Differential privacy and game theory in cybersecurity

A thesis submitted in fulfilment of the requirements
for the degree of

Doctor of Philosophy
in
Information Systems, Software Engineering, Analytics

by

Lefeng Zhang

to

School of Computer Science
Faculty of Engineering and Information Technology
University of Technology Sydney
NSW - 2007, Australia

January 2024

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Lefeng Zhang* declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the *School of Computer Science*, *Faculty of Engineering and Information Technology* at the University of Technology Sydney, Australia.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

SIGNATURE: Production Note: Signature removed prior to publication.

DATE: 08th January, 2024

PLACE: Sydney, Australia

Dedication

To all those who support me ...

ACKNOWLEDGMENTS

I acknowledge Professor Tianqing Zhu, Professor Wanlei Zhou, Professor Ping Xiong, Professor Bo Liu, Professor Haibin Zhang, Professor Farookh Khadeer Hussain, and Dr. Dayong Ye, Dr. Heng Xu for their valuable suggestions.

List of Publications

Related to the Thesis :

1. **L. Zhang**, T. Zhu, P. Xiong, W. Zhou, and Philip S. Yu. *More than Privacy: Adopting Differential Privacy in Game theoretic Mechanism Design*. ACM Comput. Surv. 54, 7, 2021.

2. **L. Zhang**, T. Zhu, F. K. Hussain, D. Ye and W. Zhou. *A Game Theoretic Method for Defending Against Advanced Persistent Threats in Cyber Systems*, in IEEE Transactions on Information Forensics and Security, vol.18, pp.1349-1364, 2023.

3. **L. Zhang**, T. Zhu, P. Xiong, W. Zhou and P. S. Yu. *A Robust Game Theoretical Federated Learning Framework with Joint Differential Privacy*, in IEEE Transactions on Knowledge and Data Engineering, vol.35, no.4, pp.3333-3346, 2023.

4. **L. Zhang**, T. Zhu, P. Xiong, W. Zhou and P. S. Yu. *A Game Theoretic Federated Learning Framework for Data Quality Improvement*, in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2022.3230959.

5. **L. Zhang**, T. Zhu, H. Zhang, P Xiong, and W. Zhou. *FedRecovery: Differentially Private Machine Unlearning for Federated Learning Frameworks*, in IEEE Transactions on Information Forensics and Security, vol. 18, pp. 4732-4746, 2023, doi: 10.1109/TIFS.2023.3297905.

Others :

5. **L. Zhang**, G. Song, D. Zhu, W. Ren, P. Xiong. *Location privacy preservation through kernel transformation*, Concurrency and Computation: Practice and Experience. 34. 10.1002/cpe.6014, 2020.

6. S. Zhao, **L. Zhang**, P. Xiong. *PriFace: a privacy preserving face recognition framework under untrusted server*. Journal of Ambient Intelligence and Humanized Computing, 14, pp.2967-2979 (2023).

7. H. Xu, T. Zhu, **L. Zhang**, W. Zhou, and Philip S. Yu. 2023. Machine Unlearning: A Survey. ACM Comput. Surv. 56, 1, Article 9 (January 2024), 36 pages. https://doi.org/10.1145/3603620.

8. X. Hu, Z. Jin, **L. Zhang**, A. Zhou, D. Ye. *Privacy preservation auction in a dynamic social network*. Concurrency Computat Pract Exper. 2022; 34:e6058.

Abstract

The vast majority of cybersecurity solutions are founded on game theory, and differential privacy is emerging as perhaps the most rigorous and widely-adopted privacy paradigm in the field. However, alongside all the advancements made in both fields, there is not a single application that is not still vulnerable to privacy violations, security breaches, or manipulation by adversaries. The current understanding of the interactions between differential privacy and game-theoretic solutions is limited. Hence, this thesis undertook a comprehensive exploration of differential privacy and game theory in the field of cybersecurity, finding that differential privacy has several advantageous properties that can make more of a contribution to game theory than just privacy protection. It can also be used to build heuristic game-theoretic models for cybersecurity solutions, to avert strategic manipulations by adversaries, and to quantify the cost of information leakage. With a focus on cybersecurity, the aim of this thesis is to provide new perspectives on the currently-held impossibilities in privacy and security issues, potential avenues to circumvent those impossibilities, and opportunities to improve the performance of cybersecurity solutions with game-theoretic and differentially private techniques. Specifically, it makes the following contributions:

- In modern federated learning, clients typically seek appropriate remuneration for the utilization of their data and resources. Meanwhile, existing frameworks do not offer sufficient protection against malicious participants attempting to manipulate a trained model with poisonous updates. This thesis proposes a jointly differentially private framework for federated learning that limits the impact of adversarial clients and provides a rigorous privacy guarantee. The proposed framework involves two truthful game-theoretic mechanisms the server can use to select participating clients for training.

- Classic federated learning frameworks assume that all clients want to improve model accuracy, and so participation is voluntary. In reality, clients usually expect compensation for the utilization of their data and resources. Additionally, today's frameworks allow clients to perturb their parameter updates locally, which seriously impacts model accuracy. This thesis presents a private reward game that allows clients in federated learning to dynamically decide the quality of their training data without disclosing the level of quality they provide. The game converges to an approximate Nash equilibrium under a guarantee of joint differential privacy. Furthermore, the model also experiences a reduction in the injected noise.

- Advanced persistent threats pose a significant risk to cyber security today. A critical part of the defender's defense strategy is finding a suitable time to adjust the strategy to ensure attackers learn the least possible information. Another challenge is determining how to make the best use of one's resources to achieve a satisfactory defense level. This thesis develops a rivalry game for advanced persistent threats that considers the information players disclose through their strategy adjustments. The defender and attacker are able to figure out the best timing for strategy adjustments based on the solution of the game.
- Machine unlearning is an emerging paradigm that aims to make machine learning models "forget" what they have learned about particular data. Nevertheless, the unlearning issue for federated learning has not been completely addressed due to its special working mode. Existing solutions heavily depend on retraining-based model calibration and the convex assumption of loss functions. This thesis quantifies the incremental effect in federated learning and proposes an efficient unlearning algorithm that reproduces a model that is statistically indistinguishable from the retrained one by only exploring clients' historical submissions.

Table of Contents

List of Figures

Figure **Page**

List of Tables

CHAPTER 1

INTRODUCTION

Cybersecurity is becoming increasingly important in our daily lives as a result of the rapid development of artificial intelligence (AI) techniques and cybersystems. Modern machine learning and data engineering methods are able to deliver comprehensive analytical results for various fields such as healthcare, automobiles, etc., making human activities more convenient but also posing new threats to cybersystems. The threats include privacy leakage, data breaches, damages to models, etc., which result in significant losses for governments, organizations, and commercial entities. Cybersecurity refers to the practice of preserving computer systems, networks, and digital data against unauthorized access, stealthy attacks, larceny, and any other cyber threats. It entails the implementation of numerous precautions, technologies, and procedures to protect computers, servers, mobile devices, electronic systems, and the data they contain.

Privacy preserving is one of the most significant issues in cybersecurity. Privacy leakage may happen in various intentional or unintentional ways, including data collection, model training, and system upgrading [28, 80]. Due to the increased dependence of modern data on human activities, adversaries can even deduce personal information from a training dataset or a published model. For example, general information about a patient can be extrapolated from a medical model if the adversary is aware of the target's identity. As a result, it is necessary to explore and develop new privacy-preserving techniques to defend against potential privacy threats.

A popular approach to enhancing the privacy and security of a cybersystem is to investigate the interactions between each party. Basically, game theory is the study

of mathematical models of conflict and cooperation between intelligent, rational, and irrational decision-makers [99]. It enables the modeling of participants' behaviors, the analysis of attacker and defender strategies, and the prediction of the tranquil status of a system. A significant amount of privacy and security solutions are founded on game theory. Incorporating game theory into cybersecurity is instrumental in designing new privacy preserving mechanisms and analyzing sophisticated interactions.

Differential privacy [32, 90] is a rigorous mathematical formulation of privacy. It is initially proposed to prevent the information leakage of individuals when their data are used in publishing, computation, and analysis [173]. In fact, differential privacy provides more desirable properties when applied to the design of game-theoretic mechanisms [30, 86]. In the following, I will introduce more detailed information in three lines: privacy and security issues in cybersecurity, differential privacy in cybersecurity, and game theory in cybersecurity.

1.1 Privacy and security issues in cybersecurity

Since the proliferation of AI techniques, privacy concerns have once again captured the public's attention. Modern recommendation algorithms can easily infer a user's hobbies and interests by simply analyzing his or her purchasing and browsing history, while face-swap techniques can readily change a user's appearance to that of any celebrity they follow on social media. Since AI-driven methods have continually been a prominent and influential trend in our society, privacy concerns cannot be overlooked.

In cybersecurity, privacy issues arise when personal or sensitive information is not sufficiently protected, leading to unauthorized access, misuse, or disclosure [79]. Typical privacy issues include data breaches, unauthorized data collection, geo-location tracking, and other similar infringements on privacy. Fortunately, many efforts have been made with a focus on reducing privacy risks in cybersecurity [115, 116, 150, 162]. Essentially, addressing privacy issues in cybersecurity requires the implementation of effective data protection methods, adherence to privacy laws and regulations, and the development of robust privacy policies. In addition, individuals should be empowered to take control over their personal information, like under the General Data Protection Regulation (GDPR) [140] in the European Union.

In comparison, the security issues are more comprehensive, various security threats can pose risks to cybersystems [82, 126]. A cyberattack is an attack launched by cybercriminals using one or multiple computers against computers, networks and organiza-

tions. Cyberattacks have the potential to steal data, intentionally disable and disrupt systems, or utilize a compromised computer as a base for more attacks. Common security threats include malware, phishing, spoofing, and Denial-of-Service attacks, etc [68].

To address security issues, it is necessary to implement a comprehensive set of defense strategies [112], such as security controls, intrusion detection and intrusion prevention systems (IDS and IPS), encryption, network segmentation, etc. Cybersecurity is an ongoing process that necessitates constant vigilance, frequent threat assessments, and the ability to adapt to ever-changing threats [3]. Therefore, continuous surveillance, threat intelligence, and proactive security measures are crucial for mitigating risks and defending against evolving threats.

1.2 Differential privacy in cybersecurity

Differential privacy [32] is a rigorous mathematical definition of privacy that requires any individual in a dataset has negligible effect on the aggregated output. Figure. 1.1 illustrates the underlying principle of differential privacy. Suppose we have two datasets D and D' that are identical except for only one record. A query function f works on the two datasets and produces aggregated output $f(D)$ and $f(D')$. A mechanism is claimed to satisfy differential privacy if, for every possible output s, the probabilities where it was generated (namely, $\Pr[s = f(D)]$ and $\Pr[s = f(D')]$) are almost identical. In other words, an adversary cannot determine whether an individual is in or not in a dataset by analyzing the query output. As a result, even if the adversary obtains enough background knowledge about other records in a database, he or she still cannot infer information about the targeted one.

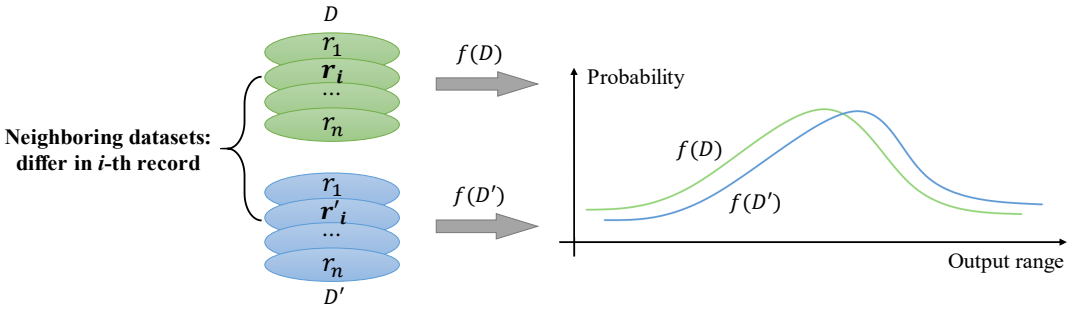


Figure 1.1: Illustration of differential privacy

Differential privacy is not only a theoretical privacy model. In the past decade, differential privacy has been widely employed in the area of artificial intelligence [173]

and cybersecurity [20]. In addition, it also attracts great interest of technology companies, such as Apple and Google [38]. Specifically, differentially private data publishing aims to release aggregated statistical information without revealing any specific data records. In comparison, private learning and analysis seek to integrate differentially private mechanisms into non-private algorithms and develop new frameworks with respect to learning tasks. For example, Hao et al. [51] developed a privacy-preserving federated deep learning scheme for the gradient descent method. They combined differential privacy with homomorphic encryption in the parameter updating process. Each local gradient is randomized by adding Laplace noise and then encrypted using the Pallier algorithm. The integration of encryption techniques successfully avoids collusion between the server and users. Hu et al. [54] presented a differentially private learning framework to train personalized models for users distributed in the Internet of Things (IoT). They formulated the learning task as a multitask optimization problem and employed the Gaussian mechanism to protect the information in dual variables. The trained model takes advantage of each user's data and maintains personalized characteristics.

1.3 Game theory in cybersecurity

Game theory is the study of mathematical models of conflict and cooperation between intelligent rational and irrational decision-makers [99]. Game-theoretic solution has been widely explored across almost every field of artificial intelligence. For example, Nash equilibrium [101] can be employed to analyze the strategy of agents in distributed control systems [62], evolutionary game is an efficient approach to understand the competition of vehicular nodes in cognitive vehicular networks [132].

Cybersecurity involves strategic interactions between defenders and attackers. However, most conventional security problems depend on heuristic solutions, such as firewalls, intrusion detection, and anti-virus programs [30]. Fortunately, game theory provides technical insight into security decisions in a methodical way. It answers the question of how each party will respond to the strategy of their opponent. Game-theoretic models can be used to describe and analyze how each player adjusts and changes their strategy. For example, Vakilinia et al. [137] explored cyber-information sharing against sophisticated cyber-attacks. They modeled the information sharing problem as a coalition game where organizations get rewards for sharing their vulnerability information. The game is modeled in a prisoner's dilemma manner – the organizations that refuse to reciprocate receive a better payoff than the participating ones. The reward and participation fee

are balanced to incentivize organizations to move from their Nash equilibrium strategy to mutual sharing. The actions of the defender and the attacker have a significant impact on each other's strategies. Consequently, both of their actions influence the effectiveness of a defense mechanism. Ye et al. [158] proposed a cyber deception game to mislead attackers by hiding or providing inaccurate system information. In that game, the defender strategically changes and obfuscates the information of vulnerable systems, while the attacker employs Bayesian inference to infer the hidden information. The defender decides whether to deploy a honeypot or disconnect some systems according to the perturbed system number, and the attacker uses ϵ-greedy strategy to select the target configurations that maximize the expected attacking gain. The proposed game reduces the influence of system number on network security and limits the impact of inference attacks from rational attackers.

1.4 Modern strategic privacy and security issues

The rapid growth of modern computing power and advanced reasoning techniques enables each participant to comprehensively balance their benefit and loss before making decisions in cybersecurity. As participants become more rational, more sophisticated approaches are required to address contemporary privacy and security issues.

Federated learning [48] is a typical machine learning paradigm that was initially proposed to train a model with the collaboration of multiple data holders (a.k.a. clients). This training mode enables the development of a machine learning model without disclosing the original data of each client. However, as clients' are rational and seek to maximize their own profit, some participants may deliberately skew the training process by injecting carefully crafted poisonous updates, incorporating low-quality data records, etc. These abnormal behaviors will contaminate the trained model and lead to significant losses for other participants. Therefore, strategic methods should be proposed to defend against adversaries as well as encourage the use of high-quality data. In addition, recent laws and regulations mandate the withdrawal of personal data. After the model has been entirely trained, it is possible for a previous participant to recall giving permission to use their data. Consequently, modern federated learning frameworks also require the removal of the contribution and influence of a specific participant.

Modern privacy and security concerns are not unique to federated learning. Every second, cybersystems confront thousands of different infiltration and intrusion attempts. The attackers are well-founded, highly determined, and equipped with the newest

attack methods, posing long-term advanced persistent threats (APTs). The attackers continuously update their attack methods based on the victim's information. To defend against APTs, strategic countermeasures should be proposed to limit the attacker's rapid evolution. As a result, game theory is an effective method for analyzing the behavior of attackers and developing an optimal defense strategy.

1.5 Research objectives and challenges

This section presents the research objectives of the thesis as well as the related research challenges. The purpose of this thesis is to investigate differentially private and game-theoretic approaches to address privacy and security issues in cybersecurity. Detailed objectives and challenges are enumerated below.

- **Objective 1: Enhancing robustness against adversaries in federated learning.** The first research objective is to improve the robustness of federated learning framework. Researchers have demonstrated that federated learning is vulnerable to manipulation by adversarial clients. As clients' raw data are never shared in federated learning, the server cannot verify the data used in each local training. Through carefully crafted updates, adversarial clients are able to poison the collaboratively learned model. Therefore, framework designs and defense strategies must be made more robust and reliable.

  Challenges: To achieve this goal, two challenges must be addressed. The first is how to incentivize and select clients to participate in federated learning. Second, how to limit the influence of adversarial clients while ensuring a rigorous privacy guarantee for the others.

- **Objective 2: Improving the quality of training data in federated learning.** Nowadays, data, especially high-quality data, is regarded as a valuable and private asset. The traditional assumption that all clients will voluntarily contribute their data to federated learning may be overoptimistic. Furthermore, the clients involved in a training task lack motivation to use their best data because the quality of the data is generally not recognized as their contribution. However, the quality of training data essentially impacts the performance of the global model. From this perspective, an incentive mechanism that encourages clients to use their high-quality data during training is an indispensable element of a modern federated

learning system. In the meantime, the incentive framework should also lead to acceptable model accuracy.

Challenges: Unfortunately, data quality information is also considered as clients' privacy. The challenges are twofold. First, how to design a private reward game that enables clients to dynamically determine the quality of their training data without disclosing the quality level they provided. Second, the game must converge to an approximate Nash equilibrium under a rigorous privacy guarantee.

- **Objective 3: Defending against Advanced Persistent Threats (APTs).** In APT campaigns, the attacker and defender continuously collect information about their opponent, learning from their opponent's strategy adjustments and upgrading their own strategies. Intriguingly, despite greater efforts to defend against APTs in recent times, frequent upgrades in defense strategies are not leading to increased security and protection. From this standpoint, it is important for the defender to choose a time to make their strategy adjustment when they can ensure the least amount of information will be learned by the attacker.

  Challenges: It is challenging to characterize and quantify the information leakage incurred during the attacker's and defender's strategy adjustments. In addition, determining the optimal defense levels and finding the appropriate defense resource allocations is another challenge.

- **Objective 4: Machine unlearning for clients in federated learning.** Recent legislation requires that the private information about a user should be removed from a database as well as machine learning models upon certain deletion requests. While erasing data records from memory storage is straightforward, removing the influence of particular data samples from a trained model has not been fully explored. As a result, it is worthwhile to investigate an efficient method to achieve machine unlearning for clients in federated learning.

  Challenges: As the conventional convex assumption cannot be applied to deep neural networks, the challenges lie in the mathematical quantification of each client's influence in federated learning, the design of learning and unlearning approaches, and theoretical proofs and analyses of the unlearning algorithm.

1.6 Thesis outline

This section aims to build the structure of the thesis. In terms of three key topics: differential privacy, game theory, and cybersecurity. Starting with federated learning, this thesis seeks to provide a jointly differentially private framework that is robust against clients' malicious manipulation. Subsequently, this thesis develops a reward game that encourages clients to contribute their high-quality data for the model in federated learning, achieving the privacy-utility tradeoff in the process. To defend against APT attacks, this thesis presents an APT rivalry game to model the information leakage by considering the fast evolution of attack techniques and the strong learning ability of attackers. With a focus on complying with data protection laws and regulations, this thesis proposes a differential privacy-based machine unlearning method to remove the influence of a participant in federated learning.

The detailed content of each chapter is organized as follows:

- Chapter 2 provides a literature review on differential privacy, game theory, federated learning, advanced persistent threats, and machine unlearning, including notations, pertinent concepts, and emerging techniques in these fields.

- Chapter 3 studies a game-theoretic framework for federated learning, aiming at guiding client selection and limiting the influence of adversaries. This chapter proposes a jointly differentially private federated learning framework that limits the impact of adversarial clients and provides a rigorous privacy guarantee. The proposed framework involves two truthful game-theoretic mechanisms the server can use to select participating clients for training. The client selection mechanisms satisfy game-theoretic properties and are robust against strategic manipulation by clients.

- Chapter 4 addresses the privacy and data quality issues of federated learning. This chapter develops the idea of "money for privacy" to improve the performance of the global model. In addition, this chapter also introduces a private reward game that allows clients to dynamically decide the quality of their training data without disclosing the level of quality they provided. The game converges to an approximate Nash equilibrium under a guarantee of joint differential privacy and satisfies desirable game-theoretic properties.

- Chapter 5 focuses on defending against APT attacks on cybersystems. This chapter develops an APT rivalry game that considers the information disclosed through a

player's strategy adjustments. The defender and attacker are able to figure out the best timing for strategy adjustments based on the solution of the game. Besides, we design two learning mechanisms based on reinforcement learning to help defenders find optimal defense levels as well as appropriate resource allocations during a game against attackers.

- Chapter 6 investigates how to unlearn a client's influence from a trained machine learning model. This chapter mathematically quantifies each client's influence in federated learning and proposes an efficient unlearning algorithm to reproduce a model that is indistinguishable from the retrained one by only exploring clients' historical submissions. The proposed algorithm achieves a rigorous unlearning guarantee by adding carefully calibrated noises.
- Chapter 7 concludes this thesis and discusses future research directions for the development of differential privacy and game theory in cybersecurity.

