \underline{P}, p, d$, one can ascertain the Tth of Tit-for-tat by selecting a δ according to their preference.

5.3.2 Elastic Trigger Strategy

So far, we have discussed the equilibrium of the Tit-for-tat strategy, which is essentially a rigid trigger strategy. Unfortunately, while preserving redundancy effectively extends the period of cooperation, we should note that the game cannot achieve infinite rounds, as the probability of termination keeps increasing and will ultimately converge to 1 in the long run. A simple way to tackle this is to allow the trigger strategy to be elastic with forgiveness, namely, applying a penalty in the next round

when a defection is detected instead of terminating the cooperation directly. This is shown in Algorithm 10.

Algorithm 10 Elastic Trigger Strategy

1: **Input:** *Quality_Evaluation()*, $\underline{T}$, $\overline{T}$, *Round_no*, k
2: **Output:** Tth_i
3: $Tth_1 \leftarrow \underline{T}$
4: $QE_i = \frac{Quality\_Evaluation(X_i)}{\max(Quality\_Evaluation(\cdot))}$
5: **for** $i \leftarrow 2$ **to** *Round_no* **do**
6: $\quad Tth_i = (1 - k \times QE_i) \times \underline{T} + k \times QE_i \times \overline{T}$
7: **end for**
8: **return** Tth_i

That implies an interaction between the adversary and the collector exists, and an equilibrium position exists such that the interaction pulls the relative utility $|u_a - u_c|$ back to the equilibrium position by a force equal to $-\frac{\partial U}{\partial u_a}$ or $-\frac{\partial U}{\partial u_c}$. We expand this interaction into a power series about $(u_a - u_c)$. When the deviation between the two is small, $u_a - u_c$ is also small, so only the first non-zero item is reserved. This is a quadratic term, and we introduce a proportionality constant k that describes the strength of the interaction. Therefore, we have:

Definition 7. *The interaction term $U(u_a, u_c)$ for the elastic trigger strategy is*

$$U(u_a, u_c) = k(u_a - u_c)^2/2. \tag{5.10}$$

According to this, we have the following theorem:

Theorem 10. *The utility functions of the adversary and the collector periodically oscillate with respect to r in the setting of the elastic strategy.*

Proof. The Lagrangian of this system is given by

$$\mathcal{L} = m_a \dot{u_a}^2 + m_c \dot{u_c}^2 + k(u_a - u_c)^2/2 \tag{5.11}$$

by plugging equation 5.10 into equation 5.7. Applying equation 5.2 to this, we have

$$m_a\ddot{u_a} + k(u_a - u_c) = 0, m_c\ddot{u_c} + k(u_a - u_c) = 0 \tag{5.12}$$

These two equations have the same form of ordinary differential equations concerning that of a double harmonic oscillator system, where two masses m_a and m_c are connected by a spring with spring constant k. The solution will also be the same, in the form of

$$u(r) = A\cos(\omega r + \phi). \tag{5.13}$$

This completes the proof. □

5.4 Experiments

In this section, the performance of the proposed approach is evaluated through its application to real-world datasets. Experiments are implemented in MATLAB R2021b on a desktop computer with Intel i7-10700K RTX 3090 eight-core CPU, 128GB RAM, and Windows 10 OS.

5.4.1 Experimental Setup

Datasets. In our experiments, we use 5 real-world numerical datasets. **Control**, **Vehicle**, and **Letter** [79] are standard UCI datasets, frequently used in machine learning research. **Taxi** [69], extracted from the January 2018 New York Taxi data, records the pick-up times during a day and includes 1,048,575 integers from 0 to 86,340, normalized to the range $[-1, 1]$. **Creditcard** [2] comprises numerical results of PCA transactions, which are sanitized to preserve confidentiality. A summary of all dataset information is shown in Table 5.2.

Parameter Settings. In order to standardize the description of our approach across different datasets, we describe the positions of poison value injection and trimming

Table 5.2: Dataset Information

Dataset	Instances	Features	Clusters
CONTROL	600	60	6
VEHICLE	752	18	4
LETTER	20000	16	26
TAXI	1048575	1	1
CREDITCARD	284807	31	4

in terms of data percentiles. The position for each trimming round is determined by the parameter Tth.

We implement several benchmark schemes. Groundtruth represents the result obtained by running the original dataset without any poison value injection. Ostrich assumes no defensive measures are taken, i.e., accepting all poison values. Since the adversary is also aware of this, the poison value is injected at the 99th percentile in each round. We also implement two baseline defence schemes where the data collector sets static thresholds. In the $Baseline_{0.9}$ scheme, the adversary randomly injects poison values in the percentile range of [0.9, 1], while in the $Baseline_{static}$ scheme, the adversary injects poison values at the percentile $(Tth - 1\%)$. The latter is the ideal attack, which assumes that the adversary has the ability to accurately determine the data collector's Tth for each round and always adds poison values at the location that benefits itself the most.

We implement our three proposed schemes, namely Titfortat, $Elastic_{0.1}$, and $Elastic_{0.5}$. These schemes employ different strategies for setting the trimming and injection positions for poison values, with varying levels of adaptability based on previous adversary actions. In the Titfortat scheme, the untriggered trim position is set at the $(Tth+1\%)$ percentile, but once the adversary triggers the judgement, the subsequent rounds will always be trimmed at the $(Tth - 3\%)$ percentile. In the Elastic schemes, the initial trim position is set at the $(Tth - 3\%)$ percentile, and the initial injection position

for poison value is set at the $(Tth + 1\%)$ percentile. In the subsequent rounds, the data collector dynamically adjusts the trimming position for the next round based on the previous round's adversary's poison injection position $A(i)$, according to the rule $T(i+1) = Tth + k(A(i) - Tth - 1\%)$, while the adversary adjusts the poison injection position for the next round $A(i+1)$ based on the previous round's data collector's trimming position $T(i)$, according to the rule $A(i+1) = Tth - 3\% + k(T(i) - Tth)$. $Elastic_{0.1}$ and $Elastic_{0.5}$ represent the parameter k taking the values 0.1 and 0.5, respectively.

5.4.2 Stackelberg Equilibrium Results on k-Means Clustering

This subsection presents the clustering results from k-means applied to **Control**, **Vehicle**, and **Letter**. We compare the performance when both the data collector and the adversary follow Stackelberg equilibrium strategies. For each experiment, we consider 20 rounds of games, with results averaged over 100 repetitions. Titfortat is assumed not to experience early terminations. Three attack ratio intervals are [0, 0.01], [0.05, 0.15], and [0.2, 0.5], corresponding to the situations where there are few, moderate, and many poison values, respectively. Results under different attack ratios are named after the dataset name and the corresponding attack ratio interval, for example, $\textbf{Control}_{[0,0.01]}$.