CHAPTER 2

BACKGROUND

2.1 Notations

Differential privacy (DP) concerns a database D with n unordered records. A dataset D' is defined as a neighboring dataset of D if it only differs by one record (i.e., $|D \Delta D'| \leq 1$). A query $f : D \to R$ on dataset D is an arbitrary function that maps D to an output range R. The premise of differential privacy is to guarantee that a randomized method of answering f must behave similarly with neighboring datasets.

Game theory concerns n self-interested agents, who endeavor to maximize their utility. Each agent has what is known as a private type t_i from type space T_i. This private type encodes what player i knows but the rest of the world does not know [109]. It could be, for example, the maximum price an auction bidder is willing to pay, or the preference order a voter holds over candidates in an election. A mechanism $\mathscr{M}$ collects the reported types of each agent and maps them to an output space O. The mechanism has two aims: first, to incentivize agents to report their true types, second, to achieve a desirable social outcome. The main notations used throughout this thesis are summarized in Table 2.1.

Table 2.1: Notations

Notations	Explanation	Notations	Explanation
D	Dataset	ϵ	Privacy budget
D'	Neighboring dataset	f	Query or mapping

Continued on next page.

Continued.

Δf	The sensitivity	∇f	The gradient of function f
$\mathscr{M}$	A mechanism	s, o	Mechanism output
$q(\cdot)$	The score function	$u(\cdot)$	The utility function
$c(\cdot)$	The cost function	v_i	Player's valuation
p_i	Player's payment	a_i	Player's action
w, θ	Model's parameters	η	Learning rate
S	The server	C_i	The i-th client
m, n	Counts our numbers	$[n]$	The set $\{1, 2, \ldots, n\}$
r_i	Player's reward	x_{-i}	Vector x except ith value
OPT	The optimal value	$O(\cdot)$	Complexity
β, δ	Confidence parameter	$\mathscr{N}(0, \sigma^2)$	The Gaussian distribution
H	Shannon entropy	f'	The derivative of f
$I(\cdot)$	Mutual information	B	Budget constraint
$\mathscr{G}$	A game	$\mathscr{T}$	Strategy set
$\mathbb{N}$	Natural number	$\mathbb{Z}$	Integer set
$\mathbb{E}[\cdot]$	The expectation	T	Time or game round
$\mathbb{R}$	Real number	$\|\|\cdot\|\|_{1,2}$	Vector norms

2.2 Differential privacy (DP)

This section introduces the basic concepts of differential privacy, including its original definition, mathematical properties, and practical variants.

2.2.1 Differential privacy basics

Differential privacy requires deleting or modifying any record in a dataset results in a negligible effect on the output distribution of a randomized algorithm.

Definition 2.1. *(**Differential privacy** [32]) A randomized method $\mathscr{M}$ gives (ϵ, δ)-differential privacy for any pair of neighboring datasets D and D', and for every possible set of outcome $S \subseteq R$, if $\mathscr{M}$ satisfies:*

$$\Pr[\mathscr{M}(D) \in S] \leq \exp(\epsilon) \cdot \Pr[\mathscr{M}(D') \in S] + \delta, \tag{2.1}$$

where the probability space is over the coin flips of the method $\mathscr{M}$.

The parameter ϵ refers to the privacy budget. If $\delta = 0$, the randomized mechanism $\mathscr{M}$ is ϵ-differentially private, which satisfies the strictest privacy requirement according to the definition.

Sensitivity determines the magnitude of perturbations required for privately answering a query.

Definition 2.2. *(l_1-**sensitivity** [32]) Given a pair of neighboring datasets D and D', the l_1-sensitivity of a query $f : D \to R^n$ is defined as*

$$\Delta f = \max_{D,D'} ||f(D) - f(D')||_1. \tag{2.2}$$

The l_1 sensitivity of a query f captures the maximum difference f may result in when made on D and D'. It indicates how much uncertainty should be introduced to protect the privacy of a single individual.

The Laplace method is used for answering numeric queries. It perturbs the true answer with scaled noise drawn from the Laplace distribution. The scale of the noise is calibrated to $\Delta f/\epsilon$.

Definition 2.3. *(**Laplace Method** [33]) Given any function $f : D \to R^n$ over dataset D, the Laplace method provides ϵ-differential privacy, which is defined as*

$$\mathscr{M}(D) = f(D) + Lap(\frac{\Delta f}{\epsilon})^n. \tag{2.3}$$

The Laplace method is a paradigm for privately answering numeric queries. The accuracy bound of the Laplace method is as follows:

Theorem 2.1. *(**Accuracy of Laplace Method** [33]) Suppose $f : D \to R^n$ over dataset D, and let $\mathscr{M}$ be a Laplace method, then for $\delta \in (0,1]$:*

$$\Pr[||f(D) - \mathscr{M}(D)||_\infty \geq \log(\frac{n}{\delta}) \cdot (\frac{\Delta f}{\epsilon})] \leq \delta, \tag{2.4}$$

where n is the dimension of the output space R.

The Gaussian method leverages Gaussian-distributed noise to perturb the exact query result. It provides a differential privacy guarantee for numeric data. Due to the introduction of the Gaussian noise, the mechanism employs l_2 norm in sensitivity.

Definition 2.4. *(**Gaussian Method** [35]) Given any function $f : D \to R^n$ over dataset D, the Gaussian method, which adds Gaussian distributed noise to each coordinate of the*

output, provides (ϵ, δ)*-differential privacy guarantee. The noise is scaled to* $\mathcal{N}(0, \sigma^2)$ *with the parameter satisfying*

$$\sigma \geq \sqrt{2 \ln \frac{1.25}{\delta}} \cdot \frac{\Delta_2 f}{\epsilon}. \tag{2.5}$$

For non-numeric queries, the exponential method is designed to provide differential privacy guarantee, paired with an application-dependent score function $q(D, r)$.

Definition 2.5. *(**Exponential Method** [90]) Let* $q(D, r)$ *be a score function that measures the quality of output* $r \in R$ *over dataset* D*. The exponential method* $\mathcal{M}$ *satisfies* ϵ*-differential privacy if*

$$\mathcal{M} = \{\text{return } r \text{ with probability} \propto \frac{\epsilon q(D, r)}{2\Delta q}\}. \tag{2.6}$$

The exponential method outputs an element with a high quality score most of the time, and it often provides a strong utility guarantee.

Theorem 2.2. *(**Accuracy of Exponential Method** [37]) Suppose* $\mathcal{M}$ *is an exponential mechanism with score function* q*, define* $OPT_{q,R}(D) = \max_{r \in R} q(D, r)$*, then*

$$\Pr[q(D, r) \leq OPT_{q,R}(D) - \frac{2\Delta q}{\epsilon} \cdot (\log(|R|) + t)] \leq e^{-t}. \tag{2.7}$$

Lastly, count queries are usually answered with the geometric method. Random integer noise is drawn from a two-sided geometric distribution and added to the count query result [43].

Definition 2.6. *(**Geometric method** [43]) For any count query* $f : D \to \mathbb{N}$ *over dataset* D*, and parameter* $\alpha \in [0, 1]$*, the* α*-geometric method provides differential privacy, which is defined as*

$$\mathcal{M}(D) = \begin{cases} f(D), & \text{if } \alpha = 0; \\ f(D) + X, & \text{if } \alpha \in (0, 1); \\ 0, & \text{if } \alpha = 1, \end{cases} \tag{2.8}$$

where X *is a random variable with probability mass function*

$$\Pr[X = x] = \frac{1 - \alpha}{1 + \alpha} \alpha^{|x|}, x \in \mathbb{Z}, \tag{2.9}$$

provides differential privacy.

The parameter α represents the level of privacy guarantee to be provided; larger values mean stronger privacy protection. For count queries in range $N = \{0, 1, \ldots, n\}$, to avoid obviously wrong output, the α-truncated geometric mechanism remaps each negative output to 0 and each output greater than n to n. The truncated mechanism is also differentially private [43].

2.2.2 Properties of differential privacy

Differential privacy is a rigorous formalization of "privacy" with composition properties. Two theorems enable the design and analysis of complex differentially private methods from simpler private building blocks [37].

Theorem 2.3. *(**Parallel Composition** [89]) Given a set of privacy mechanisms $\mathscr{M} = \{\mathscr{M}_1, \ldots, \mathscr{M}_m\}$, if $\mathscr{M}_i$ provides an ϵ_i privacy guarantee on a disjointed subset of the entire dataset, $\mathscr{M}$ will provide $\max\{\epsilon_1, \ldots, \epsilon_m\}$-differential privacy.*

Theorem 2.4. *(**Sequential Composition** [89]) Given a set of privacy mechanisms $\mathscr{M} = \{\mathscr{M}_1, \ldots, \mathscr{M}_m\}$ performed sequentially on a dataset, each $\mathscr{M}_i$ will provide an ϵ_i differential privacy guarantee, and $\mathscr{M}$ will provide $\sum_i \epsilon_i$-differential privacy.*

In the mean time, differential privacy is also immune to post-processing. Applying arbitrary data-independent post mapping f to a differentially private method $\mathscr{M}$ does not influence its privacy guarantee.

Theorem 2.5. *(**Post-processing** [37]) Suppose $\mathscr{M} : D \to R$ is a (ϵ, δ)-differentially private mechanism. Let $f : R \to R'$ be an arbitrary randomized mapping. Then $f \circ \mathscr{M} : D \to R'$ is (ϵ, δ)-differentially private.*

2.2.3 Differential privacy variants

Differential privacy has variant versions with respect to different scenarios. In this section, we discuss variant differential privacy definitions and investigate how they are developed to achieve game-theoretic properties.

Joint differential privacy is a relaxation of classic differential privacy. The intuition is that if agents' private data are computed in a differentially private manner, then each individual has limited influence on the outcome of the mechanism. In other words, in a differentially private mechanism, the outcome is almost independent of any agent's reported type. To circumvent this limitation, Kearns et al. [69] proposed joint differential

privacy. Intuitively, joint differential privacy guarantees that the joint distribution of the outcome given to all other agents $j \neq i$ is insensitive (differentially private) to agent i's reported type.

Definition 2.7. *(**Joint Differential Privacy** [69]) A randomized method $\mathscr{M}$ gives (ϵ, δ)-joint differential privacy for any pair of neighboring datasets D and D', and for every possible subset of outcome $S_{-i} \subseteq R^{n-1}$, if $\mathscr{M}$ satisfies:*

$$\Pr[\mathscr{M}(D)_{-i} \in S_{-i}] \leq \exp(\epsilon) \cdot \Pr[\mathscr{M}(D')_{-i} \in S_{-i}] + \delta, \tag{2.10}$$

where the probability space is over the coin flips of the method $\mathscr{M}$.

The parameter ϵ refers to the privacy budget. If $\delta = 0$, the randomized mechanism $\mathscr{M}$ is ϵ-joint differentially private.

An important connection between classic differential privacy and joint differential privacy is the billboard lemma [53]. The billboard lemma states that if each agent's output is a function only of an ϵ-differentially private computation and their own private data, then the overall algorithm satisfies ϵ-joint differential privacy.

Lemma 2.1. *(**Billboard Lemma** [53]) Suppose $\mathscr{M} : D \to \mathscr{R}$ is (ϵ, δ)-differentially private. Consider any set of functions $f_i : D_i \times R \to R'$, where D_i is the portion of the database containing i's data. Then the method $\mathscr{M}'$ that outputs to each agent i: $f_i(D_i, \mathscr{M}(D))$ is (ϵ, δ)-jointly differentially private.*

Marginal differential privacy is a further relaxation of differential privacy. Intuitively, marginal differential privacy guarantees that the marginal distribution of the outcome for every agent i is insensitive to another agent j's report.

Definition 2.8. *(**Marginal Differential Privacy** [66]) A randomized method $\mathscr{M}$ gives (ϵ, δ)-marginal differential privacy for any pair of neighboring datasets D and D' that differ in ith data, any $j \neq i$, and for every possible subset of outcome $S \subset R$, if $\mathscr{M}$ satisfies:*

$$\Pr[\mathscr{M}(D)_j \in S] \leq \exp(\epsilon) \cdot \Pr[\mathscr{M}(D')_j \in S] + \delta, \tag{2.11}$$

where the probability space is over the coin flips of the method $\mathscr{M}$.

Marginal differential privacy provides differential privacy guarantee on the granularity of any single agent. It promises that for every pair of agents $i \neq j$, the marginal distribution of the outcome on agent j is differentially private in agent i's data.

Theorem 2.6. *(Kannan et al. [66]) Suppose $\mathcal{M}^1 : T^n \to O$ is (ϵ_1, δ_1)-differentially private, and $\mathcal{M}_j^2 : T^{n-1} \times T \times O \to R$ for $j = 1, 2, \ldots, n$ is (ϵ_2, δ_2)-differentially private in its first argument. Then their composition $\mathcal{M} : T^n \to R^n$, namely*

$$\mathcal{M}(t) = \left(\mathcal{M}_j^2(t_{-j}, t_j, \mathcal{M}^1(t))\right)_{j=1}^n$$

is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-marginally differentially private.

Compared with joint differential privacy, which promises differential privacy on agents $j \neq i$ (i.e., protect privacy against collusions of other $n-1$ agents), marginal differential privacy only guarantees differential privacy on each single agent (i.e., protect privacy against a single agent $j \neq i$).

2.3 Game theory and mechanism design

Game theory is the science of decision-making, behavior prediction, and equilibrium analysis. Mechanism design is one of the important areas of game theory, where the goal is to design rules so that the participants' behavior not only maximizes their own utility but also leads to a desirable social outcome [59]. A typical mechanism design problem involves n rational agents $i \in [n]$, each with their own private type $t_i \in T_i$. A mechanism $\mathcal{M} : T^n \to O$ is a mapping between the reported types and the output space O. There is a utility function $u : T \times O \to [0, 1]$ over the outcome space. Each agent i receives their utility based on their private type and the outcome of the mechanism. There is also a global "utility" function that maps $T^n \times O \to R$, which relates to the social outcomes.

Example 2.1. *(**Single-item auction**) A seller wishes to sell an item to n bidders. Each bidder i has made their own valuation (type) v_i of the item, which represents the maximum amount of money i is willing to pay. Each bidder submits their bid b_i to the seller, which may not equal to v_i. The seller receives a bid vector $b = \{b_1, b_2, \ldots, b_n\}$, and runs a mechanism $\mathcal{M}(b)$ to determine the winner of the auction and, therefore, the selling price p_i. As such, each bidder's utility u_i relies on both their private valuation and the outcome of the mechanism. If i wins, $u_i(v_i, o) = v_i - p_i$. If i loses, then $u_i = 0$. The seller wishes to maximize social welfare $SW = v_i$, s.t i is the winner. In such settings, the goal of the mechanism design is to find a robust $\mathcal{M}$ that maximizes social welfare despite strategic bidding by the agents to maximize their own utility.*

One challenge in mechanism design is to make the mechanism robust to manipulation. As agents are self-interested, they may strategically report their types to maximize their

own utility. If the private type t_i is guaranteed to maximize i's utility, no matter what other agents do, adopting the private type t_i usually becomes the dominant strategy.

Definition 2.9. *(ϵ-**dominant-strategy Truthful** [90]) A mechanism $\mathcal{M} : T^n \to O$ is ϵ-dominant-strategy truthful if, for all agents, all vector of types t_{-i} and $t_i, t_i' \in T$, $\mathcal{M}$ satisfies*

$$u(t_i, \mathcal{M}(t_{-i}, t_i)) \geq u(t_i, \mathcal{M}(t_{-i}, t_i')) - \epsilon. \tag{2.12}$$

That is, no agent can improve more than ϵ utility by mis-representing his or her type.

If $\epsilon = 0$, then mechanism $\mathcal{M}$ is exactly dominant-strategy truthful (or incentive-compatible). If the parameter ϵ tends to 0 as the number of agents grows, then $\mathcal{M}$ is said to be asymptotic truthful.

Example 2.2. *(**First-price auction and second-price auction**) Continuing the single-item auction in Example. 2.1, deciding the winner is straightforward – the bidder with the highest bid wins. In addition, it is intuitive to think that the winner would pay what they bid, which is called the first-price auction. However, the first-price auctions can be hard to reason about. For example, if a bidder bids truthfully (i.e., $b_i = v_i$), their utility will always be 0, so they usually offer a lower their bid. In turn, this makes it hard for the seller to predict the bidders' behaviors. By contrast, in a second-price auction, where the winner pays the second-highest bid in the auction, which is much easier to reason about. Here, each bidder has a utility of $u_i = \max\{0, v_i - \max_{j \neq i} b_j\}$. Therefore, the dominant strategy is for each bidder to bid truthfully at the level of their private valuation. Each bidder then ends the game with nonnegative utility and social welfare is maximized for the seller.*

The above examples illustrate some of the challenges with rule-making in mechanism design. Weak rules cannot guarantee preferred game-theoretic properties and may lead to undesirable results in the face of strategic manipulations. And the vulnerabilities only increase when privacy issues are considered. Fortunately, differential privacy provides new opportunities to bridge the gap between privacy and mechanism design. How its mathematical properties that lend these advantages to game theory translate into different functional roles is discussed in the next section.

2.4 Interaction between DP and game theory

2.4.1 Properties of differential privacy for game theory

Developing a genuinely truthful and robust game-theoretic solution for cybersecurity can be difficult because the designer has to anticipate all possible strategic manipulations and close all possible loopholes. However, differential privacy has four main mathematical properties that naturally guarantee approximate truthfulness and robustness to make the job easier. These are:

- **Privacy**. Privacy protection is obviously the central tenet of differential privacy. Adopting differential privacy in a game protects each participant against information disclosure.
- **Robustness**. Differential privacy limits the influence of every single participant to strategically manipulate a game.
- **Quantification**. Differential privacy has a rigorous mathematic definition that allows a mechanism designer to quantify the influence of each agent, which helps to understand agents' behaviors.
- **Composability**. When several differentially private mechanisms are used in tandem, they still satisfy differential privacy within a process that is strictly controlled by a privacy budget. In game theory, composability makes it possible to construct more sophisticated mechanisms from simple building blocks while still guaranteeing good performance[1].

The example, illustrated in Figure 2.1, demonstrates how the natural mathematical properties of differential privacy lead to robust solutions for game-theoretic problems. Suppose a merchant goes to a village every day to sell a specific kind of good. The merchant pre-sets a reserve price, and the final price is determined by an ascending auction with the highest bidder getting the goods. Many strategic manipulations of this process are possible. Some of the more common scenarios follow.

- Alice does not need to acquire the goods; she simply wishes to interfere with the bidding process so as to cause disruption or failure. She does this by participating in the auction and deliberately over or underbidding in the rounds.

[1]To avoid ambiguity, in this chapter we use the term method to denote the techniques that satisfy differential privacy (e.g., the Laplace method) and the term mechanism to denote game-theoretic algorithms and procedures in a designed strategy.

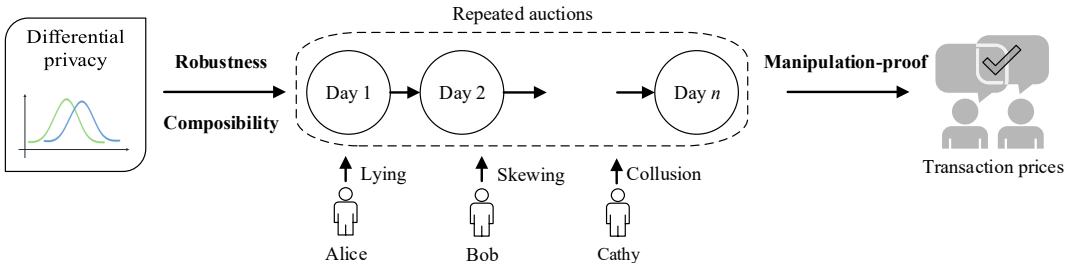


Figure 2.1: Various strategies to manipulate auctions

- Bob believes that competition in later rounds will be much weaker than in the beginning. Therefore, he may consistently underbid for several rounds, hoping to lower early prices and make the wealthier bidders drop out faster [90].
- Cathy holds a similar idea to Bob; however, in order to further lower the price, she colludes with a group of other bidders who all concurrently underbid in the auction.

All of these strategic manipulations carry a negative impact on the merchant's potential revenue. In fact, incentivizing agents to bid truthfully is always the desideratum in an auction. Fortunately, differential privacy carries with it a series of desirable mathematic properties that can ensure the auction mechanism is robust to the above manipulations.

- In the case of Alice, if the auction satisfies differential privacy, the influence of Alice's one (false) record on the final output would be limited, no matter how she changes her bid.
- In Bob's case, sequentially composed differentially private methods still satisfy differential privacy. Even if Bob tries to skew the auction by underbidding, the final price will still be largely unaffected.
- In Cathy's case, differential privacy could easily be extended to situations where the neighboring datasets contain multiple different records. An auction mechanism that satisfies differential privacy guarantees that collusion would not have much impact on the final price.

2.4.2 The roles of differential privacy can play in game theory

Figure 2.2 connects differential privacy and game theory through the different functional roles differential privacy can play in addressing game-theoretic issues. Each of these

seven roles is made possible by the mathematical properties of differential privacy, which often provide new avenues for solving classic problems. Further, each role has two sides – one played by privacy-aware agents, the other played by non-privacy-aware agents. Privacy-aware agents have privacy considerations explicitly formulated within their utility functions. As a result, they can, and frequently do, make tradeoffs between privacy and utility, i.e., information leaked vs. compensation received. Note, however, that non-privacy-aware agents still care about privacy; they are just not equipped with a specific privacy policy to guide their behavior.