Fig. 5.4 illustrates the results when the Tth is set to 0.9. The y-axis in this chart depicts two distinct measurements: the Sum of Squares Errors (SSE) and Distance. SSE is defined by the equation $SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, where y_i represents the observed values and $\hat{y}_i$ stands for the predicted values. On the other hand, 'Distance' illustrates the discrepancy between the actual centroid of the clustering and the ground truth, as measured by the Euclidean distance. We observe that during intervals of low attack ratios, the volume of poison values is minimal. As such, Ostrich performs

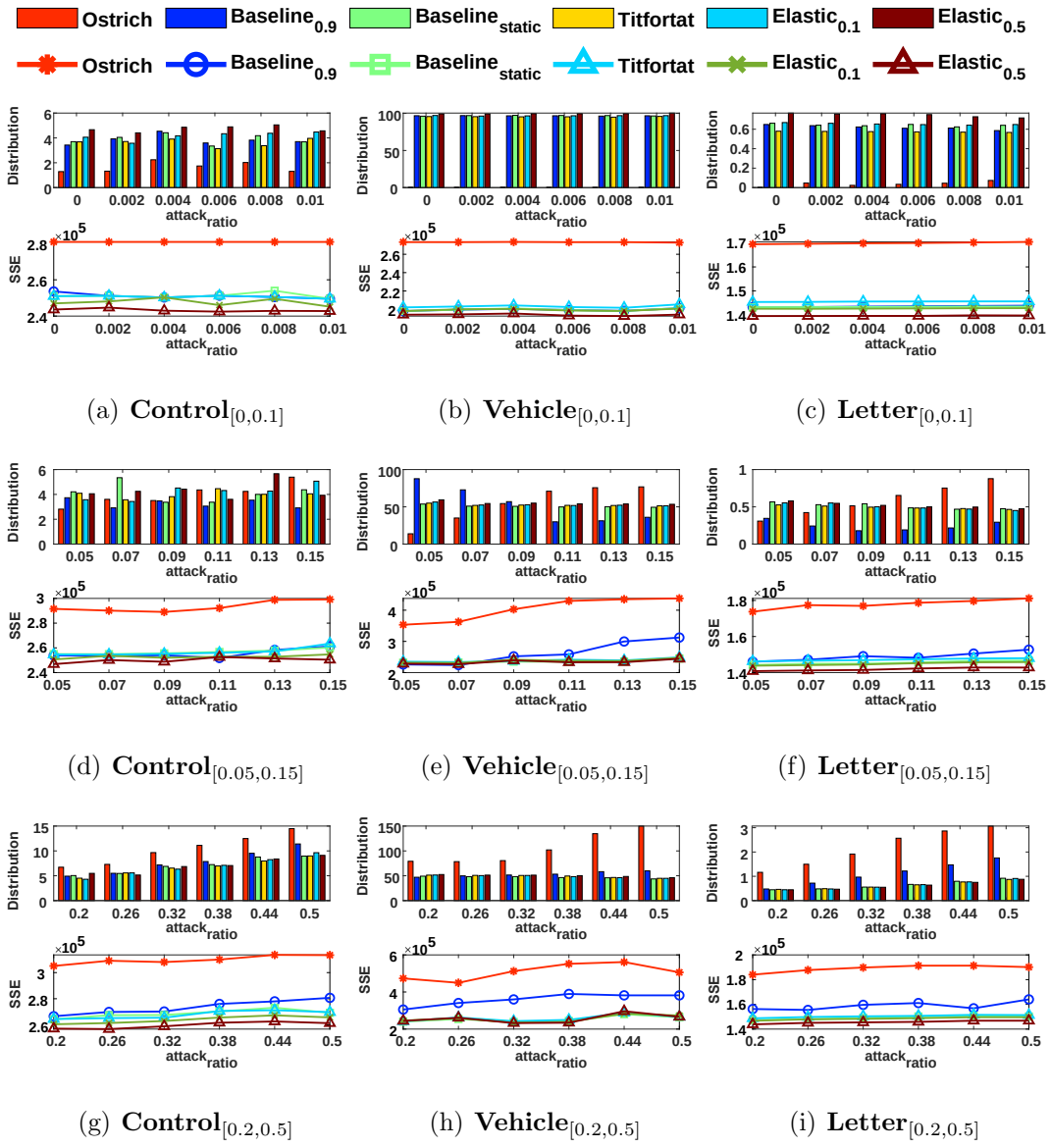
optimally and manifests the smallest offset. In such situations, all schemes implementing trimming end up with additional overhead costs. As the attack ratio escalates, pushing into a grey area where the impact of trimming to eliminate poison values is counterbalanced by false positives, the performance of the Ostrich scheme gradually deteriorates. In contrast, when the attack ratio falls within a large interval, where poison values become dominant, our proposed schemes significantly outperform both baseline schemes. Also, it is evident that Ostrich has the highest SSE. Moreover, in almost all the cases presented, our proposed solutions outperform both baseline approaches, with $Elastic_{0.5}$ demonstrating the best performance.

The results when the Tth is adjusted to 97% are depicted in Fig. 5.5, from which similar conclusions can be drawn. Here, the trimming method adopted is more conservative, thus diminishing the overhead at lower attack ratios. However, the effectiveness of this approach becomes less distinct at higher attack ratios.

5.4.3 Stackelberg Equilibrium Results on SVM and SOM Classifier

This subsection validates the on labelled datasets with regard to Support Vector Machine (SVM) and Self-Organizing Map (SOM) classifiers, respectively. SVM and SOM are both classifiers included within MATLAB. We process various datasets and use them as inputs, directly showcasing the classification results. Specifically, we set the number of neurons in SOM to $20 \times 20 = 400$. The color depth between adjacent neurons represents their distance, with darker colors signifying greater distances between neurons. All elements classified into the same class have relatively small distances between them.

The SVM experiment is carried out exclusively on **Control (with labels)**. The parameters are fixed at Tth=0.95 and attack ratio=0.4. Fig. 5.6 (a) illustrates the ground truth of SVM classification, while Fig. 5.7 provides a comparison of SVM clas-

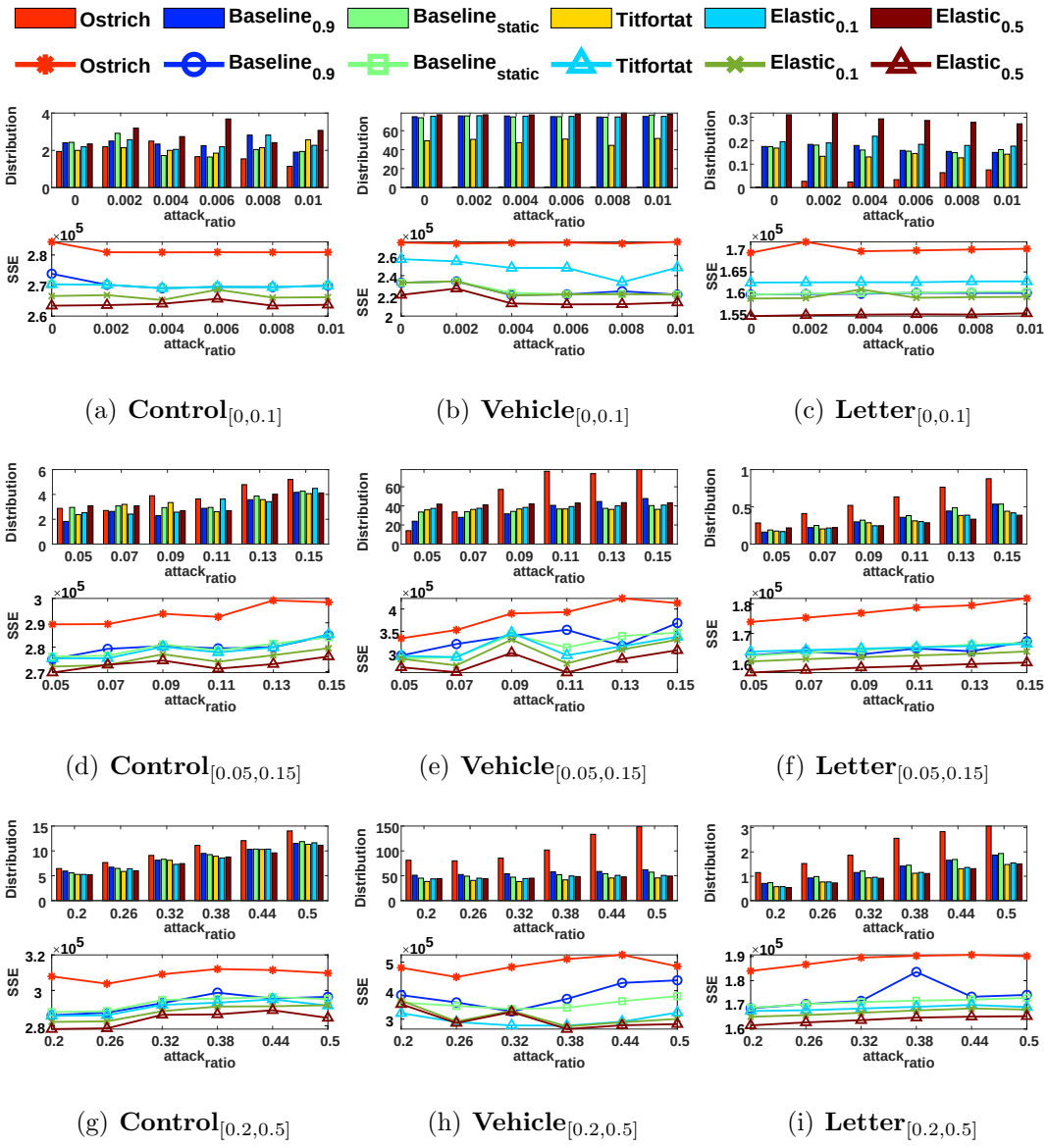


(a) **Control**$_{[0,0.1]}$ (b) **Vehicle**$_{[0,0.1]}$ (c) **Letter**$_{[0,0.1]}$

(d) **Control**$_{[0.05,0.15]}$ (e) **Vehicle**$_{[0.05,0.15]}$ (f) **Letter**$_{[0.05,0.15]}$

(g) **Control**$_{[0.2,0.5]}$ (h) **Vehicle**$_{[0.2,0.5]}$ (i) **Letter**$_{[0.2,0.5]}$

Figure 5.4: K-means clustering results over **Control**, **Vehicle**, and **Letter**, Tth=0.9

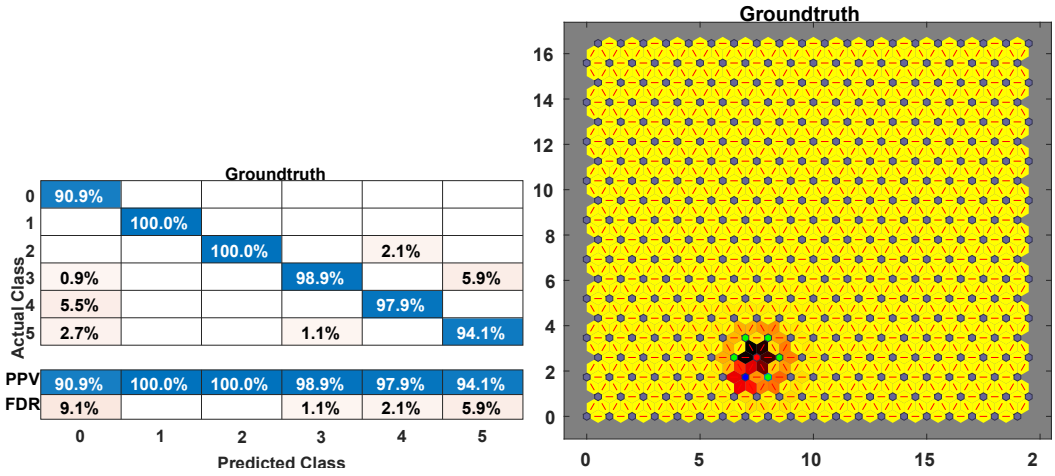


(a) **Control**$_{[0,0.1]}$ (b) **Vehicle**$_{[0,0.1]}$ (c) **Letter**$_{[0,0.1]}$

(d) **Control**$_{[0.05,0.15]}$ (e) **Vehicle**$_{[0.05,0.15]}$ (f) **Letter**$_{[0.05,0.15]}$

(g) **Control**$_{[0.2,0.5]}$ (h) **Vehicle**$_{[0.2,0.5]}$ (i) **Letter**$_{[0.2,0.5]}$

Figure 5.5: K-means clustering results over **Control**, **Vehicle**, and **Letter**, Tth=0.97



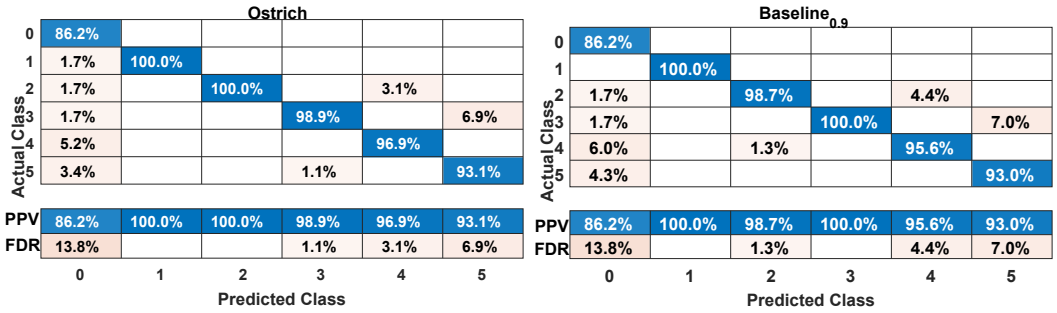
(a) Ground truth of SVM (b) Ground truth of SOM

Figure 5.6: The ground truth of SVM and SOM classification

sification methods. The results are quite clear: the ground truth achieves an average accuracy of 96.8%, while the various approaches under comparison yield respective accuracies of 95.5%, 95.1%, 94.9%, 96.1%, 95.6%, and 95.7%. The first three strategies comprise Ostrich and two Baselines. It is evident that $Baseline_{static}$ exhibits the poorest performance, even falling behind Ostrich. $Baseline_{0.9}$ also underperforms compared to Ostrich, a consequence of trimming excessive amounts of useful data. Our three approaches outperform others in terms of accuracy.