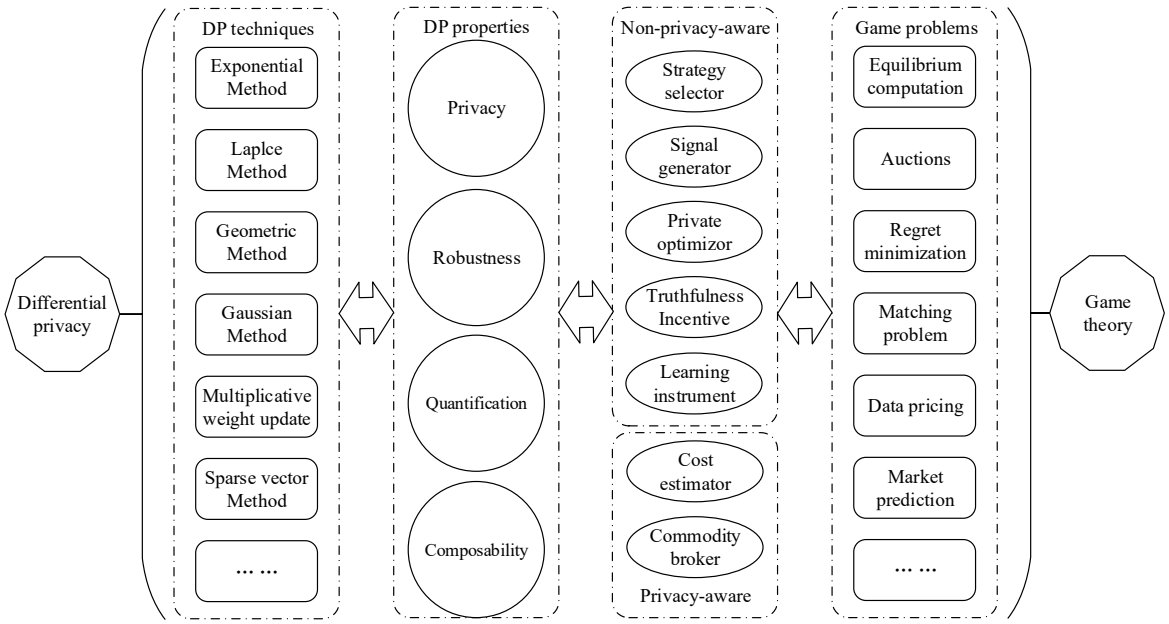


Figure 2.2: Roles and properties that connect game theory and differential privacy

2.4.2.1 A strategy selector

In this role, the mechanism designer uses a differentially private method to choose an outcome or select a strategy while guaranteeing (approximate) truthfulness and a near-optimal social result.

A digital goods auction makes for a good example of differential privacy as a strategy selection tool. In the auction, an auctioneer is selling digital goods to bidders with a demand for one unit each. Each bidder has a private type t_i, which represents the maximum price the bidder will pay. The seller has an unlimited supply of the goods, as the marginal cost of copying digital products is zero. Since there is no prior distribution on bidder types, the seller would like to maximize the revenue Rev with a fixed price p. A

reasonable way for the seller to solve the problem is to apply the exponential method and select a price from the candidate set with a score function Rev. The price p is selected in an ϵ-differentially private manner; thus, the mechanism is ϵ-dominant strategy truthful. Further, using the exponential method also ensures near-optimal revenue [90].

2.4.2.2 A signal generator

The output of differentially private methods in a game can provide informative signals of which strategies the agents have decided to play, creating another role for differential privacy as a signal generator. For instance, consider an allocation problem, where the goods are sensitive, and the bidders' demands are their private information. The allocator wishes to find an allocation that maximizes social welfare. In this problem, an ascending auction with a differentially private counting method [15, 36] can be used to provide near-optimal allocation while giving a sufficient privacy guarantee [29]. Specifically, the mechanism maintains a differentially private counter to record the number of bids received so far. Bidders iteratively estimate the current price according to the signal and decide whether to bid based on their own demands. To protect the bidders' private information, the counter is simply increased by 1 when someone bids, and 0 otherwise.

2.4.2.3 A privacy optimizer

Mechanism design problems are usually coupled with optimization problems. However, agents might be unwilling to provide their private data for computation due to privacy concerns. This is where differential privacy comes in handy. Consider a crowdsensing problem where the administrator wants to buy sensor data from workers for the minimum amount of money. Under the cover of differential privacy, workers can report their charges for performing specific sensing tasks. With this information, the administrator can allocate workers to ensure that all tasks are performed for the minimum cost. This naive worker selection problem is basically a minimum weighted set cover problem, which is NP-hard [138], but it becomes solvable with the exponential method and an iterative worker selection process [64]. Due to the monotonicity of the score function, the mechanism is also guaranteed to be truthful.

2.4.2.4 An incentive to be truthful

Truthfulness is always desirable in game theory, as it is easier to predict agent behaviors when truthful reporting is the dominant strategy. Differential privacy is instrumental in

developing truthful mechanisms. Consider a facility location problem where a city wants to build some schools while minimizing the distance between each citizen and their closest school. Citizens report their preferred locations, and the city chooses locations to maximize social welfare. In this problem, the exponential method and a commitment mechanism can be combined to achieve exact truthfulness [107]. The intuition is quite elegant – the exponential method guarantees good social welfare but is only approximate truthful, while the commitment mechanism is strictly truthful but leads to poor social welfare. Yet combining the two mechanisms with properly assigned weights yields a strictly truthful mechanism with good social welfare.

2.4.2.5 A cost estimator.

In the real world, agents may not only care about the outcome of the mechanism, they may also worry about their privacy when they participate in a mechanism. Therefore, it is worth considering mechanism design for privacy-aware agents and modeling their privacy preferences explicitly in a privacy-aware utility function. The ability to quantify the property of differential privacy is an extremely useful tool when designing new utility functions. Take the quasi-linear utility function $u_i(t_i, \mathcal{M}, o) = u_i(t_i, o) - c_i(t_i, \mathcal{M}, o)$ as an example. Here, $u_i(t_i, o)$ is a traditional utility function and $c_i(t_i, \mathcal{M}, o)$ is the privacy cost of agent i. If an agent's data is used in a differentially private computation, then that privacy cost can be modeled by ϵv_i, where v_i is agent i's private valuation [42]. With these new utility functions and quantified privacy costs, we could figure out the behaviors of privacy-aware agents, and identify the conditions that lead to truth-reporting.

2.4.2.6 A commodity broker

Individuals selling their private data is becoming increasingly common, with some agents willing to share their information as long as it fetches a high enough price. But how much should a collector pay for an individual's data? If the data is used in a differentially private manner, the answer is quite intuitive – the compensation should at least cover an agent's privacy cost, which is bounded by ϵv_i [42]. Moreover, differential privacy also works in private data pricing [108]. Consider a data broker who provides different data-based services for consumers with various interests. If the price of the data with a noise scale of $\epsilon = 0.1$ is \$1, is it reasonable to ask \$10 for data where $\epsilon = 1.0$? The quantification property of differential privacy allows us to explore the pricing strategies while avoiding arbitrage in data markets.

2.4.2.7 Learning instrument

In game theory, learning refers to the dynamic adjustments agents make to their strategies after observing the outcome of a game. It is a common process in games where agents must repeatedly make decisions but have limited knowledge about the game. In such scenarios, it is crucial to avoid a dictatorship by limiting the influence of each agent [72]. For example, in repeated auctions, the effect of an agent's current bid on future prices should be bounded; otherwise, non-myopic agents would bid strategically to cumulatively maximize their utility [57]. The robustness property of differential privacy restricts the influence of each agent, making it a natural choice when constructing game-theoretic learning algorithms. In addition, differential privacy also provides a solid guarantee over the privacy of training samples during the learning process.

2.5 Federated learning

2.5.1 Federated learning basics

Federated learning harnesses the power of a large number of distributed clients to collaboratively train a global model without revealing their local data to a central server [161]. In federated learning, each client C_i possesses a private database, denoted as D_i, and is equipped with a computing device. The server S wishes to train a global model with training data distributed across these clients. The clients who participate in the training are responsible for finding the parameter w that minimizes a given loss function. The loss function of client C_i with database D_i is

$$f_i(w) = \frac{1}{|D_i|} \sum_{j \in D_i} l_j(w), \tag{2.13}$$

where $l_j(w)$ is the loss incurred on the data sample j.

In each training round, the server obtains an aggregated model parameter by computing the weighted average over w_i from each client.

$$w_{avg} = \sum_{i=1}^{n} \alpha_i w_i, \quad \alpha_i = \frac{|D_i|}{\sum_{i=1}^{n} |D_i|}. \tag{2.14}$$

The server broadcasts the aggregated model parameters to each client, who updates their local model and starts the next round of training. The learning process can be formulated as an optimization problem, aimed at finding the parameters that minimizes of the global loss function

$$w^* = \arg\min_{w} \alpha_i f_i(w). \tag{2.15}$$

After enough rounds of local training and parameter aggregation, the solution of the optimization problem will converge to the optimal value of the global model.

2.5.2 Related work – privacy and games in federated learning

Although federated learning protects data privacy to an extent, clients' information can still be inferred from the parameter updates [124]. Differential privacy, as a rigorous privacy model, has been widely used to address this issue.

Wei et al. [147] used a Gaussian mechanism to perturb the actual parameters client-side before sending them to the server for aggregation. They calibrated the appropriate levels of noise for uploading and broadcasting, and theoretically analyzed the convergence bound with respect to the level of privacy protection offered. The strategy is robust to adversaries who eavesdrop between the server and clients.

Hu et al. [54] presented a differentially private federated learning framework to train personalized models for users in distributed IoT architectures. They formulated federated learning as a multitask optimization problem, using a Gaussian mechanism to protect the information in dual variables. The trained model takes advantage of each user's data so as to maintain a high level of personalization.

Hao et al. [51] developed a privacy-preserving deep federated learning scheme based on gradient descent. To protect the parameter update process, they combined differential privacy with homomorphic encryption. Each local gradient is randomized by adding Laplace noise and then encrypted using a Pallier algorithm. Integrating encryption techniques stops collusion between the server and users.

Zhao et al. [169] explored the notion that unreliable participants who hold low-quality data and may negatively impact learning accuracy. Hence, they employed an exponential mechanism to select high-quality participants, and applied functional mechanism expand the loss function. To protect privacy, Laplace noise is added to the coefficients of the approximated polynomial formula.

Girgis et al. [45] introduced a shuffled differential privacy model for federated learning and developed a communication-efficient and local differentially-private stochastic gradient descent algorithm. In their framework, an additional secure shuffler sits between the clients and an untrusted server. In each iteration, the shuffler collects gradients from a subset of randomly-selected clients. Each client's gradient is protected by a compression mechanism that is both differentially-private and communication-efficient.

Federated learning involves the collaboration of different clients, who may have various considerations when they participate in the training, game theory is a useful

tool for analyzing their behavior.

Kang et al. [65] proposed a reputation-based client selection scheme for reliable federated learning. They adopted a consortium blockchain to manage the reputation of each client, which is updated according to their historical behaviors. Different contracts, i.e., resource-reward bundles, are designed for different types of clients, guaranteeing individual rationality and incentive compatibility. Clients with high-quality data receive higher rewards.

Zhan et al. [163] used a Stackelberg game to model the interaction between the server and clients for federated learning. The server tries to minimize the reward it pays while each edge node aims to maximize their own revenue. An incentive mechanism based on reinforcement learning solves the game, where the server acts as a leader and the edge nodes learn the Nash equilibrium after observing the server's decision. In this way, the server and edge nodes can dynamically adjust their optimal strategies.

Tang et al. [128] proposed an incentive mechanism for a federated learning framework to motivate efficient cooperation between organizations. They developed a perfect information game where organizations try to maximize their payoffs by specifying the expected workload and compensation. The payoff is computed as a function of the global model's accuracy and the organization's computation cost. Nash equilibrium is reached when no organization has incentive to deviate from the server's arrangements.

2.6 Advanced persistent threats (APTs)

2.6.1 Introduction to APTs

Advanced persistent threats are a major threat to today's cyber systems. Different from a regular cyber attack, APTs are often performed by groups of adversaries with sophisticated levels of expertise and significant resources. The characteristics of APTs in terms of their definitions follow.

- *Advanced*: The APT attackers are usually well-funded by organizations or even governments, which allows them to tailor advanced tools and methods to generate opportunities to achieve their specific objective. A distinguishing character of APTs is the novelty of the attack methods, which usually include but are not limited to combinations of multiple different attack vectors, such as phishing, malware, and viruses. Some attack techniques may never have even been seen before, which guarantees its success.

- *Persistent*: The goal of an APT attack not only involves stealing sensitive data and undermining critical components of the targeted victim. One important aim of an APT attack is to stay undetected and position oneself for future post-exfiltration or post-impediment. Unlike traditional attacks, which often employ "smash and grab" style tactics for mere financial gain, APT attackers are much more patient and persistent. They usually follow a "low and slow" strategy, gradually expanding their foothold across the whole network.

- *Threat*: APT attacks are accurately aimed, well designed and can cause irreparable damage to the targeted victim. Even worse, they are long-term threats and hard to detect or eliminate. These have raised serious threats to companies, organizations and even nation entities who want to keep their data and secrets secure, causing significant financial and security losses across many fields.

The lifecycle of a typical APT attack consists of five stages [3]: reconnaissance, establishment of foothold, lateral movement, exfiltration, and post-exfiltration. Through continuous reconnaissance and deep exploration of the target's defense strategy, an APT attacker is very likely to avoid being detected [156]. Therefore, to defend against an APT attack, appropriate defense approaches need to be implemented at each stage so as to detect or prevent the movement of attackers. A robust defense strategy should guarantee that attackers cannot evade all the defense layers. Therefore, any defense strategy has to consider multiple points of vulnerability across multiple levels of the network.

In fact, attack and defense in the context of APT is a long and evolutionary process, where the attacker and defender endeavor to explore their opponent's strategies and maximize their strategic benefits. The National Institute of Standards and Technology (NIST) [111] has demonstrated that APT attackers may keep themselves updated with changes in the targeted network, continuously collecting and analyzing useful information about the victim organization while staying undetected. Therefore, it is crucial to consider potential risks and estimate the influence of undetected behaviors when modeling interactions between the attacker and defender.

2.6.2 Related work – defending against APTs

Since APTs involve dynamic strategy adjustment of attackers and defenders, game theory is a useful tool for analyzing their behavior. In fact, many efforts have been devoted to study game theory for APT attacks [55]. In this section, we review the state-of-the-art literature.

2.6.2.1 Cyber deception games

Cyber deception is a good way to deceive and mislead attackers in APTs. Wan et al. [142] developed a cyber deception game based on hypergame theory to mislead attacker's decision-making. They extracted subgames from different stages of cyber kill chain and modeled players' perceived uncertainty which can influence their understandings about the game. The proposed hypergames allow players view and analyze the original game differently according to asymmetric information. Therefore, the defender is able to create defensive deception strategies to manipulate an attacker's belief, which is influential to the attacker's decision-making.

Ye et al. [158] proposed a dynamic security game to mislead attackers by hiding or providing inaccurate system information. In their game, the defender decides whether to deploy a honeypot or disconnect some systems, the attacker tries to select target configurations that maximize its expected attacking gain. They adopted differentially private techniques to perturb the accurate number of systems and obfuscate system configuration. Through this way, the influence of changing system number and system configuration can be limited, and thus the cyber system is more resistant against attackers' reasoning.

Tian et al. [133] formulated a honeypot game to defend against APT attack for industrial Internet of Things. They considered bounded rationality of players and employed prospect theory to describe the rational behavior. In the proposed game, attacker and defender strategically choose periods of attack and defense to maximize their expected utility. Prelec's probability weighting function is introduced to players' utility expressions when they have incomplete information about their opponent. Evolutionary stable strategy of the game is discovered by checking the eigenvalues of utility's Jacobi matrix.

2.6.2.2 FlipIt games

FilpIt game is a useful model to model the interactions in APTs as the attacker and defender takeover the system alternatively [30]. Zhang et al. [168] used the FlipIt game to capture the strategic interactions between APT attacker and defender. They constructed local FlipIt game in distributed nodes to model the occupation of devices for single defender and single attacker, and developed a global FlipIt game over the network to describe the complex interactions among multiple defenders and multiple attackers. Besides, they employed cyber insurance to mitigate the risks for a cyber system, and explored the design of incentive compatible insurance contracts by solving the welfare

maximization problem.

2.6.2.3 Resource allocation games

Yang et al. [156] considered the repair of systems when an APT attack is detected in an organization. They formulated an APT repair game to find effective resource allocation strategies and thus can mitigate the potential loss of the victim organization. Their game falls into the differential Nash game category where the attacker tries to maximize its potential benefit and the defender aims to minimize the potential loss. The potential equilibria are obtained by solving potential systems, and numeric simulations are used to demonstrate the relationship between potential equilibria and the Nash equilibria.

Xiao et al. [153] presented an APT detection game based on cumulative prospect theory. Specifically, the attacker chooses an attack interval to launch an attack, and the defender chooses the scan interval to protect the cyber system. The proposed model incorporates the probability weighting distortion and the framing effect of the subjective attacker and defender. To find the optimal defense strategy, they developed a policy hill-climbing detection scheme to increase the policy uncertainty and thus to mislead the attacker. An experience based value initialization method is proposed to accelerate the learning speed.

2.6.2.4 Information tracking games

Moothedath et al. [95] proposed a dynamic information flow tracking game to monitor the data flow and control flow within a network, and thus to detect APT attacks. They used a graph model to capture the interaction among different nodes in the network, where the attacker aims to reach a specific node and the defender tries to detect the threat in an efficient way. In each stage of the game, the defender decides the tag status of a flow and specifies security policies a flow should follow. The best response of the attacker and defender can be derived by employing a shortest path algorithm.

Sengupta et al. [123] pointed out that there are useful inherent information contained in the multi-stage stealthy movement of APT attackers. They proposed a general-sum Markov game along with a vulnerability scoring method to figure out the utility of players in each stage of the game. They developed a system attack graph to represent the states of the game. The attacker's actions are modeled based on real-world APT attacks, while the defender's actions are allocations of security resources. The game is solved by dynamic programming and the authors showed that the Stackelberg equilibrium with some assumptions is in fact the Nash equilibrium of the general-sum Markov game.

2.6.2.5 Bayesian games

Huang et al. [55] proposed a multi-stage Bayesian game to capture the incomplete information resulted from attacker's deceptive actions and multi-phase movement. They developed a conjugate based method to incorporate Bayesian belief update and the computation of equilibrium strategy. Their game boils down to backward dynamic programming and is solved under the assumption of beta-binomial conjugate prior on users' type. With their framework, the defender is able to compute the perfect Nash equilibrium and thus predict the attacker's behaviors. However, although the proposed method leads to computationally tractable equilibrium, their assumptions may be too strong in real-world APT scenarios.

2.7 Machine unlearning

2.7.1 Machine unlearning basics

Machine unlearning is the study of removing the influence of certain data points from a learned model [47]. The demand for machine unlearning generally stems from real-world situations where data owners need to revoke their contributions to a learned model due to privacy or security issues [14]. In addition, it is legally required that individuals have the right to decide if their personal data can be used by any entity for any purpose. For example, the GDPR and the *Right to be Forgotten* grant the right for users to withdraw consent to the use of their data from companies and organizations under certain circumstances. Therefore, it is important to explore how to fulfill these legal aims in machine learning area.

Machine unlearning deals with both homogeneous (e.g., to unlearn a set of samples) and heterogeneous (e.g., to unlearn a specific class) unlearning requests. Intuitively, the naive way to unlearn targeted data from a model is simply to retrain the model from scratch using the remaining data. However, retraining from scratch results in prohibitively expensive computation costs, and is incapable of handling situations where unlearning requests come frequently. In addition, in some scenarios, such as federated learning, it is unable to perform retraining as the previous participating clients may have lost connection due to the inherently distinct learning mode.

The relationship between model training and machine unlearning is described in Figure 2.3, including the training, predicting (aka. inference), and unlearning processes. At the beginning, a learnable model is initialized and trained using data that may be

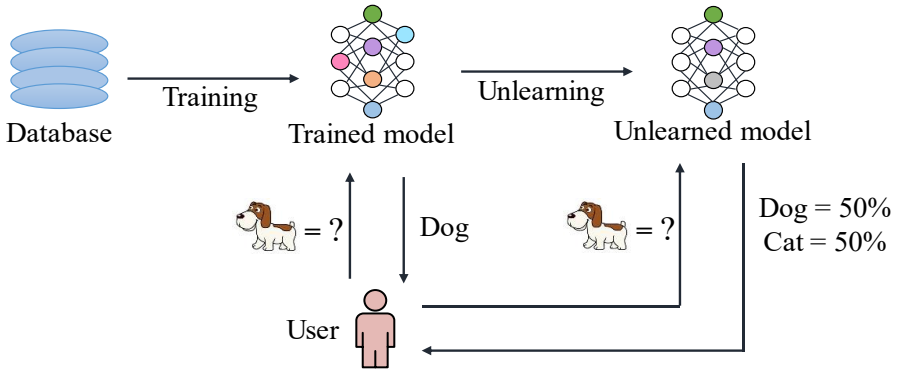


Figure 2.3: Model training, predicting and unlearning processes.

collected from multiple parties. The training process results in a model that is useful for AI tasks, and also generates internal states or data such as gradients, Hessian matrices, and intermediate model parameters. When a user submits an unlearning request, the model owner employs some unlearning oracle based on the cached data to reproduce an unlearned model. Ideally, the unlearned model should completely forget what it has learned about the data to be unlearned. For example, in the dog-cat image classification task, if the data of dogs needs to be unlearned, then the unlearned model will not be able to recognize dogs anymore.

In fact, making a model forget some training samples is quite trivial. For example, one can simply add a large amount of noise to the model's parameters, which definitely removes the influence of the unlearned data – but also destroys the model's utility on the remaining data. Therefore, the challenge of machine unlearning lies in how to unlearn the targeted data while avoiding catastrophic forgetting [170] – a phenomenon where a model rapidly loses accuracy on some tasks when fine-tuned for others.

2.7.2 Related work – machine unlearning techniques

Considering unlearning strategies, current machine unlearning techniques could be classified into three categories: training-based unlearning, tuning-based unlearning and model-based unlearning.

2.7.2.1 Training-based machine unlearning

Training-based machine unlearning aims to speed up the retraining process or use fewer training rounds to improve model accuracy. Bourtoule et al. [9] proposed an

unlearning framework SISA that partitions the training data into multiple shards. Each shared creates a unique model checkpoint so that the retraining can be invoked from the intermediate state. Liu et al. [83] developed a rapid retraining algorithm to remove data samples from a federated learning model. They employed the Fisher Information Matrix (FIM) to approximate the Hessian for Quasi-Newton optimization, which is computationally cheaper than the traditional L-BFGS algorithm [8]. Liu et al. [81] presented a client-level data removal algorithm to eliminate the influence of a client's data. The proposed algorithm relies on the storage of the central server to retain historical submissions for each client, which are then calibrated by retrained parameters to speed up the unlearning process. Wang et al. [143] proposed a pruning-based category-level unlearning algorithm. They leveraged TF-IDF [110] to evaluate the relevance between channels and classes, and erased the discriminative channels of the category to be unlearned. Their model's performance is restored by fine-tuning the pruned model on the remaining dataset. Wu et al. [149] focused on addressing the incremental effect when they removed specific clients. They directly subtracted one client's updates from the global model's parameters, and further employed knowledge distillation to improve model accuracy. The original global model is adopted as the teacher model to recover the damage caused by unlearning operations.

2.7.2.2 Tuning-based machine unlearning

Tuning-based machine unlearning directly modifies the parameters of a trained model to erase the influence of unlearned data, where the modification is usually calibrated by particular influence functions. Golatkar et al. [46] used the Hessian-gradient product to represent the contribution of the data to be unlearned. They performed a reverse Newton step on the trained model to scrub the information from the unlearned data under a local quadratic approximation. Similarly, Guo et al. [50] adopted influence theory [71] to evaluate the impact of a training sample, and employed differentially private loss perturbation [17] to mask the gradient discrepancy caused by removing a particular data point. Their proposed ϵ-certificated removal is in fact a relaxation of ϵ-differential privacy on a particular data record. Sekhari et al. [122] followed Newton's method to optimize the empirical loss, and deleted the influence of the targeted data by subtracting an inversed Hessian from the trained model. Golatkar et al. [47] derived an optimal scrubbing scheme based on the theorem of neural tangent kernels (NTK) [61]. Their algorithm modifies the model's parameters as a function of the tangent kernel, which is more component in deep learning due to the over-parameterization of deep models.

CHAPTER 3

JOINT DIFFERENTIAL PRIVACY AND AUCTION GAME: ENHANCING THE ROBUSTNESS OF FEDERATED LEARNING FRAMEWORK

Federated learning is a promising distributed machine learning paradigm that has been playing a significant role in providing privacy-preserving learning solutions. However, alongside all its achievements, there are also limitations. First, traditional frameworks assume that all the clients are voluntary and so will want to participate in training only for improving the model's accuracy. However, in reality, clients usually want to be adequately compensated for the data and resources they will use before participating. Second, today's frameworks do not offer sufficient protection against malicious participants who try to skew a jointly trained model with poisoned updates. To address these concerns, this chapter develops a more robust federated learning scheme based on joint differential privacy. The framework provides two game-theoretic mechanisms to motivate clients to participate in training. These mechanisms are dominant-strategy truthful, individual rational, and budget-balanced. Further, the influence an adversarial client can have is quantified and restricted, and data privacy is similarly guaranteed in quantitative terms. Experiments with different training models on real-word datasets demonstrate the effectiveness of the proposed approach.

3.1 Introduction

The rapidly developing techniques in artificial intelligence (AI) are greatly improving traditional machine learning methods and bringing enormous potential for future innovation along with them. The past few years have witnessed a boom in AI solutions in computer vision, speech recognition, medical diagnosis, and more [161]. However, although computing power has generally increased, so has the amount of data needed to build a high-performing model. As such, it is no longer practical to collect sufficient data and train a model with a single processing unit. Federated learning [48], a promising learning paradigm for distributed settings, is being used to overcome this issue. In federated learning, each client trains a local model using their own data, and updates the model's parameters or gradients to the server. The server aggregates all the parameters from clients and update the global model round by round. In most federated learning schemes, only the model's parameters are shared to the server, each client's data are kept invisible and anonymized to other parties and their privacy can be protected to some extent.

Yet, despite these benefits, federated learning still faces several critical challenges. The first and foremost one corresponds to clients' incentive, which is usually overlooked by most federated learning frameworks [164]. Incentive refers to the rewards or gains of clients that encourage them to participate in a training, or the inducement that make them behave actively and reliably in the training process [65]. The incentive issue seems redundant in federated learning as each client has access to the final trained model and actually benefits from it. However, researches has demonstrated that this is far from being enough [128, 131, 163]. In fact, being a part of a federated learning scheme demands a great deal of overhead on the part of the client, and so their excessive consumption of time, energy, bandwidth, etc., may need to be compensated. More importantly, nowadays data are regarded as valuable and private assets, which can rarely be contributed for free. Hence, the traditional assumption that all the clients are voluntary to participate in the federated learning is somewhat over-optimistic. From this perspective, an incentive mechanism is an indispensable element of a federated learning system.

In addition to the incentive problem, malicious participants attempting to poison the jointly learned model is another issue. Researchers have demonstrated that federated learning is generally vulnerable to manipulation by adversarial clients, like model and data poisoning [5, 7] in particular. In this chapter, we focused on dirty-label data

poisoning attack, where a malicious participant introduces a number of data samples with modified labels in the training set to cause a mis-classification of the global model. As clients' data are never shared in federated learning, the server cannot verify the data used in each local training. Thus, dirty-label data poisoning attack provides a convenient way for malicious clients to manipulate the global model. Therefore, framework designs need to be more robust and defense strategies need to be more reliable.

As a last point, federated learning works without disclosing the clients' raw data. However, their private information can still be compromised by inferring information from the parameter updates [124]. Hence, an optimal security strategy against poisoning attacks should also provide protection against privacy breaches through inference.

In this chapter, we proposed a game-theoretic framework to advance federated learning, aiming at guiding client selection and limiting the influence of adversaries. The framework not only introduces incentives and compensation for participation, but also improves defenses against poisoning attacks. Specifically, our solutions treat clients as purveyors of training power, and the server is wired to behave like a buyer. Various allocation and payment rules are designed to guarantee truthfulness and individual rationality during the "buying" process, i.e., the process of selecting clients to participate in training. Further, should a particular client be adversarial, the mechanism applies joint differential privacy to limit its impact. To demonstrate the effectiveness of this framework, we conducted experiments with several real-world datasets and different learning models. The main contributions of this chapter therefore include:

- A jointly differentially private federated learning framework that limits the impact of adversarial clients and provides a rigorous privacy guarantee.

- Two truthful game-theoretic mechanisms the server can use to select participating clients for training.

- Theoretical proofs and analyses of the proposed mechanisms. The client selection mechanisms satisfy game-theoretic properties and are robust against strategic manipulation by clients.

- Experimental simulations that show the the rationale behind the mechanisms. Further experiments with real-world datasets demonstrate the effectiveness of the proposed federated learning scheme.

In the rest of this chapter, we first introduce basic background knowledge of federated learning and then detail the client selection mechanisms and learning framework. Theoretical analyses and experimental results are also given.

3.2 Preliminaries

3.2.1 The game-theoretic properties used in this chapter

In federated learning, the server needs to recruit a number of clients to perform a training task. However, given that the clients each have a different dataset and diverse computation abilities, the energy and time costs etc. for them to participate in the training also varies. Hence, our framework treats the client selection process as an auction game, where clients submit their cost c_i as a bid and the server decides the winners and payments p_i based on its budget B. If a client is selected by the server, its utility is $u_i = p_i - c_i$, and $u_i = 0$ otherwise.

Clients are assumed to be rational and self-interested. In other words, each client only cares about its own utility and tries to maximize it, even if that means misreporting information. However, the server is incentivized to attract as many participating clients as possible with its limited budget B. Specifically, the client selection mechanism should satisfy the following game-theoretic properties:

Definition 3.1. *(Dominant-strategy truthful) A client selection mechanism $\mathscr{M}$ is dominant-strategy truthful if, for every client, truthfully reporting his cost c_i is a dominant strategy. That is, given c_i and all possible $c_i' \in \mathbb{R}$:*

$$u_i(c_i, c_{-i}) \geq u_i(c_i', c_{-i}), \tag{3.1}$$

where c_{-i} represents the strategy profile of clients except for the ith component, $c = (c_i, c_{-i})$.

Truthfulness means that no client can improve their utility by misrepresenting their data, which is an important property in developing game-theoretic solutions.

Definition 3.2. *(Individual rationality) A mechanism $\mathscr{M}$ is individual rational for each client if the clients gain non-negative utility by participating in $\mathscr{M}$.*

$$u_i = p_i - c_i \geq 0. \tag{3.2}$$

Individual rationality guarantees the baseline utility of each client. If participating in a mechanism possibly leads to negative utility, clients may decline to participate at all.

Lastly, from the server's perspective, its total payments must stay within its predefined budget.

Definition 3.3. *(Budget balance) A client selection mechanism $\mathscr{M}$ is budget-balanced if the total amount paid by the server does not exceed its budget. Formally, given m selected clients, each receives a payment p_i, then*

$$\sum_{i=1}^{m} p_i \leq B. \tag{3.3}$$

3.2.2 Problem definition

First, we need to find a dominant-strategy truthful client selection mechanism, such that the server can select participating clients while minimizing the risk of strategic manipulation. Second, we need to consider situations where one adversarial client has "slipped through the cracks" and been selected to participate in the training: we need to find a way to limit the impact of this adversarial client. Finally, the solutions we derive must protect each client's data privacy. Formal definitions of these problems follow.

Consider a server S that wishes to recruit m out of n clients to collaboratively train a global model. Each client C_i has a database D_i that contains $d_i = |D_i|$ data records. The server S publishes the training task and each client submits their estimated costs c_i, which is the minimum amount of payment that incentivizes them to participate in the training. Given a cost vector $(c_i, \ldots, c_n)$, the server must select m participating clients while satisfying the game-theoretic properties outlined in Definitions 3.1-3.3. Moreover, the m clients selected include one adversarial client, who will try to skew the trained model by submitting dirty parameters during the training process. The server cannot distinguish which client is adversarial but wishes to reduce its impact.

For the purposes of this chapter, we assume that the server S is not trustworthy. As a result, no client can submit their trained parameters directly to the server for fear of a privacy breach. Moreover, we also assume that the server knows the amount of data d_i used by each client. This assumption is without loss of generality because in many federated learning scenarios d_i is not regarded as private information. For example, in a training of default prediction model, the central authority (e.g., bank regulator) actually knows the amount of registered members in each organization (e.g., banks). And, many input recommendation models in mobile phone are locally trained based on a length-fixed prefixes of user's previous inputs (e.g. 5 words). Even if d_i cannot be known, we can simply set $d_i = 1$, which removes d_i's influence in the proposed mechanisms. As d_i is not

involved in the derivation of game-theoretic properties, the proposed mechanisms still work without knowing an actual d_i.

3.3 The jointly differentially private federated learning framework

3.3.1 Overview of the framework

The proposed framework consists of a central server and multiple clients, as pictured in Figure 3.1.

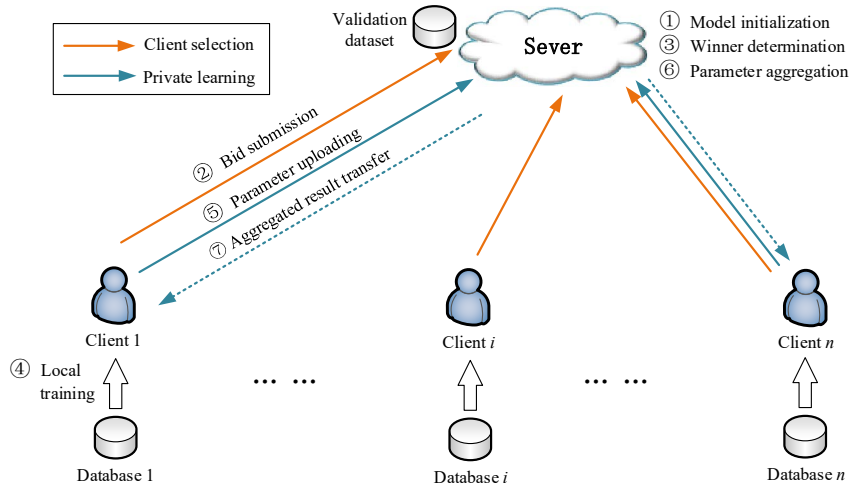


Figure 3.1: Our jointly differentially private federated learning framework

We provide an example to describe the workflow of the proposed framework. Consider there are $n = 15$ clients $\{C_1, C_2, \ldots, C_{15}\}$, each with their own private database that might be used to collaboratively train a machine learning model. A central server S is responsible for selecting participating clients and aggregating clients' submitted parameter updates. To arrive at a global model, each party in the framework must execute the following steps in succession.

① *Model initialization.* The server S initializes the model that is to be trained and publishes necessary information about this model to all clients, e.g., the number of parameters, the network structure, etc.

② *Bid submission.* Each client C_i receives the model's information from the server, and evaluates their cost c_i of participating in the training. The cost can be evaluated by considering the energy consumption, communication cost and the valuation of their training dataset [42, 131]. Each client then submits their evaluated costs c_i as a bid to the server S.

③ *Winner determination.* The server S employs a dominant-strategy truthful mechanism to decide the winning clients and corresponding payments. It then publishes the result of the bidding process. The winning clients then prepare for the coming local training (e.g., Client 1 in Figure 3.1). Clients that were not selected are not included in any further training. For the purposes of this example, we assume that clients $\{C_1, C_2, \ldots, C_{10}\}$ are winners selected by the server.

④ *Local training.* Each winner downloads the global model from the server S and performs local training using their own device and data D_i. In each round, the locally trained model parameters of client C_i are denoted as θ_i.

⑤ *Parameter uploading.* The winners add local perturbation to their parameters θ_i, and then upload the perturbed parameters $\hat{\theta}_i$ to the server S according to predefined uploading rules. Each θ_i is perturbed with Gaussian noise $v_i \sim \mathcal{N}(0, \sigma)$, i.e., $\hat{\theta}_i = \theta_i + v_i$.

⑥ *Parameter aggregation.* The server constructs client groups $g_i = \{C_j\}_{j \neq i}$, each contains all the winning clients except for a particular one.. The server then employs an exponential mechanism to select a group g_i^*, and computes $(agg = \sum d_i \hat{\theta}_i, sum = \sum d_i)$ for $C_i \in g_i^*$. The probability of being selected is determined using the model's accuracy evaluated on the server's validation dataset. For example, if $g_1 = \{C_2, \ldots, C_{10}\}$ is selected by the mechanism, then the aggregated parameters will be $(agg = \sum_{i=2}^{10} d_i \hat{\theta}_i, sum = \sum_{i=2}^{10} d_i)$.

⑦ *Aggregated result transfer.* The server S transfers the aggregated parameters (agg, sum) to each client. The clients update their local model by $\theta_i = \frac{agg + d_i \theta_i}{sum + d_i}$, and return to Step ④ for the next round of training. For example, Client 1 would update its local parameters with the formula $\theta_1 = \frac{agg + d_1 \theta_1}{sum + d_1}$.

After each round, the server S tests the accuracy of the global model using the newest θ_i selected by the exponential mechanism. Once the accuracy of the global model reaches

a satisfactory threshold, the server stops the training. Each client will receive their payment after the training stops.

The proposed framework involves an auction process compared to traditional ones. As clients may deliberately overbid or underbid their costs, the bids are not considered private information. Suppose there are n candidate clients participating in the bidding, each client submits their evaluated cost to the server, and the server publishes the result after determining winners. During the auction process, the communication overhead is $O(n)$.

We note that, non-myopic rational adversaries may bear the loss incurred in the client selection process and strategically underbid their cost to make themselves selected. This happens when an adversary's expected future gain by attack outweighs its previous loss in client selection. Such scenarios are complicated to analyze because it is hard to quantify the actual gain of an adversary without introducing more assumptions about their (may be heterogeneous) utility function. To address this issue, we introduced joint differential privacy in the private learning stage, which limits the influence of an adversary.

3.3.2 The client selection stage

The keys to the client selection stage (i.e., Steps ②~③) are the client selection mechanisms, which are designed to choose appropriate clients while incentivizing truthful reporting. As the proposed framework is built for the possibility that an adversarial client may be a participant, both normal and adversarial clients are considered.

- *Normal clients.* Normal clients are those benign clients who participate in the training without any intention of sabotaging the trained model. These clients are rational and wish to maximize their utility u_i. Therefore, to get a higher payment, a normal client may strategically misreport their expected costs c_i to the server S either by over- or underbidding the amount. In addition, normal clients will follow the predefined rules about parameter uploading, and they care about their privacy.

- *Adversarial clients.* Adversarial clients want to sabotage the trained model in some way. For example, they may upload random noise to reduce the model's accuracy [19], or train their local model with incorrectly labeled data so as to skew the global update [7]. Like normal clients, adversarial clients can also strategically overbid or underbid their costs c_i to obtain higher payments. The adversarial client's utility is generally determined by the misleadingness of a global model

or the extent of disruption caused to the training process. Since the adversarial clients may have distinct purposes, we do not explicitly quantify their utility.

Importantly, the server cannot distinguish between the two types of clients. Thus, the client selection mechanisms must be dominant-strategy truthful to avoid strategical manipulation. We present two mechanisms that satisfies this requirement. The first mechanism is dominant-strategy truthful, individual rational and envy-free, which means that each client likes its outcome at least as well as the outcome of anyone else. The second mechanism is also dominant-strategy truthful and individual rational, besides, it satisfies the server's specific optimization requirements for selecting clients. In addition, both mechanisms are budget-balanced.

3.3.2.1 A simple, envy-free client selection mechanism

The aim of the client selection mechanism $\mathscr{M}_1$ is to guarantee game-theoretic properties, such as truthfulness and individual rationality. After collecting the costs c_i from each client i, the server runs Algorithm 1 to decide the winners and corresponding payments. The algorithm first computes the unit prices of each piece of data $q_i = c_i/d_i$ at client i, and sorts them into increasing order. Then it finds the largest integer $m \in [n]$ that satisfies $q_m \leq \frac{B}{\sum_{i=1}^{m} d_i}$. The winning clients are C_i, $i \in [m]$ and their payments are determined by $p_i = \min\{\frac{B}{\sum_{i=1}^{m} d_i}, q_{m+1}\} \cdot d_i$. Any clients who are not selected to participate receives zero payment, i.e., $p_i = 0$ for $i \in [m+1, n]$, and are no longer considered.

Algorithm 1 Truthful client selection mechanism $\mathscr{M}_1$

Input: cost c_i from each client, the server's total budget B .
Output: winning clients for FL training, the payment p_i for each client.

for $i \leq n$ **do**
  $q_i \leftarrow \frac{c_i}{d_i}$.
end for
sort q_i in an increasing order, $q_1 \leq q_2 \leq \cdots \leq q_n$, breaking ties arbitrarily.
find the largest $m \in [n]$ that satisfies $q_1 \leq q_2 \leq \cdots \leq q_m$ and $q_m \leq \frac{B}{\sum_{i \in S} d_i}$, breaking ties arbitrarily.
for $1 \leq i \leq m$ **do**
  $p_i \leftarrow \min\{\frac{B}{\sum_{i \in S} d_i}, q_{m+1}\} \cdot d_i$.
end for
for $m < i \leq n$ **do**
  $p_i \leftarrow 0$.
end for

The design rationale of $\mathscr{M}_1$ is as follows. What makes a client attractive to a server is low costs and a large amount of data offered. Therefore, the heuristic trades off between these two properties by considering the data's unit price. In other words, the server S *buys* the *goods* (data) from the *sellers* (clients) according to the unit *selling price* of their goods. The detailed game-theoretic properties of the mechanism are proofed and discussed in Section 3.4.

3.3.2.2 Client selection with the server's requirement

Although mechanism $\mathscr{M}_1$ satisfies the desired game-theoretic properties, it does not consider the server's specific requirements. To treat server S as a data buyer in a market, we must assume a specific goal that is driving its data purchases. For example, the server may wish to reach an accuracy threshold with the minimum amount of spending, or to achieve the highest possible accuracy with a fixed monetary budget [42].

However, in our framework, the server S only knows the amount of data each client holds, it cannot assess the quality of the data. From the server's perspective, guaranteeing the performance of the trained model demands as much training data as it can possibly buy with its money. As such, it is reasonable to formulate the client selection problem as a social welfare maximization problem, where the social welfare is defined as the total amount of data the server can possibly buy. Formally, the problem is described as follows.

$$\begin{aligned} &\max \sum_{i=1}^{n} x_i \cdot d_i \\ &\text{s.t. } \sum_{i=1}^{n} x_i \cdot c_i \leq B,\ \forall i, x_i \in \{0,1\}. \end{aligned} \tag{3.4}$$

This formulation is a typical 0/1 knapsack problem, where d_i represents each client's weight, and c_i is the corresponding size. The client selection process is therefore a variant of the knapsack auction, where the size is the private information of bidders, and the weights are public.

Knapsack problems are NP-hard; therefore, an approximation algorithm [96] is used to find winning clients, and the corresponding payments are determined following Myerson's lemma [98]. The details of the mechanism are shown in Algorithm 2.

The design rationale of $\mathscr{M}_2$ is as follows. The algorithm approximately solves the knapsack problem and decides the winning clients. The payment p_i for winner i is then determined using the critical bid – the infimum of the bids i could make and continue to win. The allocation function alloc_i describes the allocation status of client i (i.e., win

Algorithm 2 Truthful client selection mechanism $\mathscr{M}_2$

Input: cost c_i from each client, the server's total budget B .