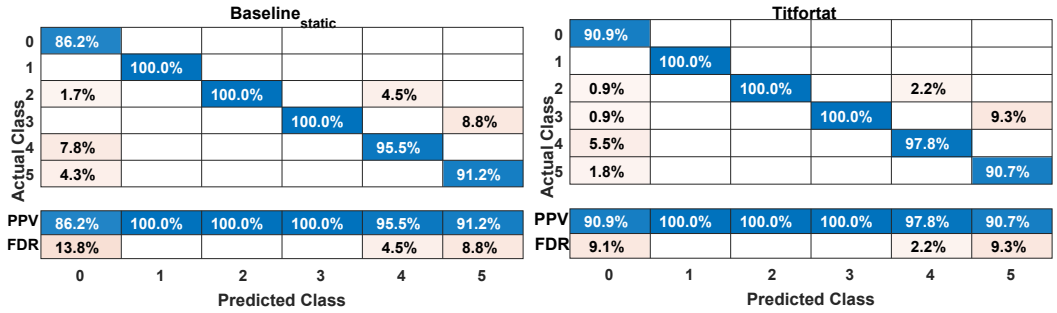
We carry out SOM classification on **Creditcard**, which contains credit card consumption data. The ground truth classification of this dataset, divided into four classes, is depicted in Fig. 5.6 (b). The classification results exhibit significant skewness and can be interpreted as follows:

The vast majority of data points belong to the same class, signifying the general public. The two isolated points, colored red and blue, are notably distant from other classes, representing fraudulent and premium users, respectively. The figure also includes five green points that symbolize a distinct category. These points are distant from both fraudulent and premium users, so they exhibit behaviors different from the



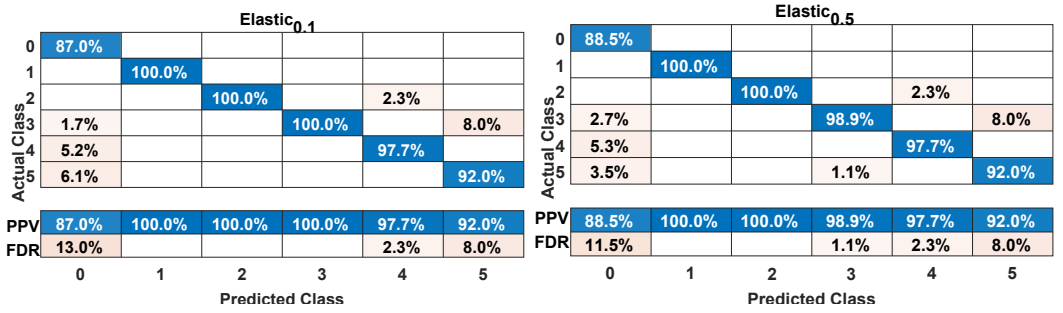
(a) *Ostrich*

(b) $Baseline_{0.9}$



(c) $Baseline_{static}$

(d) $Titfortat$



(e) $Elastic_{0.1}$

(f) $Elastic_{0.5}$

Figure 5.7: Comparison of SVM classification

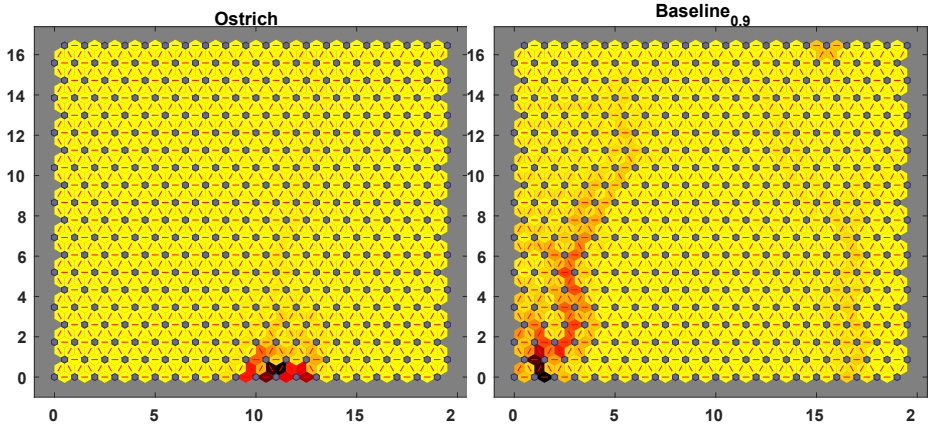
general public. We can reasonably infer that these data points represent a segment of the general public with potential to evolve into high-value customers over time.

Fig. 5.8 presents a comparison of SOM classification results. We observe that Ostrich entirely disregards the large class corresponding to the green points. $Baseline_{0.9}$ performs worse than Ostrich, as it not only failed to differentiate the class corresponding to the green points but also lost the unique characteristics of the two isolated smaller classes. Though $Baseline_{static}$ successfully divides the data into four classes, it only includes a single isolated point, and the other three classes are overrepresented. Titfortat omits one isolated point and expands the area of the original green class. $Elastic_{0.1}$ and $Elastic_{0.5}$ each drop one isolated point but effectively represent the unique characteristics of the original class corresponding to the green points.

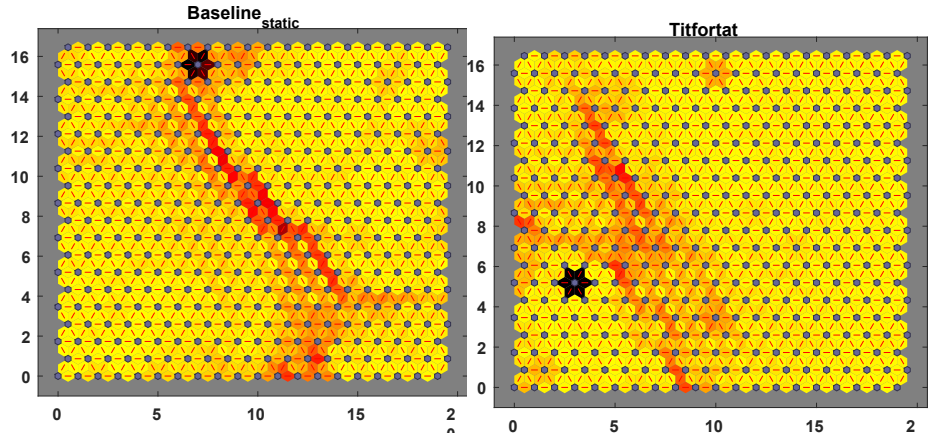
5.4.4 Non Equilibrium Results and Cost Analysis

This subsection evaluates the utility under conditions where the adversary opts not to follow Stackelberg equilibrium strategies. The experiment is conducted on **Control** with attack ratio 0.2, involving 20 rounds of games. Given that all strategies can be represented as mixed strategies comprised of two linear combinations, we establish the 99th and 90th percentiles as the bases for these combinations, manipulated by parameter p. Poison values are injected at the 99th percentile with a probability of p and at the 90th percentile with a probability of $1 - p$.