Output: winning clients for FL training, the payment p_i for each client.

for $i \leq n$ **do**
  $r_i \leftarrow \frac{d_i}{c_i}$.
end for
sort r_i in an decreasing order, $r_1 \geq r_2 \geq \cdots \geq r_n$, breaking ties arbitrarily.
$paid \leftarrow 0$, $selected \leftarrow \emptyset$, $i = 1$.
while $paid + c_i \leq B$ **do**
  $selected \leftarrow selected \cup \{i\}$.
  $paid \leftarrow paid + c_i$.
  $i \leftarrow i + 1$.
end while
if $\sum_{j \in selected} d_j \geq d_{j+1}$ **then**
  Output $selected$.
else
  Output $\{i^*\}$ that satisfies $i^* = \arg\max_i d_i$.
end if
for $i \in selected$ **do**
  $p_i \leftarrow \int_0^{c_i} z \cdot \frac{\mathrm{d}}{\mathrm{d}z} \mathsf{alloc}_i(z, c_{-i}) \mathrm{d}z$.
end for

or lose) given the other clients' bids are fixed. The game-theoretic properties of the mechanism are discussed in Section 3.4.

The difference between mechanisms $\mathscr{M}_1$ and $\mathscr{M}_2$ follows. Mechanism $\mathscr{M}_1$ is easier to compute. It allows the server to decide winning clients by simply sorting the unit prices of the received bids, and the desired game-theoretic properties are satisfied without extra argumentation. In comparison, mechanism $\mathscr{M}_2$ is based on an optimization problem, which may be hard to solve, e.g., NP-hard problems. Moreover, the incentive compatibility of mechanism $\mathscr{M}_2$ also needs further proof, i.e., the monotonicity of client selection rule. Generally, $\mathscr{M}_2$ leads to more computational cost of the server. In practice, the choice of the mechanism depends on the server's requirements.

Remark 1. In mechanism $\mathscr{M}_2$, the heuristic used to solve the knapsack problem is a typical 2-approximation algorithm. Therefore, according to the Myerson's lemma, we can deduce a truthful polynomial time mechanism that achieves at least 50% of the optimal social welfare. Note that there are advanced approximation algorithms that can achieve a higher social welfare while still producing a monotone allocation rule. For example, the truthful polynomial time approximation scheme (PTAS) indeed produces at least a $(1-\beta)$-fraction of the optimal social welfare and runs in time polynomial in n and $1/\beta$ [10].

But we do not discuss them further. It suffices to provide a feasible solution here.

Remark 2. In the client selection stage, maximizing the amount of training data need not necessarily be a requirement for the server. This federated learning framework supports various server requirements. For example, the server may want to involve more clients in the training to increase diversity in the data sources. With different requirements, the optimization problem will also vary. In fact, the optimization problem itself does not influence the game-theoretic properties. What really matter is whether the (approximate) solution of the optimization problem leads to a monotone client selection rule [98]. The Myerson's lemma works as a linkage between the optimization solution and game-theoretic properties. Therefore, the solution of the optimization problem should be carefully designed.

3.3.3 The private learning stage

The private learning stage spans Steps ④~⑦ of the proposed framework. The following explanation is based on the same assumption that one of the clients participating in the training is adversarial. Intuitively, the ideal way to deal with an adversarial client is to simply pick it out and throw it away. However, as the server cannot judge which clients are adversarial and which are not, the second-best method is to limit the impact that an adversary can have on the global model. Differential privacy, which limits the influence of any individual on the final aggregated output, is a useful tool to achieve this goal.

Unfortunately, traditional full differential privacy imposes its bounds over the whole client set, which would here make the aggregated parameters almost independent of each client's submission. This would impair the contribution of high-caliber clients. As a result, the relaxed definition offered by joint differential privacy better suits our needs. The rest of this section outlines the process of guaranteeing joint differential privacy during parameter aggregation process.

In Step ④, the winning clients download the global model with the initialized parameters from the server S, and perform local training using their own data and device. After a certain number of local iterations, each client uploads its locally trained parameters to the server S for aggregation. However, as the parameter set still contains sensitive information [5, 124], these data must be sanitized before transmission.

Hence, in Step ⑤, each client uses mechanism $\mathscr{M}_3$ to perturb their parameters with randomized noises drawn from Gaussian distribution. The noise is calibrated to a sensitivity of $\Delta_2 f$ and a privacy budget ϵ. The actual value of $\Delta_2 f$ depends on specific training model. For example, with linear regression, the sensitivity would be derived

using the support bound and the norms of the parameters [23]. Once sanitized, the parameters $\hat{\theta}_i$ are sent to the server.

Algorithm 3 Local perturbation mechanism $\mathscr{M}_3$

Input: the private data θ_i of client i, privacy budget ϵ_L, sensitivity $\Delta_2 f$, clipping threshold C.

Output: perturbed data $\hat{\theta}_i$.

1: sample $Z_i \sim \mathcal{N}(0,\sigma)$, where $\sigma = \sqrt{2\ln\frac{1.25}{\delta}} \cdot \frac{\Delta_2 f}{\epsilon_L}$

2: $\theta_i \leftarrow \theta_i / \max\{1, \frac{\theta_i}{C}\}$.

3: $\hat{\theta}_i \leftarrow \theta_i + Z_i$.

4: **output** $\hat{\theta}_i$.

In Step ⑥, the server S receives the noisy parameters and uses the parameter aggregation mechanism $\mathscr{M}_4$ to produce feedback for each client. The most widely used parameter aggregation method in traditional federated learning frameworks is a weighted average with respect to the client's data, specifically, $\theta = \sum_{i=1}^{m} d_i \hat{\theta}_i / \sum_{i=1}^{m} d_i$. However, this approach incorporates an adversarial client's bad updates alongside all the normal clients' good updates. Hence, to reduce this negative influence and achieve joint differential privacy, the server evaluates the performance of parameters following a leave-one-out strategy, where an exponential mechanism is employed to pick up a high-quality output. Algorithm 4 shows the details.

After receiving the noisy parameters, the server S first constructs m client groups, each contains $m-1$ different clients. The server then computes the weighted average of parameters within each group, and employs an exponential mechanism to select an aggregated result as final output. The score function is the model's accuracy evaluated on the server's validation dataset, namely $y = \mathsf{Acc}_D(\hat{\theta}_i)$. The score function represents the goodness of aggregated parameters from each client group. We note that the accuracy is not the only way to measure the goodness of clients' updates, there are also important factors, such as the number of data used in training, that can be adopted in the score function to further weight the importance of each group. We use the model accuracy here for simplicity. The sensitivity of the score function is $\Delta y = 1/(m-1)$ because the model's accuracy is within interval $[0,1]$. In this way, the adversarial client can only influence the output of parameter aggregation *with some probability*, which is related to the quality of parameters it submits. Moreover, although the adversarial client's current submission could be selected in an aggregation, there is still a chance that its next submission be rejected in the next round. The negative impact of the adversarial client's submission is rigorously limited by differential privacy.

Algorithm 4 Parameter aggregation mechanism $\mathscr{M}_4$

Input: the noisy data $\hat{\theta}_i$ of each client, privacy budget ϵ_E
Output: aggregated data (agg, sum) for each client

1: **construct** m client groups $g_i = \{Cj\}_{j \neq i}$.
2: **for** each group g_i **do**
3: $\quad agg_i \leftarrow \sum_{j \in g_i} d_j \hat{\theta}_j$, $sum_i \leftarrow \sum_{j \in g_i} d_j$.
4: $\quad \hat{\theta}_i = agg_i / sum_i$.
5: **end for**
6: **for** $1 \leq i \leq m$ **do**
7: $\quad$ **compute** $y \leftarrow \mathsf{Acc}_D(\hat{\theta}_i)$.
8: **end for**
9: $\Delta y \leftarrow \frac{1}{m-1}$.
10: **pick up** a g_i^* with probability $\propto \frac{\epsilon_E \cdot y(D,g)}{2\Delta y}$.
11: **for** $1 \leq i \leq m$ **do**
12: $\quad$ **send** client C_i the aggregated data pair (agg_{i^*}, sum_{i^*}) computed from g_i^*.
13: **end for**

In Step ⑦, each client $i \in [m]$ receives the aggregated parameter (agg, sum) from the server S. The data pair needs to be further processed to satisfy joint differential privacy. Specifically, client i first computes $d_i \theta_i$ using its true data θ_i, and then combines the private $d_i \theta_i$ with the public data pair to update its local parameters, as shown in Algorithm 5. Once combined, each client can test the new parameters with their local data, and continue the training process to for the next round of updates.

The intuition behind the private training stage is as follows. According to the billboard lemma (Lemma 4.1), to achieve joint differential privacy, we need to generate a differentially private output as signal. Then, if each client's result is only a function of the signal and their own data, the overall algorithm is jointly differentially private. Here, the data pair (agg, sum) selected by the exponential mechanism is treated as signal, and each client's parameter is updated by combining the agg and their true data θ_i. Therefore, the new parameters of all the clients together satisfy joint differential privacy – namely, no matter what data a client reports, the joint distribution of the other clients' new parameters is almost the same. A complete proof of the privacy guarantee is presented in Section 3.4.

3.4 Theoretical analysis

The theoretical analysis consists of two parts: a game-theoretic analysis of the client selection strategies (namely $\mathscr{M}_1$ and $\mathscr{M}_2$) and a privacy analysis of the private learning

Algorithm 5 Jointly differentially private local update $\mathscr{M}_5$

Input: the data pair (agg, sum), C_i's true parameters θ_i
Output: jointly differentially private parameter for client i

1: $\theta_i \leftarrow \frac{agg+d_i\theta_i}{sum+d_i}$.
2: **test** the new parameters θ_i using client i's local dataset.
3: **generate** the parameters for next round of submission.

stage. The client selection analysis explores the game-theoretic properties $\mathscr{M}_1$ and $\mathscr{M}_2$ provide, such as truthfulness and individual rationality. The private learning analysis examines the privacy guarantee promised by the mechanism with respect to specific privacy budget constraints.

3.4.1 Analysis of the client selection stage

The analysis begins with proof that the mechanism $\mathscr{M}_1$ satisfies desired game-theoretic properties.

Theorem 3.1. *The mechanism $\mathscr{M}_1$ is individual rational. That is, each client gets non-negative utility $u_i = p_i - c_i \geq 0$ by participating in $\mathscr{M}_1$.*

Proof. By the payment rule, we know that $q_m \leq \frac{B}{\sum_{i\in S} d_i}$ and $q_{m+1} \geq q_m$. Thus, $\min\{\frac{B}{\sum_{i\in S} d_i}, q_{m+1}\} \geq q_m \geq q_i$ for $i \in S$. For each selected clients, $u_i = p_i - c_i = p_i - q_i \cdot d_i = \min\{\frac{B}{\sum_{i\in S} d_i}, q_{m+1}\} \cdot d_i - q_i \cdot d_i \geq 0$. For the clients who are not selected, their payment is $p_i = 0$, and their cost is $c_i = 0$, either. Therefore, $u_i \geq 0$ for every $i \in [n]$. □

Theorem 3.2. *The mechanism $\mathscr{M}_1$ is dominant-strategy truthful; clients do not benefit from mis-reporting.*

Proof. Suppose client i reports c_i' instead of c_i, and the position of $q_i' = \frac{c_i'}{d_i'}$ in the sequence $\{q_1, \ldots, q_n\}$ changes from k to k', the payment changes from p_i to p_i'.

Situation 1. If client i is selected by reporting c_i truthfully, that means, $k \leq m$ and $q_k \leq q_m$.

- If client i reports a fake $c_i' < c_i$, then $q_i' \leq q_i$, and the position of client i moves up in the sequence of $\{q_1, \ldots, q_n\}$, i.e., $k' \leq k \leq m$. In this situation, client i is still selected, according to the payment rule, $p_i' = p_i$, and the utility of agent i will not change.

- If client i reports a fake $c_i' > c_i$, then $q_i' \geq q_i$, the position of client i moves back in the sequence of $\{q_1, \ldots, q_n\}$, i.e., $k' \geq k$. If $k \leq k' \leq m$, client i is still selected,

according to the payment rule, $p_i' = p_i$. If $k \le m \le k'$, then client i would not be selected, according to the payment rule, $p_i' = 0$. In summary, $p_i \ge p_i'$ and client utility $p_i - c_i \ge p_i' - c_i$.

Situation 2. If client i is not selected by reporting c_i truthfully, that means, $k > m$ and $q_k > q_m$.

- If client i reports a fake $c_i' > c_i$, then $q_i' \ge q_i$, the position of client i moves back in the sequence of $\{q_1, \ldots, q_n\}$, i.e., $k' \ge k > m$. In this situation, client i will not be selected, $p_i' = p_i = 0$.

- If client i reports a fake $c_i' < c_i$, then $q_i' \le q_i$, and the position of client i will move up in the sequence of $\{q_1, \ldots, q_n\}$, i.e., $k' \le k$. If $m \le k' \le k$, then client i will not be selected and the payment $p_i' = p_i = 0$. If $k' \le m \le k$, then client i would be selected, according to the payment rule, $p_i' = \min\{\frac{B}{\sum_{i \in S} d_i}, q_{m+1}\} \cdot d_i$. In this case, $p_i' = \min\{\frac{B}{\sum_{i \in S} d_i}, q_{m+1}\} \cdot d_i \le q_{m+1} \cdot d_i \le q_k \cdot d_i = c_i$. As a result, $u_i = p_i - c_i \le 0$. That means, although client i was selected after misreporting $c_i' < c_i$, the payment it receives will not cover the cost it incurs by participating in the training.

Given these scenarios, we conclude that clients will not benefit from mis-reporting. □

Theorem 3.3. *The mechanism $\mathscr{M}_1$ is budget-balanced, the total payment paid by the server does not exceeds its total monetary budget.*

Proof. By the payment rule, we know that for the selected clients, the total payment they receive is $\sum_{i \in S} p_i = \sum_{i \in S} \min\{\frac{B}{\sum_{i \in S} d_i}, q_{m+1}\} \cdot d_i \le B$, and the payments for unselected clients are 0, therefore, $\sum_{i=1}^{n} p_i \le B$. □

Before showing properties of mechanism $\mathscr{M}_2$, we first introduce Myreson's lemma.

Definition 3.4. *(Implementable allocation rule) An allocation rule alloc for a single-parameter environment is implementable if we can find a payment rule p such that the mechanism $\mathscr{M} = (alloc, p)$ is truthful.*

Definition 3.5. *(Monotone allocation rule) An allocation rule alloc for a single-parameter environment is monotone if for every client $i \in [n]$ and bids b_{-i} of the other clients, the allocation $alloc_i(z, b_{-i})$ to client i is nondecreasing in bid z.*

Intuitively, this monotonicity means the winner in an auction still wins (or wins more) by bidding at a higher value (or equivalently, a lower cost).

Lemma 3.1. *(Myerson's lemma [98]) For single parameter environments, the following three claims hold.*

(1) An allocation rule is implementable if and only if it is monotone.

(2) If an allocation rule alloc is monotone, then there exists a unique payment rule p such that the mechanism $\mathscr{M} = (alloc, p)$ is truthful, assuming that a zero bid implies a zero payment.

(3) The payment rule is given as

$$p(b_i, b_{-i}) = b_i alloc_i(b_i, b_{-i}) - \int_0^{b_i} alloc_i(z, b_{-i}) dz. \tag{3.5}$$

Theorem 3.4. *The mechanism $\mathscr{M}_2$ is dominant-strategy truthful; clients will not benefit from mis-reporting.*

Proof. According to Myerson's lemma, the core of the proof is to show that the allocation rule in mechanism $\mathscr{M}_2$ is monotone. Note that, in our framework, the server S *buys* data from clients, which is performed as a reversed auction. Therefore, we need to show that the winning client still wins by lowering its reported costs.

For any $(d, c) \in \mathbb{R}^n_{\geq 0}$, and $\forall i$, let $c'_i < c_i$. Denote $c' = (c'_i, c_{-i})$, $T = \mathscr{M}_2(d, c)$, and $T' = \mathscr{M}_2(d, c')$. Denote $Q = selected(d, c)$, and $Q' = selected(d, c')$.

We first show that $i \in Q \Rightarrow i \in Q'$. This is because Q and Q' are the prefixes of clients with cumulated costs less than B in the ordered sequence of r_i. When client i lowers its costs from c_i to c'_i, the position of the client only moves up in the sequence. Therefore, client i is still in the prefix Q' if it was in the prefix Q.

Based on the observation that $i \in Q \Rightarrow i \in Q'$, we know that if $T = Q$ and $T' = Q'$, the mechanism is monotone. In addition, note that if $i \in Q$, then $\sum_{j \in Q'} d'_j \geq \sum_{j \in Q} d_j$. Therefore, if $T = Q$ and $T' = Q$, the mechanism is also monotone.

Finally, if $i \in T$, $i = i^*$ and $d_i > \sum_{j \in Q} d_j$, in this case, we must also have $i \in T'$. Client i remains the one with the highest d_i, and thus the output is either $T' = \{i^*\}$ or $T' = Q'$ with $i \in Q'$.

Therefore, we conclude that $i \in T \Rightarrow i \in T'$. That is, client i still wins by lowering its bid. In other words, the allocation rule produced by mechanism $\mathscr{M}_2$ is monotone. According to the Myerson's lemma, mechanism $\mathscr{M}_2$ is truthful, and the payment is computed by Equation 3.5. □

3.4.2 Analysis of the private learning stage

Before discussing the privacy guarantee of the mechanisms, we need to introduce some theorems that are widely used in proofs of differential privacy's mathematical properties.

Lemma 3.2. *(Parallel Composition [89]) Given a set of mechanisms $\mathscr{M} = \{\mathscr{M}_1, \ldots, \mathscr{M}_m\}$, if $\mathscr{M}_i$ provides ϵ_i privacy guarantee on a disjointed subset of the entire dataset, $\mathscr{M}$ will provide* $\max\{\epsilon_1, \ldots, \epsilon_m\}$*-differential privacy.*

Lemma 3.3. *(Sequential Composition [89]) Suppose a set of mechanisms $\mathscr{M} = \{\mathscr{M}_1, \ldots, \mathscr{M}_m\}$ are sequentially performed on a dataset, and each $\mathscr{M}_i$ satisfies ϵ_i differential privacy, $\mathscr{M}$ will provide $\sum_i \epsilon_i$-differential privacy.*

Differential privacy is immune to post-processing. Applying an arbitrary data-independent mapping function f to a differentially private mechanism $\mathscr{M}$ does not influence its privacy guarantee.

Lemma 3.4. *(Post-processing [37]) Suppose $\mathscr{M} : D \to \mathscr{R}$ is a (ϵ, δ)-differentially private mechanism. Let $f : \mathscr{R} \to \mathscr{R}'$ be an arbitrary randomized mapping. Then $f \circ \mathscr{M} : D \to \mathscr{R}'$ is (ϵ, δ)-differentially private.*

Theorem 3.5. *The parameter updating mechanisms ($\mathscr{M}_3$, $\mathscr{M}_4$, $\mathscr{M}_5$) satisfies $(\epsilon_L + \epsilon_E)$-joint differential privacy.*

Proof. The mechanism $\mathscr{M}_3$ has each client add Gaussian distributed noise to their data θ_i, which can be thought of as using a Gaussian mechanism on a disjoint partition of a database $\{\theta_i\}^m$. According to the theory of parallel composition (Lemma 3.2), the sequence $\{\hat{\theta}_1, \ldots, \hat{\theta}_m\}$ satisfies ϵ_L-differential privacy.

In mechanism $\mathscr{M}_4$, the server computes $d_i\hat{\theta}_i$ using the noisy parameters $\hat{\theta}_i$ provided by each client, which are post processing operations on an ϵ_L-differentially private computation $\hat{\theta}_i$. According to the theory of post-processing (Lemma 3.4), $\{d_j\hat{\theta}_j\}^m$ satisfies ϵ_L-differential privacy. The server employs an exponential mechanism to select high-quality parameter aggregations, which is based on ϵ_L-differentially private $\{d_j\hat{\theta}_j\}^m$. According to the theory of post-processing (Lemma 3.4) and the theory of sequential composition (Lemma 3.3), this process satisfies $(\epsilon_L + \epsilon_E)$-differential privacy.

Mechanism $\mathscr{M}_5$ has each client update their parameters θ_i as a function of their private information θ_i and an $(\epsilon_L + \epsilon_E)$-differentially private signal (i.e., the result of the exponential mechanism). According to the billboard lemma (Lemma 4.1), the results $\{\theta_i\}^m$ satisfy $(\epsilon_L + \epsilon_E)$-joint differential privacy. □

3.5 Experimental results

3.5.1 Experimental setup

The two datasets selected were the MINST [75] and CIFAR-10 [73] datasets. The MINST dataset contains 60,000 training images and 10,000 test images of handwritten numerical digits. Each image is grayscale and normalized to 28×28 pixels. The CIFAR-10 dataset consists of 60,000 32×32 color images spanning 10 classes; 50,000 of them are training images and 10,000 are for testing. With the MINST dataset, we trained a CNN with two convolution layers, and two fully connected layers with dropout regularization. The network for the CIFAR-10 dataset comprised six convolution layers with ReLU units and maxpooling layers. The learning rate for both networks was 0.001.

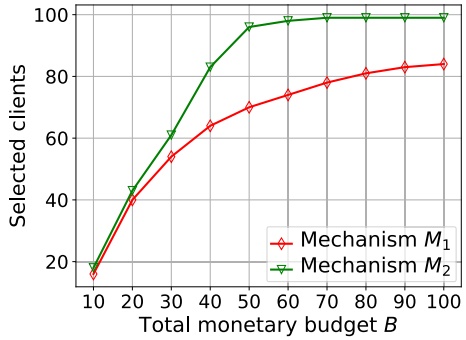
As a baseline, we used the traditional parameter aggregation method that is common to federated learning, where the server simply computes a weighted average of the parameters submitted by clients. We also included one adversarial client whose aim was to skew the global model via a dirty-label attack [7]. Hence, with the MINST dataset, the adversary changes the labels of all its training images to the digit 2. With the CIFAR-10 dataset, it changes all the labels to cat. For all other clients, the data is uniformly sampled from different classes. Each client perturbs their data locally using the Gaussian mechanism. The privacy budgets used were set to $\epsilon = \epsilon_L = \epsilon_E$.

3.5.2 The effectiveness of the proposed methods

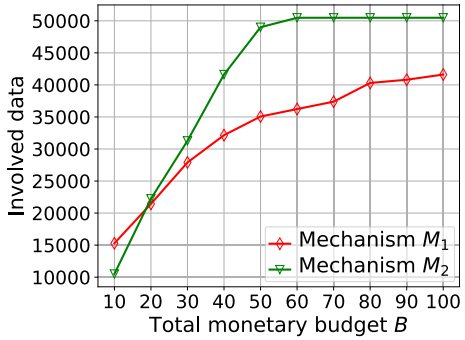
The first simulation was designed to examine how differing privacy budgets influence client selection and the incentive to participate. Here, we randomly generated the client's private costs in the range of [1,1000], and the amount of data they hold is in the interval [1,1000]. Different distributions of client's cost will not influence the game-theoretic properties of proposed mechanisms, thus we only limited the range of clients' private costs here. We also varied the server's total monetary budget from 10 units to 100 units.

Figure 3.2(a) shows the change in the number of clients selected with respect to the server's total monetary budget. In each simulation, $n = 100$ clients submit bids to participate in the training. Initially, a large number of clients were selected because the higher the monetary budget, the more clients the server can attract. In fact, with a sufficiently large monetary budget, the server was able to involve almost every client in the training, and so, at higher budget levels, the number of clients selected did not change much.

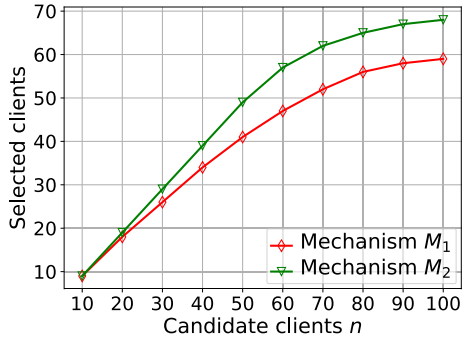
Figure 3.2(b) shows the amount of data provided by the selected clients. The results show similar tendencies to Figure 3.2(a). In addition, mechanism $\mathscr{M}_2$ (Algorithm 2) is designed with the aim of maximizing the amount of training data. As a result, it converges faster than the envy-free $\mathscr{M}_1$.



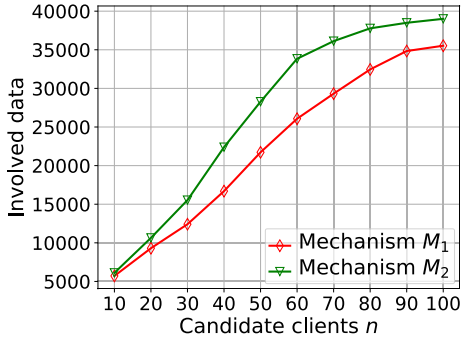
(a) The number of selected clients vs. the server's monetary budget



(b) The number of involved data vs. the server's monetary budget



(c) The number of selected clients vs. the number of candidate clients



(d) The number of involved data vs. the number of candidate clients

Figure 3.2: The performance of the method with various server budgets and numbers of candidate clients.

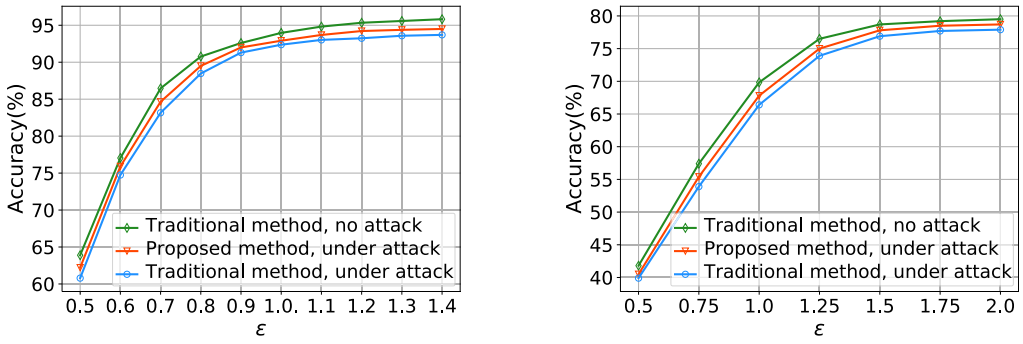
Figure 3.2(c) shows the change in the number of selected clients when the number of candidates submitting bids changes, where the server's monetary budget is fixed. The results show that both $\mathscr{M}_1$ and $\mathscr{M}_2$ selected more clients as the number of candidates increased. However, because the total monetary budget is fixed, there came a sort of saturation point where it was simply impossible to add more clients given that each has a base cost of more than 0. A similar tendency can also be observed in Figure 3.2(d), which shows the amount of data provided by clients with respect to the number of candidates. When the number of candidates reaches 90, the amount of data stabilized at around $35,000$ units for $\mathscr{M}_1$ and $39,000$ units for $\mathscr{M}_2$. Notably, because $\mathscr{M}_2$ works to solve a social

welfare maximization problem (Equation 3.4), it always incorporated more training data and selected more clients.

3.5.3 The proposed methods vs traditional methods

Our next experiment involved a comparison between the proposed method and traditional method, both under attack, with varied privacy budget. The accuracy of traditional method without attack was used as a baseline. This simulation included $m = 20$ clients, one of which was an adversary in the two attack scenarios. The data was evenly distributed among clients and performance was assessed in terms of the global model's accuracy. The results are shown in Figure 3.3.

The result patterns were similar for both datasets. With a small privacy budget, the clients had to inject a good deal of noise, which impacted accuracy. But, as the privacy budget increased, less noise was needed and accuracy improved. The "no attack" method was obviously the most accurate, followed by our method under attack, and then the traditional method. It is worth pointing out that once the privacy budget reaches a certain level, noise is no longer the main factor influencing accuracy; thus, the curves flatten. However, even at this point, our method was still more accurate than the traditional method.



(a) Accuracy of the global model vs. privacy budget, MNIST dataset.

(b) Accuracy of the global model vs. privacy budget, CIFAR-10 dataset.

Figure 3.3: A comparison between our method and traditional methods.

In Figure 3.3, we observe that the malicious client does not cause a serious decrease in the global model's accuracy. This is because the malicious client only changes the label of one class in its local training. In addition, the malicious client cannot have access to

the number of training data used by other clients, its update is more likely to be detected directly by the stealth metrics of the server.

3.5.4 Model accuracy with privacy budget

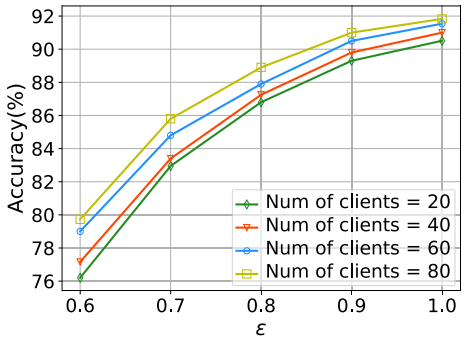
Figure 3.4(a) and Figure 3.4(b) show the performance of the method with different privacy budgets. With a larger privacy budget, each client injects less noise into the submitted parameters. As a result, the accuracy of the model increases. Similar variation tendencies can also be observed in Figure 3.3(a) and Figure 3.3(b). Besides, with a small number of clients, involving more clients in the training reduces the impact of the adversary, which also improves model accuracy.

However, we note that the privacy budget ϵ should be set carefully. On the one hand, a smaller ϵ means that any single client has less influence over other clients' data, which makes the learning framework more robust to adversarial manipulations. On the other hand, a small ϵ also flattens the probability distribution generated by the exponential mechanism, and further influences the model's performance. Therefore, although smaller ϵ protects the client's data privacy better, introducing more randomizations in the exponential mechanism impacts the model's accuracy. Hence, it is important to make the right tradeoff between robustness and performance for different situations.

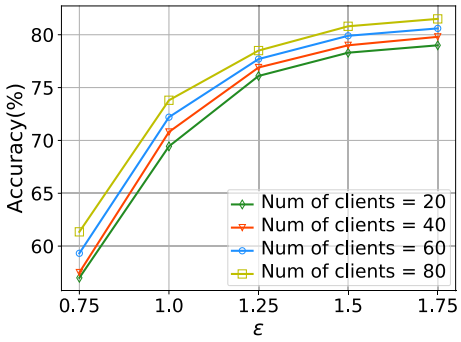
3.5.5 Model accuracy and the number of clients

Figures 3.4(c) and 3.4(d) show the model accuracy with different numbers of clients. In Figure 3.4(c), we see that the accuracy of the global model increases with an increase in participating clients. This is because involving more benign clients in the training process reduces the impact of the adversary. In other words, incorporating more clients increases the number of benign clients within each candidate groups, and the weighted average of parameters in each candidate group becomes more accurate. By contrast, Figure 3.4(d) shows that the accuracy of the model first increases but then flattens (and even decreases) as more clients participate. The reason is that the adversarial client is only in one candidate group, and the score of this group is at most as good as that of the other groups. Therefore, the exponential mechanism is making a selection over an almost uniform distribution when the number of clients is very large, and thus the global model's accuracy fluctuates.

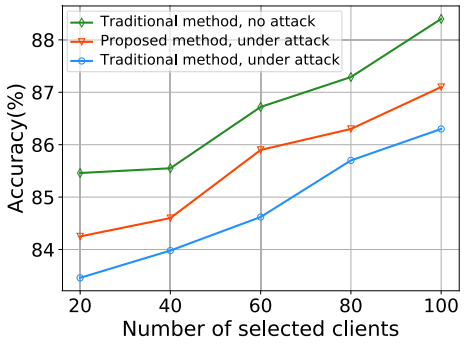
What Figure 3.4(d) demonstrates is that involving more clients is not always beneficial to the training, because incorporating more clients cannot continuously improve the



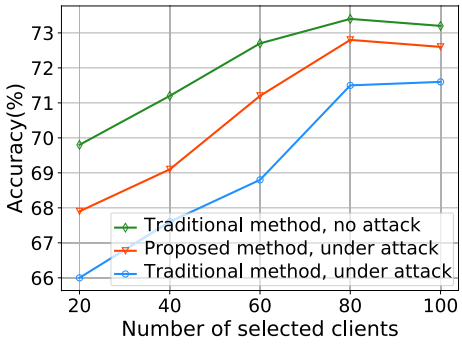
(a) The accuracy of the global model vs. the privacy budget, MNIST dataset.



(b) The accuracy of the global model vs. the privacy budget, CIFAR-10 dataset.



(c) The accuracy of the global model vs. the number of client, MNIST dataset.



(d) The accuracy of the global model vs. the number of client, CIFAR-10 dataset.

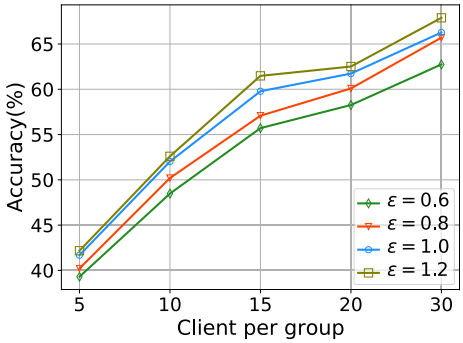
Figure 3.4: The change in global model accuracy with various privacy budgets and numbers of clients.

global model's accuracy, but recruiting more clients in the training requires the server to spend more monetary budgets. Thus, again, a balance is needed – this time, between the number of participating clients and model accuracy.

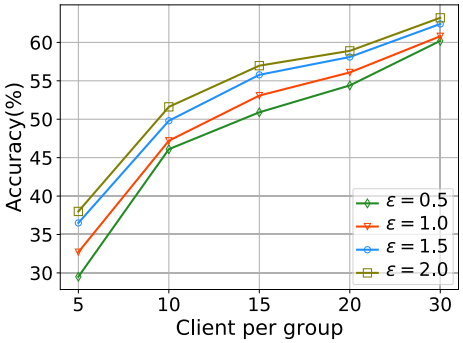
3.5.6 Model accuracy and the server's grouping strategy

Figure 3.5 presents the influence of the server's grouping strategies on model accuracy. In Algorithm 4, the server applies leave-one-out strategy to construct different client groups. In this experiment, we took $m = 60$ clients including 1 adversary, and had the server randomly partition them into groups where each contains 5, 10, 15, 20 and 30 clients. From the figures for both datasets, we can see that the model accuracy increases as the number of clients in each group increases, but the accuracy in these scenarios are lower than that obtained using the leave-one-out strategy. This is because averaging

over more clients reduces the variance of the noise on the averaged value.



(a) The accuracy of the global model vs. clients per group, MNIST dataset.



(b) The accuracy of the global model vs. clients per group, CIFAR-10 dataset.

Figure 3.5: The change in global model accuracy with various numbers of clients allocated to each group.

Note, however, that this observation only holds when each client has a similar amount of training data. In our framework, the amount of data used in the training process is considered as the weight of each client in parameter aggregation. So, if the adversary locally trains the model with much more data than the other clients, the impact of the poisoned dirty label data might be more substantial.

3.5.7 Discussion

In these experiments, we considered the influence of exactly one adversarial participant in the training process. With the traditional method, the adversary was able to cause a $4\% \sim 5\%$ decrease in model accuracy via its dirty-label attack. However, under our joint differential privacy regime, the adversary's impact was limited to a $1\% \sim 2\%$ decrease.

Our experiments validate the effectiveness of the privacy budget as a means of controlling both the noise level in privacy protection and the influence of an adversary. We also examined the game-theoretic properties of the proposed mechanisms, and demonstrated the effectiveness of the framework in terms of privacy budget, the number of clients and the server's group strategy.

Similar to classic differential privacy, an appropriate tradeoff between accuracy and privacy should be carefully considered when setting privacy budgets in a joint differential privacy setting. Having a smaller ϵ reduces the influence of individual clients, hence enhancing the robustness of the framework against adversarial manipulations.

Conversely, increasing the amount of randomization in the framework also negatively affects the performance of the model.

3.6 Summary

This chapter proposes a robust federated learning framework that addresses the practical problem of incentivizing clients to participate in federated learning. As a solution, this chapter suggests two novel, game-theoretic mechanisms that formulate client selection as an auction game. The clients report their costs as bids and the server uses different payment strategies to maximize its objectives. The framework supports different objectives, such as social welfare maximization, cost minimization. Of the two mechanisms, the first is envy-free, dominant strategy truthful, and individual rational. The second is also truthful and individual rational, but also allows the server to select clients so as to achieve its own specific optimization goal. The overall framework is jointly differentially private, which limits the impact of adversarial clients. We have provided detailed proofs and analyses of the algorithms' properties, along with an examination of the privacy budget. This analysis serves as a parameter-setting guide to help the server achieve the desired level of client privacy and robustness against adversaries. Additionally, we have conducted simulations with real-world datasets to validate the framework's performance. The results demonstrate that under our scheme, an adversarial client has little impact on the global model's accuracy, and the accuracy is better than it is under traditional frameworks.

This chapter inspires us that we can explore more complex federated learning scenarios where more than one adversarial client has been selected to participate in the training. At present, joint differential privacy limits the impact of a single adversarial client. With more adversarial clients involved, it is also possible to explore ways to extend and/or relax the current definition of differential privacy, and develop new federated learning frameworks based on these revised definitions. In addition, designing one or more client selection mechanisms based on different kinds of auctions is also possible.

CHAPTER 4

JOINT DIFFERENTIAL PRIVACY AND POTENTIAL GAME: IMPROVING THE DATA QUALITY FOR FEDERATED LEARNING FRAMEWORK

When it comes to privacy-preserving machine learning problems, federated learning has emerged as a viable solution. Despite all of its successes, the framework has several drawbacks. First, traditional frameworks assume that all clients want to improve model accuracy and so participation is voluntary. However, in reality, clients usually want to be appropriately compensated for the data and resources they will need to commit to the training process before contributing. Second, today's frameworks allow clients to perturb their parameter updates locally, which introduces a great deal of noise to the trained model and can seriously impact model accuracy. To address these concerns, this chapter develops a private reward game that incentivizes clients to contribute high-quality data to the training process. The game converges to a Nash equilibrium under the guarantee of joint differential privacy, and each client maximizes their reward following an equilibrium strategy. The noise injected into the model is reduced by introducing a centralized differential privacy model that aggregates the parameters and compensates clients via a data trading market. Experimental simulations show the rationales behind and effectiveness of the proposed game approach. Additionally, we present comparisons between different training models to demonstrate the performance of the proposed approach in real-world scenarios.

4.1 Introduction

Federated learning [48] is a promising distributed machine learning paradigm that has been playing a significant role in providing privacy-preserving artificial intelligent solutions. The core idea of federated learning is to train a machine learning model on separate datasets that are distributed across different devices or parties. In most federated learning schemes, only the model's parameters or the gradients are shared, each client's data are kept invisible and anonymized to other parties. Therefore, the model can be trained without disclosing clients' raw data and, to a certain extent, their data privacy can be preserved.

Yet, despite these benefits, federated learning still faces several critical challenges. The first and foremost challenge corresponds to a client's incentive to participate in the training – a detail which is often overlooked in a framework's design [164]. And, even if a client does want to participate in the training process, they may need some inducement to ensure they contribute actively and behave reliably throughout [65]. Most developers consider the final trained model to be enough incentive to guarantee conscientious participation. However, research has demonstrated that this is far from enough [78, 128, 131, 163]. In fact, being a part of a federated learning scheme demands a great deal of overhead on the part of the client, and so their excessive consumption of time, energy, bandwidth, etc., may need to be compensated. More importantly, nowadays, data are regarded as valuable private assets, that may not necessarily be given away for free as easily as they once were. Hence, the traditional assumption that all clients will voluntarily wish to participate in federated learning may be over-optimistic. From this perspective, a federated learning scheme that incorporates an incentive mechanism may be very beneficial.

In addition to the participation problem, the tradeoff between privacy and data utility is another long standing problem that has not yet been adequately addressed. For example, even though federated learning operates on the principal that only the parameter or gradient updates are transferred off the local device, a client's private information can still be compromised by inference [5, 7, 124]. A popular method of preventing this problem is for the client to add local randomizations at their end to obscure the data. This affords differential privacy [1] , but the amount of noise injected this way is generally much greater than that of a centralized differential privacy model [6, 16], and large amounts of noise seriously impact the performance of the model. As such, new technologies need to be explored to address this issue.

Our solution to both these problems is a reward game that not only incentivizes clients to participate in federated learning training schemes but also incentivizes them to contribute high-quality data when they do. Data quality is a major concern in federated learning as updates trained from low-quality data can directly influence the performance of global model [169]. A client may hold different qualities of data for the same class or domain – for example, different accuracies for a statistic or different resolutions of an image – and they may not voluntarily contribute their high quality data unless provided with sufficient compensation for doing so. Not only might using that data demand extra time and/or resources, but, today, high-quality data are valuable assets on the data market [100]. Thus, in our game, clients dynamically decide the quality of data they wish to contribute. The game keeps the level of quality contributed by each client confidential, and finally converges to an approximate Nash equilibrium by maximizing each client's reward. The aim of forming a Nash equilibrium is to incentivize clients to contribute their high-quality data in the local training, and thus to improve the global model's performance. Without the equilibrium, clients may be unwilling to use their high quality data. The Nash equilibrium is established using rewards, which is a reasonable way to incentivize clients' contributions. Besides, each client also decides whether to use a centralized differential privacy model or whether to add noise locally, with additional compensation should a client choose to use the central model. By evaluating the costs resulting from a potential privacy breach, they can decide which option has the better cost-benefit ratio.

The contributions of this chapter are summarized bellow.

- We introduce a private reward game that allows clients to dynamically decide the quality of their training data without disclosing the level of quality they provided. The game converges to an approximate Nash equilibrium under a guarantee of joint differential privacy.

- We develop the idea of "money for privacy" to improve the performance of the global model – a concept borrowed from private data trading.

- We provide theoretical proofs and analyses of our proposed game, which satisfies desirable game-theoretic properties and rigorous privacy requirements.

- We conduct experimental simulations that show the behaviors and utility change in clients and also demonstrate the rationale behind the reward game. Further

experiments with real-world datasets show the outstanding performance of the proposal when incorporated into an actual federated learning scheme.

In the rest of this chapter, we first introduce basic background knowledge of potential games and then detail the private voting mechanisms for data quality improvement. Theoretical analyses and experimental results are also given.

4.2 Preliminaries

4.2.1 Potential game and Nash equilibrium

In this chapter, we take advantage of potential games [94] to model the interaction between clients. Denote N the set of players, and let A be the possible action profiles and u be player's utility. A game $\mathscr{G} = (N, A, u)$ is an exact potential game if there exists a potential function $\Phi : A \to \mathbb{R}$ such that, for every $a_i \in A_i$ and $a_{-i} \in A_{-i}$, function Φ satisfies

$$\Phi(a) - \Phi(a_i', a_{-i}) = u_i(a) - u_i(a_i', a_{-i}), \tag{4.1}$$

where a_{-i} represents the vector a except for the ith element.

Potential functions track the change in utility when a player deviates from one strategy to another. In other words, if a player i unilaterally changes their strategy from a_i to a_i', a potential function will increase or decrease by the same amount as the player would have obtained with u_i. For a continuous and differentiable utility function u_i over an action space A, a game $\mathscr{G}$ is an exact potential game if

$$\frac{\partial u_i(a)}{\partial a_i} = \frac{\partial \Phi_i(a)}{\partial a_i}, \quad \forall i \in [n], a \in A. \tag{4.2}$$

Potential games are a special class of games with the desirable property that there exists a Nash equilibrium.