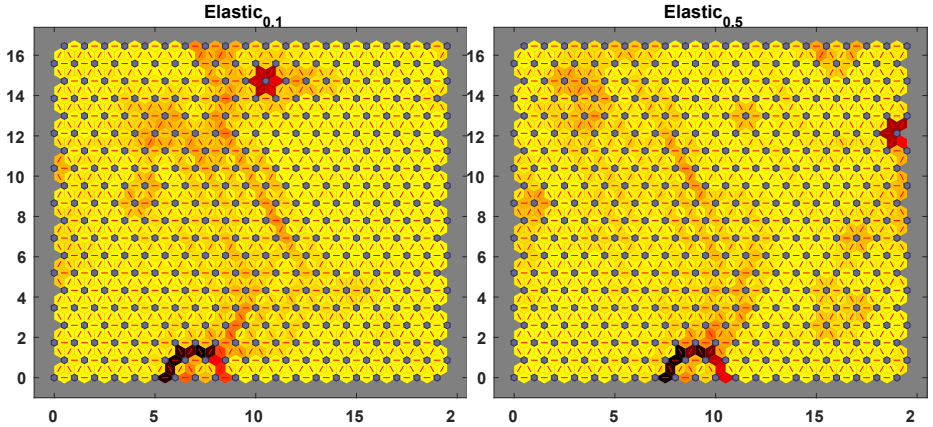
In order to examine the early termination of Titfortat, we allow a redundancy of 5%. This means that the stopping trigger condition is set to the initial observation where the ratio of poison values in a round exceeds $1 - p + 0.05$. Once this condition is triggered, the trimming position of Titfortat in subsequent rounds is permanently shifted to the 90th percentile. When $p = 1$, it corresponds to an adversary who consistently adheres to the Stackelberg equilibrium strategy, while $p = 0$ corresponds to an adversary who is both greedy and shortsighted. The effectiveness of any evasion



(a) *Ostrich* (b) $Baseline_{0.9}$



(c) $Baseline_{static}$ (d) $Titfortat$



(e) $Elastic_{0.1}$ (f) $Elastic_{0.5}$

Figure 5.8: Comparison of SOM classification

strategy falls between these two extremes, controlled by parameter p. The experimental results are summarized in Table 5.3. The figures under the Titfortat and Elastic columns represent the proportion of untrimmed poison values in the remaining data, while the Average Termination Rounds denote the mean number of rounds the Titfortat strategy underwent before termination. The results suggest that an adversary following the Stackelberg equilibrium strategy realizes higher utility than one who does not.

We also conduct a cost analysis for the Elastic scheme, and the experimental results are presented in Table 5.4. In the Elastic scheme, the handling of poison values involves imposing a penalty to the trimming threshold of the subsequent round based on a specified intensity and thus more rounds are required to achieve an equilibrium state. We define the cost of the Elastic scheme as the difference between the percentile of the data collector's soft trim and the actual percentile of the injected poison value before reaching equilibrium. The results displayed in the table represent the roundwise cost, which is the average cost over all rounds. As expected, the cost is higher in the initial rounds. However, as the Elastic strategy progressively adjusts the trimming threshold, the attacker's poison placement gradually approaches the equilibrium point, and the cost per round decreases accordingly. Hence, the roundwise cost diminishes with an increasing number of rounds, denoted by *Round_no*. Additionally, because the response intensity at $k = 0.5$ is greater than at $k = 0.1$, the former achieves equilibrium more rapidly, resulting in a lower roundwise cost.

5.4.5 Performance under LDP perturbations

This subsection evaluates the effectiveness of our proposed approach in privacy protection scenarios where data are perturbed by LDP techniques. For comparison, we use the Expectation-Maximization Filter (EMF) [21] as a baseline. It serves as a filtering mechanism designed to mitigate the effect of poison values under LDP data

Table 5.3: Non-equilibrium results and average termination rounds

p	Average termination rounds	Titfortat	Elastic
0	25	0.22727	0.22727
0.1	24.24	0.19157	0.22309
0.2	21.56	0.19645	0.21844
0.3	23.44	0.19264	0.21232
0.4	19.44	0.18381	0.20924
0.5	20.6	0.17904	0.20483
0.6	17.52	0.17363	0.19017
0.7	14.44	0.16874	0.17114
0.8	16.52	0.17011	0.15952
0.9	14.28	0.17041	0.15036
1	13	0.18182	0.14449

Table 5.4: Roundwise cost of $Elastic_{0.1}$ and $Elastic_{0.5}$

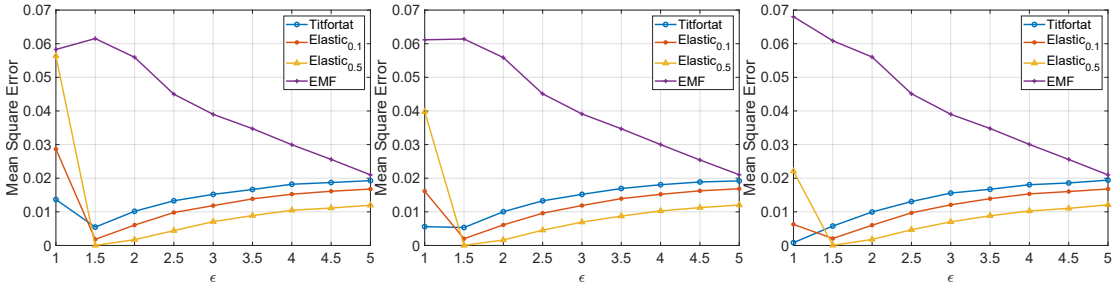
Round_no	k=0.5 (%)	k=0.1 (%)
5	0.608%	0.8%
10	0.30404%	0.43281%
15	0.20269%	0.28887%
20	0.15202%	0.21667%
25	0.12162%	0.17333%
30	0.10135%	0.14444%
35	0.086869%	0.12381%
40	0.07601%	0.10833%
45	0.067565%	0.096296%
50	0.060808%	0.086667%

collection. The experiment is conducted on **Taxi**, the same dataset used in [21]. For the adversary of EMF, we employ the input manipulation attack [20], which is identified as a potent evasion strategy against detection mechanisms within LDP-driven data collection scenarios.