Theorem 4.1. *(The existence of a equilibrium [94]) Every finite potential game has a pure-strategy Nash equilibrium.*

In other words, every action profile $a^* \in A$ that maximizes the potential function $\Phi(a)$ is a Nash equilibrium of game $\mathscr{G}$.

One way to compute the Nash equilibria of potential games is to employ the best response dynamic [141]. Given an action profile a, the best response of player i is

$$BR_i(a_{-i}) = \arg\max_{a_i \in A_i} u_i(a_i, a_{-i}). \tag{4.3}$$

In the best response dynamic, players iteratively make their best response following a predefined order. For finite potential games, all executions of the best response dynamics terminate when Nash equilibrium is reached.

Definition 4.1. *(Nash equilibrium [102]) An action profile a^* is a Nash equilibrium if each a_i^* is the best response of a_{-i}^*. That is, $\forall i \in [n]$, $a' \in A$,*

$$u_i(a_i^*, a_{-i}^*) \geq u_i(a_i', a_{-i}^*). \tag{4.4}$$

In this chapter, we focus on games of incomplete information, where the *ex-post* Nash equilibrium is considered.

Definition 4.2. *(Ex-post Nash equilibrium [24]) Given a set of functions, $g_i : \mathcal{T} \rightarrow A$, which map client types to actions, the $\{g_i\}$ forms an η-approximate ex-post Nash equilibrium if, for every type vector $\tau \in \mathcal{T}$, and every action $a_i' \in A_i$,*

$$u_i(g_i(\tau_i), g_{-i}(\tau_{-i})) \geq u_i(a_i', g_{-i}(\tau_{-i})) - \eta. \tag{4.5}$$

4.2.2 Differentially private counter

The key tool to achieving differential privacy in our framework is the private stream counting technique proposed in [15, 36]. Given a bit stream $\sigma = \{\sigma_1, \ldots, \sigma_T\} \in \{-1, 0, 1\}^T$, a counter $\mathscr{M}(\sigma)$ releases an approximation to $C_\sigma(t) = \sum_{i=1}^{T} \sigma_i$ at every step t. We use $Binary(\sigma, \epsilon)$ to denote the binary mechanism proposed in [15]. The mechanism is differentially private and satisfies the following accuracy guarantee.

Theorem 4.2. *(Binary mechanism [15]) The mechanism $Binary(\sigma, \epsilon)$ is ϵ-differentially private with respect to a single bit change in the stream that has length T, and with probability at least $(1-\beta)$, it satisfies $|\mathscr{M}(\sigma)(t) - C_\sigma(t)| \leq \alpha$ for*

$$\alpha = \frac{2\sqrt{2}}{\epsilon} \ln(\frac{2}{\beta}) \log(T)^{\frac{5}{2}}. \tag{4.6}$$

We will use differentially private counters to track clients' votes on different data quality levels, and thus their private quality information can be preserved.

4.2.3 Problem definition

This chapter addresses three issues associated with federated learning. First, we develop game-theoretic solutions to incentivize clients to contribute their high-quality data to the

training process, where the clients' truthful behaviors form an *ex-post* Nash equilibrium. Second, we aim to protect the information pertaining to the data quality each client contributed. This is because some clients may collude to infer the sensitive information of others by analyzing their outcome of the game. Hence, we need to find a mechanism that computes the desired equilibrium and is also robust to colluded inference. Finally, the framework we propose must protect each client's data privacy, while maintaining high model accuracy. This will require a new payment strategy and a new parameter aggregation method that motivates clients to submit their updates in a less noisy way.

4.3 The game-theoretic federated learning framework

4.3.1 Design rationale and mechanism overview

The proposed game-theoretic federated learning framework is depicted in Figure 4.1, which consists of a private game module and a private learning module.

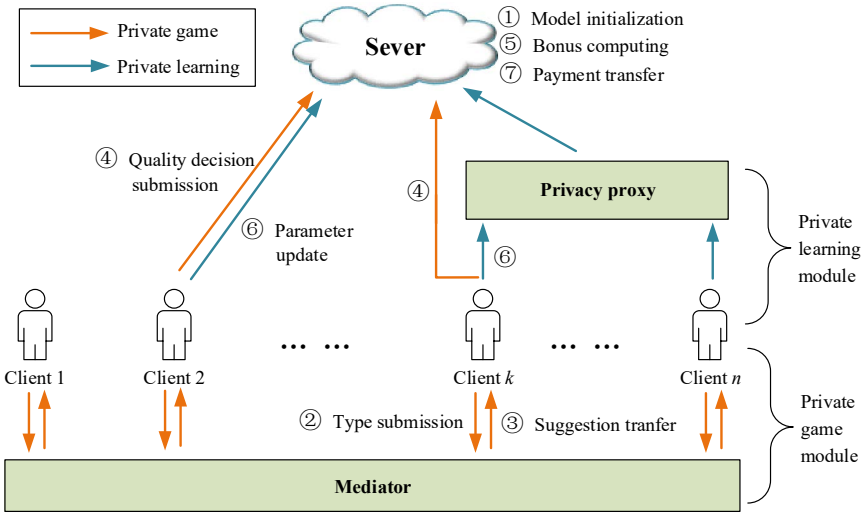


Figure 4.1: Game-theoretic federated learning framework

4.3.1.1 Design rationale of the game

In the proposed framework, there are n clients participating in the training, denoted as $\{c_1, \ldots, c_n\}$. Each client has a private database D_i that will be used to collaboratively train

a machine learning model according to the schedule of a central server S. Considering the quality of each client's data, it is assumed that the data in each database D_i has m classes, denoted as $\{p_1^i, p_2^i, \ldots, p_m^i\}$. Within each class p_j^i, the data are partitioned into l levels $\{q_{j1}^i, q_{j2}^i, \ldots, q_{jl}^i\}$ according to quality. In practice, the quality of data is specified by the server in advance according to the structure of training data. Before the game starts, the server can publish thresholds for each of the l quality levels so that this information can form a common knowledge between the server and clients. For example, in a handwritten digit database, there are $m = 10$ classes (i.e., 0 to 9), and the quality of the images within each class can be determined by image resolution, sharpness, etc. Without loss of generality, we assume the data classes $[p_i]_{i=1}^m$ and quality levels $[q_i]_{i=1}^l$ are known to the server, but the specific quality levels of a client's data are private.

Ideally, the server S hopes that all clients use high quality data. However, these assets are more valuable so, given a fixed reward, a client would always prefer to contribute the lowest quality data possible. Thus, as an incentive to use better data, the server S convenes a reward game, where the reward for a specific quality of data is a monotonically decreasing function parameterized by the number of clients providing it. In this game, the reward rule provides enough incentive – that is, all clients will get low reward if all only use low-quality data. Consequently, each client will at least consider using higher quality data for their local training.

Unfortunately, in federated learning, clients do not have contact with each other; they may not even be aware of each other's existence. Therefore, the incentive game also makes it a challenge for clients to decide what to play – one client's reward is not solely determined by their own choice, but also on the strategies of others. To address this challenge, in the game module, a mediator is introduced to collect the information of all clients and make suggestions as to what to do. Each client c_i can choose to use the mediator by submitting the quality level of data they have. The mediator then generates suggestions over which quality of data to use to the clients who opt-in to the service. Of course, clients are free to ignore the mediator and play their own strategies, or to report (possibly fake) quality levels and then not follow the proposed suggestion. Clients' good behaviors form an *ex-post* Nash equilibrium of the game – *good behavior* being that each client truthfully reports their quality level to the mediator and then follows its suggestion. Besides, the data quality, which is regarded as private information about a client's data, is also protected during the course of the game.

4.3.1.2 Overview of the mechanism

The work flow of the proposed frameworks is as follows.

- First, the server S initializes the model to be trained and publishes necessary information about this model to all clients, e.g., the number of parameters, the network structure, and so on (Step ①).
- Second, each client decides whether to use the mediator. If yes, then client c_i submits its type τ_i to the mediator (Step ②), which contains $c_i's$ data quality level information. The mediator will return its suggestion to client c_i based on the types it collected (Step ③).
- With the equilibrium strategy, each client c_i submits their final decision on the data quality used to the server S (Step ④). The server computes the rewards r_i according the predefined reward rule, and publishes the bonus b according to its total budget (Step ⑤).
- Based on the reward r_i and the bonus b, each client decides whether to use the centralized proxy to protect their data privacy, or whether to employ a local perturbation to the trained parameters it submits (Step ⑥).
- The server S stops training once it arrives at satisfactory model parameters. It then pays corresponding rewards and bonuses to each client c_i (Step ⑦).

In the game module, the server proposes a training requirement and recruits clients who are interested in participating in it. The mediator is jointly determined by the server and clients. After that, the reward game runs to produce an approximate equilibrium, where each client decides the quality of data they would use in local training.

In the private learning module, the server S provides two ways for the clients to submit their trained parameters. First, clients can locally perturb their parameters by adding noise sampled from the Gaussian distribution, and then submit them to the server. Second, each client can submit their trained parameters directly to a centralized differential privacy proxy, which runs by the server S. The server *promises* that the proxy will employ a centralized Gaussian mechanism to protect the privacy of its received data. The first way is suitable for the conservative clients who believe the server is untrustworthy, while the second way is preferred by the clients who believe the server to some extent, or the clients who get enough reward that can overweight their expected potential privacy loss. Of course, the server S will offer extra bonus b for the clients who

choose the second way. The learning process ends when the aggregated model reaches a satisfactory threshold.

In this chapter, we assume that the clients are honest and no adversarial clients are involved. Namely, each client will follow the result of game and use corresponding quality of data in their local training. Otherwise, as the server cannot monitor client's local training, malicious clients would claim a contribution of high quality data but actually use low quality ones during training process.

4.3.1.3 Discussion of the proposed framework

The location of the mediator. In the proposed framework, it is natural to ask where to place the mediator – now that the server is not trustworthy, why should clients believe a mediator, which looks like a new trusted third party? In fact, the primary purpose of the mediator is to schedule the reward game and provide suggestions that form a Nash equilibrium. In practice, the mediator does not have to be an independent third party, which could pose new privacy risks. Simply as the GPS navigation, where users can choose to follow the suggested route or drive independently, the mediator can be thought of more as a service provided when players sign up for a federated learning task – they can send their data quality information to it and receive suggestions, or they can choose to ignore it. Without the coordination of the mediator, players' may not know what is the optimal strategy, and the incomplete information game may fail to reach a Nash equilibrium. The mediator enables players' behaviors to be coordinated and thus federated learning can benefit from better data quality.

The role of the mediator. Throughout this chapter we assume that client's data quality is sensitive information, and thus the mediator is independent of the server. If the quality information is not regarded as privacy, the mediator could be embedded in the server, and the proposed incentive mechanism still works. In practical scenarios such as vehicle networks, the mediator could be organized by clients themselves with further assumptions such as clients can communicate with each other. For example, in a training of shopping recommendation model, each client (e.g., shop) can submit the quality of data they would use to the server (i.e., the mediator is maintained by the server) or to a trusted proxy (i.e., the mediator is maintained independently of the server, e.g., by a regulator). Alternatively, if clients could communicate with each other, they can share their quality information and perform the reward game by themselves (i.e., the mediator is maintained by clients). As for privacy issue, the mediator schedules the reward game under differential privacy guarantee and returns an approximate Nash equilibrium,

client's privacy will not be leaked unless the mediator itself corrupts.

The interpretation of the reward. In real-world scenario, the reward could be simply money, which is spent by the server to improve the quality of training data. In addition, the reward could be also related services provided by the server. For example, the clients who contribute high-quality data will get high-quality aggregated parameters. The reward is a key factor that incentivizes client's good behaviors. However, the implementation of reward must also follow economic principles. For example, excessively use of reward not only causes a huge cost for the server, but also indirectly lowers the value of high quality data. In the proposed game, the server can set a reward budget, and dynamically adjusts the total amount of reward according to client's behaviors throughout the game. It is a commonly adopted strategy in data trading market.

4.3.2 The private game module

4.3.2.1 Reward game formulation

Consider a reward game $\mathcal{G} = (N, A, r)$ that contains n players, where $N = \{c_1, c_2, \ldots, c_n\}$ is the set of players (clients), A_i is the client c_i's possible action space, and r is the reward function. $A = A_1 \times \cdots \times A_n$ is written to denote the union of all action spaces, and $a = \{a_1, a_2, \ldots, a_n\} \in A$ is an action profile. Recall that the data in each database D_i has m classes, each class is partitioned into l levels according to the data quality. Therefore, each client has a type τ_i from type space $\mathcal{T} = \{q_{jk}\}^{m \times l}$, $j \in [m], k \in [l]$, where q_{jk} corresponds to the qualities of data c_i has. For example, suppose the data in each D_i has two classes of images {dog, cat}, each class of image is partitioned into three quality levels q = {high, medium, low}. A client c_i may have q_1 = {high, medium} qualities of images for the dog class and q_2 = {medium, low} qualities of images for the cat class, this information is referred as its type τ_i. The action set $A_i = q_1 \times q_2$ of client c_i is all possible qualities of data it may contribute.

The monotonically decreasing reward function $f : \mathbb{N} \to [0, 1]$ for each quality level q_{jk} maps the number of clients at q_{jk} to a value between 0 and 1. Clients will wish to maximize their reward, which is calculated as the sum of the rewards for the quality levels they choose to use. We denote the number of clients on quality level q_{jk} to be n_{jk}. For a client of type τ_i and a given action profile a, we have

$$r : \mathcal{T} \times A \to [0, m], \quad r_i(\tau_i, a) = \sum_{q_{j,k} \in a_i} f(n_{jk}). \tag{4.7}$$

Nash equilibria are always desired solutions to a game because it is easy to reason about clients' behaviors from a stable point. However, in federated learning, clients do not know the types of other clients, so the reward game is in fact a game of incomplete information. As a result, Nash equilibria cannot be established without knowing all the clients' types. To address this issue, we introduce a mediator and compute Nash equilibria of the complete information game induced by clients' types in a incomplete information setting.

Formally, a mediator $M : (\mathcal{T} \cup \{\varnothing\})^n \to (A \cup \{\varnothing\})^n$ is an algorithm that maps the clients' reported types to its suggested actions, where $\varnothing$ represents clients who decline to use the mediator and, thus, the mediator does not provide any suggested action for them. $\mathcal{G}_M$ denotes the reward game augmented with a mediator M. In a $\mathcal{G}_M$ game, the clients have several options. They can choose not to use the mediator, i.e., report $\varnothing$, and simply select their own actions. Alternately they can report a (possibly fake) type to the mediator and receive a suggested action a_i. An arbitrary mapping function $g : A_i \to A_i$ can be used to decide whether to follow the mediator's suggestion. Formally, the action space of the mediator game $\mathcal{G}_M$ is

$$A_M = \{(\tau_M, g) : \tau_M \in \mathcal{T} \cup \{\varnothing\}, g : (A \cup \{\varnothing\}) \to A\}. \tag{4.8}$$

Let G be the set of clients' all possible mapping functions, and let $g(a) = (g_1(a_1), \ldots, g_n(a_n))$. The reward function in the mediator game $\mathcal{G}_M$ then becomes

$$r_i(\tau_i, (\tau', g)) = \mathbb{E}_{a \sim M(\tau')}[r_i(\tau_i, g(a))]. \tag{4.9}$$

That is, the reward in the mediator game $\mathcal{G}_M$ is the expected reward in the original game $\mathcal{G}$ given the actions players play as a function of the suggestions made by the mediator. The reason we use expected reward is that the mediator will introduce randomizations when it makes an suggestion, and we assume that the clients are risk-neutral.

In this chapter, we consider a large reward game. In other words, each unilateral move of some clients $j \neq i$ can have only a small effect on the reward of client i. Formally,

$$\Delta\mathcal{G} = \max_{j \in [m], k \in [l]} |f(n_{jk} + 1) - f(n_{jk})|. \tag{4.10}$$

The largeness assumption is without loss of generality as the number of clients in federated learning is usually large. $\Delta\mathcal{G}$ guarantees that no client can significantly decrease other clients' reward, and thus clients do not need to make frequent best response moves after they have made one.

4.3.2.2 Computing equilibrium with joint differential privacy

We now consider how to compute Nash equilibrium with a reported type profile. In the mediator game, the data quality each client will contribute is regarded as private information. Therefore, the process of computing equilibrium must include a privacy guarantee. Classic full differential privacy imposes a rigorous probabilistic limitation based on the whole set of clients, which means the equilibrium is almost independent of each client's report. Hence, we instead use joint differential privacy, where client c_i's reported type actually influences the equilibrium strategy. A jointly differentially private mechanism protects a client's privacy against the arbitrary collusion of other clients. That is, even if some groups of clients collude together, they still would not be able to deduce the data quality that other clients are contributing.

The rationale behind the equilibrium computation mechanism is quite intuitive. Suppose the game begins at a random status, where each client chooses one quality q_{jk} for each data class $k \in [m]$. Then in each round, clients are able to evaluate their current rewards r_i using the count of the number of clients choosing each quality, namely $r_i = \sum f(n_{jk})$. Based on the current counts and reward, each client can adjust their strategy and make their current best response. The adjustment is made iteratively following a predefined order, and the mechanism will finally converge to a Nash equilibrium.

Theoretically, if clients play the most recent moves chosen during the course of the mechanism, each will be able to figure out what action they should take by just looking at the counts. The reason is that the counts not only specify the rewards they currently have, they also indicate whether or not they should change their action and which new action to take. Therefore, it is possible to implement the best response dynamic using a set of counters – one for each data quality – each indicating the number of clients on different qualities.

The privacy of each client is protected by employing a private counting technique. Here, noisy counters are introduced to maintain a noisy version of the client counts for each quality. These counters keep track of bit streams that represents clients' dynamic decisions. In each iteration, if a client is moved to or away from a quality, the related counter receives a bit 1 or -1, otherwise the counter receives 0.

Formally, C_{jk} represents the private counter for tracking data quality q_{jk}, $j \in [m]$, $k \in [l]$. There are $m \times l$ private counters, each maintaining a stream of bits $\{e_i\} \in \{-1, 0, 1\}^t$ to record the number of clients contributing the corresponding data quality. In each round t, the best move is calculated for each client, then each counter C_{jk} is updated with its noisy counts $\hat{n}_{jk} = Binary(\{e_0, \ldots, e_t\}, \epsilon)$, where $Binary$ is the binary mechanism. Rewards are

then evaluated for each client based on the noisy counts. These are calculated as the sum of the rewards for m data qualities they are currently assigned to using:

$$\hat{r}_i(\tau_i, a) = \sum_{q_{j,k} \in a_i} f(\hat{n}_{jk}). \tag{4.11}$$

We assume that each move by a client will lead to at least an α improvement in their reward, as the following definition outlines.

Definition 4.3. *(α-noisy best response [119]) Given an action profile a of the reward game, an α-noisy best response a_i^* for client i with type τ_i is*

$$a_i^* \in \arg\max_{a_i \in A_i} \{\hat{r}_i(\tau_i, (a_i, a_{-i})) - \hat{r}_i(\tau_i, a) \geq \alpha\}. \tag{4.12}$$

The α-noisy best responses will be made for each client in rounds when its reward will substantially increase given noisy reward estimates. The parameter α influences the convergence of Nash equilibrium. Namely, the game requires more iterations to reach a steady point with a smaller α, which in turn introduces more noise to the equilibrium given a fixed privacy budget. Generally, the server needs to make a balance between the reward it spends and the precision of the equilibrium. Therefore, parameter α is determined by the server according to its specific requirement. The details of the mechanism are presented in Algorithm 6.

The algorithm first chooses feasible actions for each client based on their reported type and updates the bit streams for each private counter according to the initial allocation (Lines 2 – 6). It then computes the noisy counts for each quality level (Lines 8). After initialization, the best moves for each client are established iteratively and in a predefined order (Lines 9 – 23). If there is no α-best response for a client, each counter is updated with a bit 0 (Lines 13 – 15). If a client's reward could be improved by more than α by changing their choice to a different quality level, each private counter will be updated with a 1, −1, or 0, corresponding to whether the client is being moved to, away from, or sticking with the related quality level (Lines 17 – 18). $max(i)$ is used to keep track of the number of best responses made for each client c_i. If $max(i)$ exceeds a threshold, the mechanism halts and returns a fail (Lines 20 – 22). Otherwise, the algorithm returns its suggested actions for each client, which is the most recent move they made during the course of the algorithm.

4.3.3 The private learning module

The private learning module spans Steps ④~⑦ of the proposed framework. The private learning module addresses two issues: first, the privacy protection of client's trained

Algorithm 6 Jointly-DP equilibrium computation $\mathscr{M}_1$

Input: classes of data $[p_i]_{i=1}^m$, data quality levels $[q_i]_{i=1}^l$, clients' type vector.
Output: suggested auction profile a.

initialize $T = \frac{2mnl}{\alpha}$, $k = \frac{8m^2nl\Delta\mathscr{G}}{\alpha^2}$, $\epsilon' = \frac{\epsilon}{4km\log(T)}$.
for $i \in [n]$ **do**
  choose $a_i \in A_i$ for each client c_i at random.
  update the bit stream $\{e_i\}$ for each counter C_q.
  $max(i) \leftarrow 1$
end for
$a^t = (a_1, \ldots, a_n)$, $t \leftarrow n$
compute $\hat{n}_q^t \leftarrow Binary(\{e_i\}, \epsilon')$ for each $q \in [q_{jk}]^{m \times l}$.
for $t \in [nT]$ **do**
  for $q \in [q_{jk}]^{m \times l}$ **do**
    $t \leftarrow t+1$, $i \leftarrow (t \mod n) + 1$
    $a_i^t \leftarrow \alpha$-best response for c_i given previous $\hat{n}_q^{t-1}$.
    if a_i^t is NULL **then**
      update each C_q's stream by sending 0 to them.
      $a_i^t = a_i^{t-1}$
    else
      update each C_q's stream by sending 1 (-1 or 0) to them according to c_i's latest choice.
      $max(i) \leftarrow max(i) + 1$
    end if
    if $max(i) \geq k$ **then**
      return FAIL
    end if
  end for
  compute $\hat{n}_q^t \leftarrow Binary(\{e_i\}, \epsilon')$ for each q.
  assign $a_j^t \leftarrow a_j^{t-1}$ for $j \neq i$.
end for
return a_i^t to each client c_i for $i \in [n]$.

parameters in data submission; second, the computation of the extra bonus for clients who provide more accurate trained parameters.