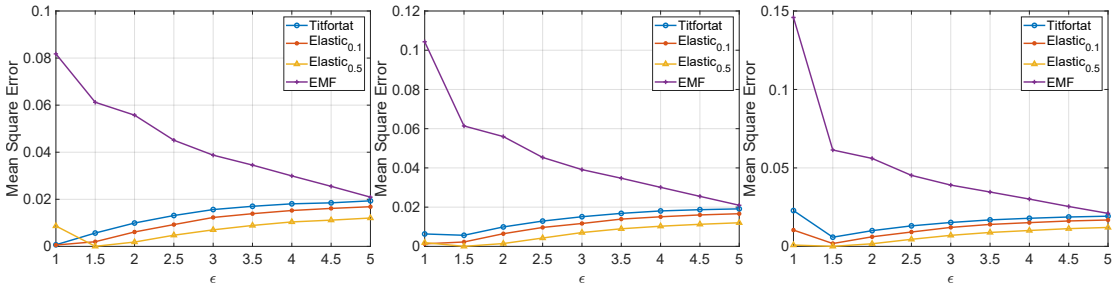
Fig. 5.9 shows the results, where the x-axis represents the privacy budget ϵ, and the y-axis indicates the Mean Square Error (MSE). As evidenced by the graph, in all parameter settings, the EMF consistently falls short of our scheme's performance. Notably, when ϵ is small, corresponding to a high perturbation intensity, the trimming scheme must accommodate increasing overhead due to false positives. This produces a notable inflection point around $\epsilon = 1.5$ in the figure, an effect that becomes eminent when the attack ratio is small.

5.5 Chapter Summary

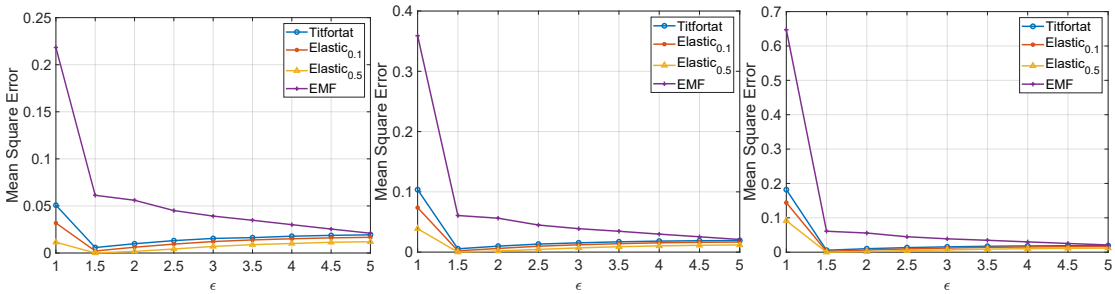
This chapter presents a comprehensive game-theoretic model to counter online data poisoning attacks by establishing a viable Stackelberg equilibrium. We utilize the trimming strategy for defense, apply theoretical physics principles to construct an analytical model, and extend the adaptability in privacy-preserving systems with non-deterministic utility functions. Our experimental results, derived from various real-world datasets, validate the effectiveness of our approach.



(a) Attack ratio=0.05 (b) Attack ratio=0.1 (c) Attack ratio=0.15



(d) Attack ratio=0.2 (e) Attack ratio=0.25 (f) Attack ratio=0.3



(g) Attack ratio=0.35 (h) Attack ratio=0.4 (i) Attack ratio=0.45

Figure 5.9: Comparison of EMF and our proposed approaches

Chapter 6

Conclusions and Future Works

6.1 Conclusions

This thesis successfully addresses the research goal of enhancing distributed system security from a data management perspective. By advancing techniques in password authentication, secure indexing, and data integrity, it significantly contributes to mitigating vulnerabilities in distributed environments. The proposed solutions demonstrate strong theoretical foundations and practical applications, providing a robust framework for data protection.

- **Enhanced Cryptographic Password Authentication.** This contribution revolves around the development of a cryptographic solution that significantly reduces the computational burden associated with handling large volumes of authentication requests. This solution not only strengthens defenses against brute-force attacks but also ensures efficient transaction processing, which is crucial in IoT-heavy environments. By optimizing both security and efficiency, this approach addresses critical needs in modern digital infrastructures.

- **Optimized Secure Indexing for Bi-attribute Datasets.** The second con-

tribution introduces a secure indexing scheme specifically designed for bi-attribute datasets, which are prevalent in AI and data analytics. This scheme effectively balances query performance with stringent data privacy requirements, utilizing advanced probabilistic data structures to minimize privacy risks from inference attacks. It enables efficient and secure data retrieval, maintaining both the usability and confidentiality of sensitive information in database systems.

- **Dynamic Game-Theoretical Model for Data Integrity.** The final contribution integrates a game-theoretical model that dynamically protects data integrity against sophisticated online manipulation. This model adapts to evolving threat landscapes in real-time, employing strategic responses to counter malicious data alteration attempts. It ensures the accuracy and reliability of data, which is vital for critical decision-making processes, thereby preventing the influence of malicious actors on system outcomes.

6.2 Future work

While making substantial progress, this thesis does not fully close the technical gap in the context of the scalability of the solutions in extremely large-scale environments and potential corner cases in highly dynamic threat landscapes. In future work, it is planed to address the limitations by improving the scalability of the authentication and indexing systems, developing adaptive models to better handle dynamic and unforeseen cyber threats and exploring additional cryptographic techniques to further strengthen data protection. It is also aimed to construct theoretical frameworks for games with incomplete information pertinent to black-box models and to provide corresponding experimental results.

References

[1] Basheer Al-Duwairi and Manimaran Govindarasu. Novel hybrid schemes employing packet marking and logging for ip traceback. *IEEE Transactions on Parallel and Distributed Systems*, 17(5):403–418, 2006.

[2] Reid A. Johnson Andrea Dal Pozzolo, Olivier Caelen and Gianluca Bontempi. Openml datasets. `https://www.openml.org/search?type=data&sort=runs&id=1597&status=active`, 2015.

[3] Robert Axelrod and William D Hamilton. The evolution of cooperation. *science*, 211(4489):1390–1396, 1981.

[4] Prithu Banerjee, Lingyang Chu, Yong Zhang, Laks VS Lakshmanan, and Lanjun Wang. Stealthy targeted data poisoning attack on knowledge graphs. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2069–2074. IEEE, 2021.

[5] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, pages 2288–2296, 2017.

[6] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 127–135, 2015.

[7] Steven Michael Bellovin and William R Cheswick. Privacy-enhanced searches using encrypted bloom filters. 2007.

[8] Giuseppe Bianchi, Lorenzo Bracciale, and Pierpaolo Loreti. " better than nothing" privacy with bloom filters: To what extent? In *International Conference on Privacy in Statistical Databases*, pages 348–363. Springer, 2012.

[9] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[10] Andrea Bittau, Ulfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 441–459, 2017.

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

[12] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Data poisoning attacks to local differential privacy protocols. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 947–964, 2021.

[13] J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2):143–154, 1979.

[14] Francis Chang, Wu-chang Feng, and Kang Li. Approximate caches for packet classification. In *IEEE INFOCOM 2004*, volume 4, pages 2196–2207. IEEE, 2004.

[15] Denis Charles and Kumar Chellapilla. Bloomier filters: A second look. In *European Symposium on Algorithms*, pages 259–270. Springer, 2008.

[16] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks

without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[17] Rui Chen, Benjamin CM Fung, S Yu Philip, and Bipin C Desai. Correlated network data publication via differential privacy. *The VLDB Journal*, 23(4):653–676, 2014.

[18] Rui Chen, Haoran Li, A Kai Qin, Shiva Prasad Kasiviswanathan, and Hongxia Jin. Private spatial data aggregation in the local setting. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 289–300. IEEE, 2016.

[19] Ruiliang Chen, Jung-Min Park, and Randolph Marchany. Nisp1-05: Rim: Router interface marking for ip traceback. In *IEEE Globecom 2006*, pages 1–5. IEEE, 2006.

[20] Albert Cheu, Adam Smith, and Jonathan Ullman. Manipulation attacks in local differential privacy. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 883–900. IEEE, 2021.

[21] R. Du, Q. Ye, Y. Fu, H. Hu, J. Li, C. Fang, and J. Shi. Differential aggregation against general colluding attackers. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 2180–2193, Los Alamitos, CA, USA, apr 2023. IEEE Computer Society.

[22] Rong Du, Qingqing Ye, Yue Fu, and Haibo Hu. Collecting high-dimensional and correlation-constrained data with local differential privacy. In *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2021.

[23] Jiawei Duan, Qingqing Ye, and Haibo Hu. Utility analysis and enhancement of ldp mechanisms in high-dimensional space. *arXiv preprint arXiv:2201.07469*, 2022.

[24] John C Duchi, Michael I Jordan, and Martin J Wainwright. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 429–438. IEEE, 2013.

[25] John C Duchi, Michael I Jordan, and Martin J Wainwright. Privacy aware learning. *Journal of the ACM (JACM)*, 61(6):1–57, 2014.

[26] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[27] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[28] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, pages 139–147, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[29] Fatih Emekci, Ahmed Methwally, Divyakant Agrawal, and Amr El Abbadi. Dividing secrets to secure data outsourcing. *Information Sciences*, 263:198–210, 2014.

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

[31] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM transactions on networking*, 8(3):281–293, 2000.

[32] Yue Fu, Qingqing Ye, Rong Du, and Haibo Hu. Collecting multi-type and correlation-constrained streaming sensor data with local differential privacy. *ACM Transactions on Sensor Networks*, 2023.

[33] Quan Geng, Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The staircase mechanism in differential privacy. *IEEE Journal of Selected Topics in Signal Processing*, 9(7):1176–1184, 2015.

[34] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[35] Shahabeddin Geravand and Mahmood Ahmadi. A novel adjustable matrix bloom filter-based copy detection system for digital libraries. In *2011 IEEE 11th International Conference on Computer and Information Technology*, pages 518–525. IEEE, 2011.

[36] Eu-Jin Goh. Secure indexes. *Cryptology ePrint Archive*, 2003.

[37] Deke Guo, Yunhao Liu, Xiangyang Li, and Panlong Yang. False negative problem of counting bloom filter. *IEEE transactions on knowledge and data engineering*, 22(5):651–664, 2010.

[38] Deke Guo, Jie Wu, Honghui Chen, and Xueshan Luo. Theory and network applications of dynamic bloom filters. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–12. IEEE, 2006.

[39] Fang Hao, Murali Kodialam, TV Lakshman, and Haoyu Song. Fast multiset membership testing using combinatorial bloom filters. In *IEEE INFOCOM 2009*, pages 513–521. IEEE, 2009.

[40] Kai Huang, Gaoya Ouyang, Qingqing Ye, Haibo Hu, Bolong Zheng, Xi Zhao, Ruiyuan Zhang, and Xiaofang Zhou. Ldpguard: Defenses against data poi-

soning attacks to local differential privacy protocols. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[41] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[42] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE symposium on security and privacy (SP)*, pages 19–35. IEEE, 2018.

[43] A John and T Sivakumar. Ddos: Survey of traceback methods. *International Journal of Recent Trends in Engineering*, 1(2):241, 2009.

[44] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur Berger. Botz-4-sale: Surviving organized ddos attacks that mimic flash crowds. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 287–300, 2005.

[45] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[46] Marius Kloft and Pavel Laskov. Security analysis of online centroid anomaly detection. *The Journal of Machine Learning Research*, 13(1):3681–3724, 2012.

[47] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *Machine Learning*, pages 1–47, 2022.

[48] Joseph Louis Lagrange. *Mécanique analytique*, volume 1. Mallet-Bachelier, 1853.

[49] Ricky Laishram and Vir Virander Phoha. Curie: A method for protecting svm classifier from poisoning attack. *arXiv preprint arXiv:1606.01584*, 2016.

[50] Tiancheng Li and Ninghui Li. On the tradeoff between privacy and utility in data publishing. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–526, 2009.

[51] Xiaoguang Li, Neil Zhenqiang Gong, Ninghui Li, Wenhai Sun, and Hui Li. Fine-grained poisoning attacks to local differential privacy protocols for mean and variance estimation. *arXiv preprint arXiv:2205.11782*, 2022.

[52] Fang Liu. Generalized gaussian mechanism for differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 31(4):747–756, 2018.

[53] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

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

[55] Sharad Mehrotra, Shantanu Sharma, Jeffrey Ullman, and Anurag Mishra. Partitioned data security on outsourced sensitive and non-sensitive data. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 650–661. IEEE, 2019.

[56] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the aaai conference on artificial intelligence*, volume 29, 2015.

[57] Alper T Mizrak, Stefan Savage, and Keith Marzullo. Detecting compromised routers via packet forwarding behavior. *IEEE network*, 22(2):34–39, 2008.

[58] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 27–38, 2017.

[59] Andrew Newell, Rahul Potharaju, Luojie Xiang, and Cristina Nita-Rotaru. On the practicality of integrity attacks on document-level sentiment analysis. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, pages 83–93, 2014.

[60] Martin A Nowak and Karl Sigmund. Tit for tat in heterogeneous populations. *Nature*, 355(6357):250–253, 1992.

[61] Georgios Oikonomou and Jelena Mirkovic. Modeling human behavior for defense against flash-crowd attacks. In *2009 IEEE International Conference on Communications*, pages 1–6. IEEE, 2009.

[62] Kerim Yasin Oktay, Murat Kantarcioglu, and Sharad Mehrotra. Secure and efficient query processing over hybrid clouds. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 733–744. IEEE, 2017.

[63] Kerim Yasin Oktay, Sharad Mehrotra, Vaibhav Khadilkar, and Murat Kantarcioglu. Semrod: secure and efficient mapreduce over hybrid clouds. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 153–166, 2015.

[64] Yifan Ou and Reza Samavi. Mixed strategy game model against data poisoning attacks. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 39–43. IEEE, 2019.

[65] Saibal K Pal, Puneet Sardana, and Ankita Sardana. Efficient search on encrypted data using bloom filter. In *2014 International Conference on Comput-*

ing for Sustainable Global Development (INDIACom), pages 412–416. IEEE, 2014.

[66] Kihong Park and Heejo Lee. On the effectiveness of probabilistic packet marking for ip traceback under denial of service attack. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, volume 1, pages 338–347. IEEE, 2001.

[67] Kihong Park and Heejo Lee. On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets. *ACM SIGCOMM computer communication review*, 31(4):15–26, 2001.

[68] Yanqing Peng, Jinwei Guo, Feifei Li, Weining Qian, and Aoying Zhou. Persistent bloom filter: Membership testing for the entire history. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1037–1052, 2018.

[69] The pick-up time in a day extracted from 2018 January New York Taxi data. Taxi dataset. `https://www.kaggle.com/code/wti200/exploratory-analysis-nyc-taxi-trip`, 2018.

[70] Supranamaya Ranjan, Ram Swaminathan, Mustafa Uysal, and Edward W Knightly. Ddos-resilient scheduling to counter application layer attacks under imperfect detection. In *INFOCOM*, pages 1–14. Citeseer, 2006.

[71] Supranamaya Ranjan, Ram Swaminathan, Mustafa Uysal, Antonio Nucci, and Edward Knightly. Ddos-shield: Ddos-resilient scheduling to counter application layer attacks. *IEEE/ACM Transactions on networking*, 17(1):26–39, 2008.

[72] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A McCann, and S Yu Philip. Lopub: high-dimensional crowdsourced data publication

with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 13(9):2151–2166, 2018.

[73] Dazhong Rong, Shuai Ye, Ruoyan Zhao, Hon Ning Yuen, Jianhai Chen, and Qinming He. Fedrecattack: model poisoning attack to federated recommendation. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 2643–2655. IEEE, 2022.

[74] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for ip traceback. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 295–306, 2000.

[75] Junshuai Song, Zhao Li, Zehong Hu, Yucheng Wu, Zhenpeng Li, Jian Li, and Jun Gao. Poisonrec: an adaptive data poisoning framework for attacking black-box recommender systems. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 157–168. IEEE, 2020.

[76] Mudhakar Srivatsa, Arun Iyengar, Jian Yin, and Ling Liu. Mitigating application-level denial of service attacks on web servers: A client-transparent approach. *ACM Transactions on the Web (TWEB)*, 2(3):1–49, 2008.

[77] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30, 2017.

[78] Xinyue Sun, Qingqing Ye, Haibo Hu, Jiawei Duan, Tianyu Wo, Jie Xu, and Renyu Yang. Ldprecover: Recovering frequencies from poisoning attacks against local differential privacy, 2024.

[79] UCI. Uci datasets. `http://archive.ics.uci.edu/datasets`, 2023.

[80] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. Captcha: Using hard ai problems for security. In *Advances in Cryptology—EUROCRYPT*

2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22, pages 294–311. Springer, 2003.

[81] Jiacong Wang, Mingzhong Xiao, and Yafei Dai. Mbf: A real matrix bloom filter representation method on dynamic set. In *2007 IFIP International Conference on Network and Parallel Computing Workshops (NPC 2007)*, pages 733–736. IEEE, 2007.

[82] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local differential privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 638–649. IEEE, 2019.

[83] Zhu Wang, Tiejian Luo, Guandong Xu, and Xiang Wang. A new indexing technique for supporting by-attribute membership query of multidimensional data. In *International Conference on Web-Age Information Management*, pages 266–277. Springer, 2013.

[84] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[85] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. Fedattack: Effective and covert poisoning attack on federated recommendation via hard sampling. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4164–4172, 2022.

[86] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *ECAI 2012*, pages 870–875. IOS Press, 2012.

[87] Sisi Xiong, Yanjun Yao, Qing Cao, and Tian He. kbf: a bloom filter for key-value storage with an application on approximate state machines. In *IEEE INFOCOM*

2014-IEEE Conference on Computer Communications, pages 1150–1158. IEEE, 2014.

[88] Cheng Xu, Qian Chen, Haibo Hu, Jianliang Xu, and Xiaojun Hei. Authenticating aggregate queries over set-valued data with confidentiality. *IEEE Transactions on Knowledge and Data Engineering*, 30(4):630–644, 2017.

[89] Cheng Xu, Jianliang Xu, Haibo Hu, and Man Ho Au. When query authentication meets fine-grained access control: A zero-knowledge approach. In *Proceedings of the 2018 International Conference on Management of Data*, pages 147–162, 2018.

[90] Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. Lf-gdpr: A framework for estimating graph metrics with local differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 34(10):4905–4920, 2020.

[91] Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. Towards locally differentially private generic graph metric estimation. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1922–1925. IEEE, 2020.

[92] Qingqing Ye, Haibo Hu, Ninghui Li, Xiaofeng Meng, Huadi Zheng, and Haotian Yan. Beyond value perturbation: Local differential privacy in the temporal setting. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[93] Chin-Yuan Yeh, Hsi-Wen Chen, De-Nian Yang, Wang-Chien Lee, S Yu Philip, and Ming-Syan Chen. Planning data poisoning attacks on heterogeneous recommender systems in a multiplayer setting. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 2510–2523. IEEE, 2023.

[94] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.

[95] Minlan Yu, Alex Fabrikant, and Jennifer Rexford. Buffalo: Bloom filter forwarding architecture for large organizations. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 313–324, 2009.

[96] Saman Taghavi Zargar, James Joshi, and David Tipper. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE communications surveys & tutorials*, 15(4):2046–2069, 2013.

[97] Kehuan Zhang, Xiaoyong Zhou, Yangyi Chen, XiaoFeng Wang, and Yaoping Ruan. Sedic: privacy-aware data intensive computing on hybrid clouds. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 515–526, 2011.

[98] Rui Zhang and Quanyan Zhu. A game-theoretic defense against data poisoning attacks in distributed support vector machines. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4582–4587. IEEE, 2017.

[99] Xiaojian Zhang and Xiaofeng Meng. Differential privacy in data publication and analysis. *Chinese journal of computers*, 37(4):927–949, 2014.

[100] Yulin Zhu, Yuni Lai, Kaifa Zhao, Xiapu Luo, Mingquan Yuan, Jian Ren, and Kai Zhou. Binarizedattack: Structural poisoning attacks to graph-based anomaly detection. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 14–26. IEEE, 2022.