Clients have two choices when they submit their trained parameters; both are protected by differential privacy. They can add noise to their parameters locally before sending them to the server. Alternatively, they can submit their data to a centralized privacy proxy, who will perform differentially-private parameter aggregation in a centralized manner. Obviously, it is in the server's interests that more clients update their parameters through a centralized way.

In Step ④, each client c_i sends its preferred data quality level to the server, which determines the reward r_i it can receive for the contribution of its data. In Step ⑤, the server further computes and publish an extra bonus b based on its total budget B. Each client who chooses to use the centralized differential privacy proxy will receive compensation b for its potential privacy loss.

In Step ⑥, each client decides which method to use according to the amount of money it will receive, namely $r_i + b$. As shown in Algorithm 7, if the total payment a client can receive is greater than the expected value of its compensation, the client simply submits its parameters for centralized differential privacy computation. Otherwise, if a client values its privacy more, it employs a Gaussian mechanism to perturb its parameters following a predefined privacy budget ϵ_L. Choosing the latter approach, clients will only receive a reward r_i, which is the total amount of money they can get throughout the entire federated learning.

Algorithm 7 Client's private data submission $\mathscr{M}_2$

Input: the trained parameter θ_i, the expected payment v_i of client c_i, bonus b, privacy budget ϵ_L for local perturbation, sensitivity $\Delta_2 f$, clipping threshold C.
Output: submitted parameter.

$\bar{\theta}_i \leftarrow \theta_i / \max\{1, \frac{\theta_i}{C}\}$
if $r_i + b \geq v_i$ **then**
    return θ_i
else
    sample $Z_i \sim \mathscr{N}(0, \sigma)$, where $\sigma = \sqrt{2 \ln \frac{1.25}{\sigma}} \cdot \frac{\Delta_2 f}{\epsilon_L}$
    $\hat{\theta}_i \leftarrow \bar{\theta}_i + Z_i$
    return $\hat{\theta}_i$
end if

To compute the extra bonus b, the server has to evaluate each client's private valuation about its privacy, namely, how much compensation is enough to incentivize clients who are inclined toward the centralized approach. Here, we choose the widely-used cost

evaluation method in private data commercialization [42], which provides a rigorous way to quantify a client's expected privacy cost under differentially private conditions. Suppose a client with type τ_i has a utility $u_i : O \to \mathbb{R}^+$ from a randomized mechanism M. If M satisfies differential privacy, by taking expectation of the inequality, we have

$$\mathbb{E}_{o\sim M(\tau)}[u_i(o)] \leq \exp(\epsilon)\mathbb{E}_{o\sim M(\tau'_i,\tau_{-i})}[u_i(o)]. \tag{4.13}$$

Together with the fact that $\exp(\epsilon) \approx 1 \pm \epsilon$ for small ϵ, we know that, for any utility function u_i of client c_i, its future utility in expectation will decrease by at most an additive factor $\epsilon\mathbb{E}(u_i)$. Therefore, it is enough to compensate each client ϵv_i for its expected privacy cost if its submitted data is used in a differentially private computation.

The above discussion provides a way for the server to compute bonus b. That is, the server can predefine a value v that represents the private valuations of the majority of clients. Then, the extra bonus is computed as

$$b = \epsilon \cdot v. \tag{4.14}$$

The value v can be adjusted based on the server's total monetary budget and its requirement for data accuracy. If the server wishes to get more accurate parameters, it can set a large v to increase compensation amounts, which may drive more clients to choose the centralized option.

The server is also responsible for aggregating the model parameters received from each clients. This includes both the noisy parameters and the accurate parameters. Usually, the most widely used parameter aggregation method in federated learning is a weighted average over the clients' data. In the proposed framework, it is not reasonable to simply compute an average over the client submissions because it violates the privacy of clients who submit parameters without adding noise. In addition, the server will spend more money in exchange for more accurate client data. Averaging the accurate parameters with noisy ones is a waste of the server's budget. To make the best of the received data, the server therefore aggregates the parameters following Algorithm 8, which contains a traditional weighted average method (Line 2, weights have been omitted for simplicity). For the parameters without local perturbation, the server employs a centralized Gaussian mechanism with ϵ_C to achieve a differential privacy guarantee (Lines 4 – 8). A parameter w is used to adjust the weight between above two aggregated values (Line 9). Finally, the server returns θ to each client for local update (Line 10).

In Step ⑦, the server stops the training once it receives satisfactory model parameters and pays corresponding rewards and bonus to each client according to their choice.

Algorithm 8 Private parameter aggregation mechanism $\mathcal{M}_3$

Input: the noisy parameters $\theta_1,\dots\theta_k$, the parameters without noise $\theta_{k+1},\dots\theta_n$, the privacy budget ϵ_C for centralized perturbation, sensitivity $\Delta_2 f$, weight w.

Output: the aggregated parameters θ.

1: **for** $i \in [k]$ **do**
2: $\quad \theta_L \leftarrow \sum_{i=1}^{k} \theta_i / k$
3: **end for**
4: **for** $i \in [k+1, n]$ **do**
5: $\quad \theta_C \leftarrow \sum_{i=k+1}^{n} \theta_i / (n-k)$
6: **end for**
7: sample $Z \sim \mathcal{N}(0, \sigma)$, where $\sigma = \sqrt{2 \ln \frac{1.25}{\sigma}} \cdot \frac{\Delta_2 f}{\epsilon_C}$
8: $\hat{\theta}_C \leftarrow \theta_C + Z$
9: $\theta \leftarrow w \cdot \theta_L + (1-w) \cdot \hat{\theta}_C$
10: **return** θ

4.4 Theoretical analysis

The theoretical analyses begin with the necessary stop criteria of the equilibrium computation mechanism. We then connect the differentially private property to the existence of an approximate Nash equilibrium. A convergence analysis of the reward game is also given.

We first show that the proposed reward game is a potential game. Consider a client c_i who switches from playing strategy $a_i \in A_i$ to playing $b_i \in A_i$ in a single step of best response update. Such a switch will increase c_i's reward.

$$\begin{aligned} \Delta r_i &= r_i(\tau_i, (b_i, a_{-i})) - r_i(\tau_i, (a_i, a_{-i})) \\ &= \sum_{(j,k) \in b_i \setminus a_i} f(n_{jk} + 1) - \sum_{(j,k) \in a_i \setminus b_i} f(n_{jk}) > 0. \end{aligned} \tag{4.15}$$

Consider the potential function $\Phi(a) : A \to \mathbb{R}$ defined as follows,

$$\Phi(a) = \sum_{j=1}^{m} \sum_{k=1}^{l} \sum_{n=1}^{n_{jk}} f(n) \tag{4.16}$$

The change in the potential function is therefore

$$\begin{aligned} \Delta\Phi(a) &= \Phi(b_i, a_{-i}) - \Phi(a_i, a_{-i}) \\ &= \sum_{(j,k) \in b_i \setminus a_i} f(n_{jk} + 1) \sum_{(j,k) \in a_i \setminus b_i} f(n_{jk}) \\ &= \Delta r_i. \end{aligned} \tag{4.17}$$

Based on the potential game, we can bound the total number of α-noisy best responses clients can make before no one can improve his reward by choosing a new move.

Theorem 4.3. *The total number of α-noisy best responses for a client before its reward will improve no further is bounded. It is sufficient to set the bound of iteration to*

$$T = \frac{2mnl}{\alpha}. \tag{4.18}$$

Proof. Consider a binary mechanism that is used to track the number of clients contributing data at each quality level. According to Theorem 4.2, with a probability of at least $1-\beta$, the maximum difference at a single timestep t between $\hat{n}_{jk}$ and n_{jk} is

$$\max_{t\in T} |\hat{n}_{jk} - n_{jk}| \leq \frac{2\sqrt{2}}{\epsilon} \ln(\frac{2}{\beta}) \log(T)^{\frac{1}{2}}. \tag{4.19}$$

If the confidence bound β is allocated over all possible quality levels, the error for each quality level is

$$E_\beta = \frac{2\sqrt{2}}{\epsilon} \ln(\frac{2ml}{\beta}) \log(T)^{\frac{1}{2}}. \tag{4.20}$$

Assume that the error in a private counter at each timestep is bounded by E_β and, therefore, the difference between the noisy reward and the accurate reward for a client can also be bounded. Let a^t denote the action profile at time $t \in [T]$, at timestep t, with a probability of at least $1-\beta$, we have

$$\begin{aligned} |\hat{r}_i(\tau_i, a^t) - r_i(\tau_i, a^t)| &= \sum_{j,k\in a_i^t} |f(\hat{n}_{jk}) - f(n_{jk})| \\ &\leq m\Delta\mathcal{G}E_\beta. \end{aligned} \tag{4.21}$$

Suppose at timestep t, a client is allocated an α-noisy best response and the action profile is updated from a^t to a^{t+1}. Then, we have

$$\begin{aligned} r_i(\tau_i, a^{t+1}) &\geq \hat{r}_i(\tau_i, a^{t+1}) - m\Delta\mathcal{G}E_\beta \\ &\geq \hat{r}_i(\tau_i, a^t) + \alpha - m\Delta\mathcal{G}E_\beta \\ &\geq r_i(\tau_i, a^t) + \alpha - 2m\Delta\mathcal{G}E_\beta. \end{aligned} \tag{4.22}$$

That is, although it may appear as though a client is significantly improving its reward, due to the effect of noise, the actual improvement in reward may be only $\alpha - 2m\Delta\mathcal{G}E_\beta$.

The potential function $\Phi(a) \leq mnl$ in Equation 4.1 can be bounded based on the fact that the reward function f on each quality level is in $[0,1]$. Therefore, the total number of α-noisy best response clients can make is bounded by

$$T \leq \frac{mnl}{\alpha - 2m\Delta\mathcal{G}E_\beta}. \tag{4.23}$$

We can assume $\alpha \geq 4m\Delta\mathcal{G}E_\beta$ and obtain $T \leq \frac{2mnl}{\alpha}$. □

Theorem 4.4. *On the assumption of a bounded E_β, the number of α-noisy best responses available to one client is bounded by*

$$k \leq \frac{8m^2nl\Delta\mathscr{G}}{\alpha^2}. \tag{4.24}$$

On the other hand, the proposed equilibrium computation mechanism will never halt for $k \geq \frac{8m^2nl\Delta\mathscr{G}}{\alpha^2}$.

Proof. One client can improve its reward by making a best response. The concept of *regret* is used to capture increases in the reward, which is computed using the exact reward function.

$$regret(\tau_i, a) = \max_{a_i' \in A_i}\{r_i(\tau_i, (a_i', a_{-i}))\} - r_i(\tau_i, a). \tag{4.25}$$

Obviously, if the counters are computed without noise, at each step, client c_i's regret $regret(\tau_i, a) = 0$ because rational clients will always choose quality levels that maximize their rewards.

Let $a_i^* = \max_{a_i' \in A_i}\{r_i(\tau_i, (a_i', a_{-i}))\}$. At time t, we have

$$\begin{aligned} regret(\tau_i, a^t) &= r_i(\tau_i, (a_i^*, a_{-i}^t)) - r_i(\tau_i, a^t) \\ &\leq \hat{r}_i(\tau_i, (a_i^*, a_{-i}^t)) + m\Delta\mathscr{G}E_\beta \\ &\quad - (\hat{r}_i(\tau_i, a^t) - m\Delta\mathscr{G}E_\beta) \\ &\leq 2m\Delta\mathscr{G}E_\beta. \end{aligned} \tag{4.26}$$

Then, we bound the difference between $regret(\tau_i, a^{t+1})$ and $regret(\tau_i, a^t)$. Note that after one client $j \neq i$'s move, for client c_i, we have

$$|r_i(\tau_i, a^{t+1}) - r_i(\tau_i, a^t)| \leq \sum_{j,k \in a_i^t} |f(\hat{n}_{jk} + 1) - f(\hat{n}_{jk})| \leq m\Delta\mathscr{G}. \tag{4.27}$$

This corresponds to the worst case where client c_j chooses the same quality levels as client c_i, and thus, in computing c_i's rewards, all the related counts are increased by 1.

Then, if client c_i does not move in its round, we have

$$\begin{aligned} regret(\tau_i, a^{t+1}) &= r_i(\tau_i, (a_i^*, a_{-i}^{t+1})) - r_i(\tau_i, a^{t+1}) \\ &\leq r_i(\tau_i, (a_i^*, a_{-i}^t)) + m\Delta\mathscr{G} \\ &\quad - (r_i(\tau_i, a^t) - m\Delta\mathscr{G}) \\ &= regret(\tau_i, a^t) + 2m\Delta\mathscr{G}. \end{aligned} \tag{4.28}$$

That is, if a client does not move on their turn, its regret will be $2m\Delta\mathscr{G}$.

Supposing a client makes an α-noisy best response in timestep t_1, the next time it can make an α-noisy best response is t_2. Recursively using Equation 4.28, we have

$$regret(\tau_i, r^{t_1+t_2}) \leq 2m\Delta\mathcal{G}E_\beta + (t_2 - t_1)2m\Delta\mathcal{G}. \tag{4.29}$$

Therefore, when the $regret(\tau_i, r^{t_1+t_2})$ accumulates to α, that is $t_2 - t_1 = \frac{\alpha - 2m\Delta\mathcal{G}E_\beta}{2m\Delta\mathcal{G}}$, client c_i can make another α-noisy best response. Based on Equation 4.23, we have

$$k \leq \frac{T}{t_2 - t_1} \leq \frac{2m^2 nl\Delta\mathcal{G}}{(\alpha - 2m\Delta\mathcal{G}E_\beta)^2}. \tag{4.30}$$

Taking the assumption of $\alpha \geq 4m\Delta\mathcal{G}E_\beta$ into the above equation, we can obtain a bounded value for k. □

Next we examine the privacy guarantee promised by the proposed mechanism $\mathcal{M}_1$. To prove $\mathcal{M}_1$ satisfies joint differential privacy, we first introduce the billboard lemma [53] and the post-processing property of differential privacy. The billboard lemma states that if each client's output is a function only of an ϵ-differentially private computation and their own private data, then the overall mechanism satisfies ϵ-joint differential privacy.

Lemma 4.1. *(**Billboard Lemma** [53]) Suppose $\mathcal{M} : D \to \mathcal{R}$ is (ϵ, δ)-differentially private. For any function $f_i : D_i \times \mathcal{R} \to \mathcal{R}'$, where D_i is the portion of the database containing i's data, mechanism $\mathcal{M}'$ that outputs to each client i: $f_i(D_i, \mathcal{M}(D))$ is (ϵ, δ)-jointly differentially private.*

In addition, differential privacy is immune to post-processing. Applying an arbitrary data-independent post mapping f to a differentially private mechanism $\mathcal{M}$ does not influence its privacy guarantee.

Lemma 4.2. *(**Post-processing** [37]) Suppose $\mathcal{M} : D \to \mathcal{R}$ is a (ϵ, δ)-differentially private mechanism. Let $f : \mathcal{R} \to \mathcal{R}'$ be an arbitrary randomized mapping. Then $f \circ \mathcal{M} : D \to \mathcal{R}'$ is (ϵ, δ)-differentially private.*

Theorem 4.5. *The equilibrium computation mechanism $\mathcal{M}_1$ satisfies (ϵ, δ)-joint differential privacy.*

Proof. First consider the case where mechanism $\mathcal{M}_1$ does not fail, which happens with a probability of $1 - \beta$. Let $\mathcal{M}$ to denote the mechanism that outputs all the partial sums [36] during the course of $\mathcal{M}_1$, and let p_τ denote the probability of generating a specific set of partial sums, and $p_{\tau'}$ be the same probability when client c_i reports τ_i' rather than τ_i. According to Theorem 4.2, each private counter is ϵ-differentially private,

and each single element in a bit stream can appear in at most $\log(T)$ partial sums in the binary mechanism $Binary(\{e_i\},\epsilon')$.

From Theorem 4.4 we know that each client can make at most k best response moves, and thus there are at most $2k$ changes in the probability ratio $p_\tau/p_{\tau'}$ given different τ and τ'. Considering the fact that one client's change can affect at most $2m$ private counters, we conclude that mechanism $\mathscr{M}_1$ is $(2k \cdot 2m \cdot \epsilon \cdot \log(T))$-differentially private. Therefore, with a probability of $1-\beta$, mechanism $\mathscr{M}$ satisfies ϵ-differential privacy for

$$\epsilon' = \frac{\epsilon}{4km\log(T)}. \tag{4.31}$$

Note that mechanism $\mathscr{M}_1$ will fail with a probability of β . Overall, $\mathscr{M}$ satisfies (ϵ,β)-differential privacy.

As the output of $Binary(\{e_i\},\epsilon')$ can be regarded as a post-processing of partial sums, according to Lemma 4.2, $\mathscr{M}$ still satisfies differential privacy if it computes noisy counts on each quality level using the output of $Binary(\{e_i\},\epsilon')$.

Finally, we invoke the billboard lemma – where each client can observe the output of $\mathscr{M}$ and figure out which quality levels they are currently contributing by combining their private type – therefore, the mechanism $\mathscr{M}_1$ is jointly differentially private. □

Theorem 4.6. *With a probability of at least $1-\beta$, the equilibrium computation mechanism $\mathscr{M}_1$ outputs an η-approximate Nash equilibrium.*

Proof. According to Theorem 4.4, we know that, under the bounded assumption of E_β, mechanism $\mathscr{M}_1$ outputs an action profile. In this scenario, the difference between the noisy reward $\hat{r}_i$ and the exact reward r_i for each client is bounded by $m\Delta\mathscr{G}E_\beta$ (Equation 4.21). In addition, nT rounds of moves are enough for n clients to make T α-noisy best responses, where T is given in Theorem 4.3. Therefore, at the end of mechanism $\mathscr{M}_1$, no clients can make an α-noisy best response. At that time, we have

$$\eta = regret(\tau_i,a) \leq \alpha + 2m\Delta\mathscr{G}E_\beta. \tag{4.32}$$

Note that the bounded assumption about E_β holds with a probability of at least $1-\beta$ (Equation 4.20). Therefore, with a probability of at least $1-\beta$, mechanism $\mathscr{M}_1$ outputs an η-approximate Nash equilibrium. □

Theorem 4.7. *Good behavior by clients forms an η-approximate ex-post Nash equilibrium of the game $\mathscr{G}_M$, where good behavior is when a client truthfully reports his type to the mediator, and then faithfully follows its suggestion.*

Proof. Recall that we use $BR_i(a_{-i}) = \arg\max_{a_i \in A_i} r_i(a_i, a_{-i})$ to denote the best response of client c_i. According to the definition of reward (i.e., Equation 4.9), we have

$$r_i(\tau_i, (\tau', g)) = \mathbb{E}_{a \sim M(\tau')}[r_i(\tau_i, g(a))] \tag{4.33}$$

In addition, based on Theorem 4.6, mechanism $\mathscr{M}_1$ (with a probability of $1-\beta$) computes an η-approximate Nash equilibrium given the reported types. Therefore,

$$\begin{aligned} r_i(\tau_i, (\tau, g)) &= \mathbb{E}_{a \sim M(\tau)}[r_i(\tau_i, g(a))] \\ &\geq \mathbb{E}_{a \sim M(\tau)}[r_i(\tau_i, (BR_i(a_{-i}), g_{-i}(a_{-i}))] - \eta. \end{aligned} \tag{4.34}$$

According to Theorem 4.5, mechanism $\mathscr{M}_1$ satisfies (ϵ, δ)-joint differential privacy. Taking the restraints of joint differential privacy into the above equation, we have

$$\begin{aligned} & r_i(\tau_i, (\tau, g)) \\ & \geq e^{-\epsilon} \cdot \mathbb{E}_{a \sim M(\tau_i', \tau_{-i})}[r_i(\tau_i, (BR_i(a_{-i}), g_{-i}(a_{-i}))] - \delta - \eta \\ & \geq \mathbb{E}_{a \sim M(\tau_i', \tau_{-i})}[r_i(\tau_i, (BR_i(a_{-i}), g_{-i}(a_{-i}))] - m\epsilon - \delta - \eta \\ & \geq \mathbb{E}_{a \sim M(\tau_i', \tau_{-i})}[r_i(\tau_i, (g'(a_i), g_{-i}(a_{-i}))] - m\epsilon - \delta - \eta \\ & = r_i(\tau_i, (\tau_i', g_i'(a_i), g_{-i}(a_{-i}))) - m\epsilon - \delta - \eta. \end{aligned} \tag{4.35}$$

The first inequality follows from the definition of joint differential privacy. The second inequality is because $\exp(-\epsilon) \geq 1 - \epsilon$ for $\epsilon \leq 1$. The last inequality is from the definition of best response.

Considering the fact that, with a probability of β, $\mathscr{M}_1$ will fail to return an approximate Nash equilibrium, where clients may increase their reward by m, we have

$$r_i(\tau_i, (\tau, g)) \geq r_i(\tau_i, (\tau_i', g_i'(a_i), g_{-i}(a_{-i}))) - m\epsilon - \delta - \eta - m\beta. \tag{4.36}$$

That is, good behaviors by clients form an approximate equilibrium for the game $\mathscr{G}_M$ for every possible configuration of client types. According to Definition 4.2, good behaviors form an *ex-post* Nash equilibrium of $\mathscr{G}_M$. □

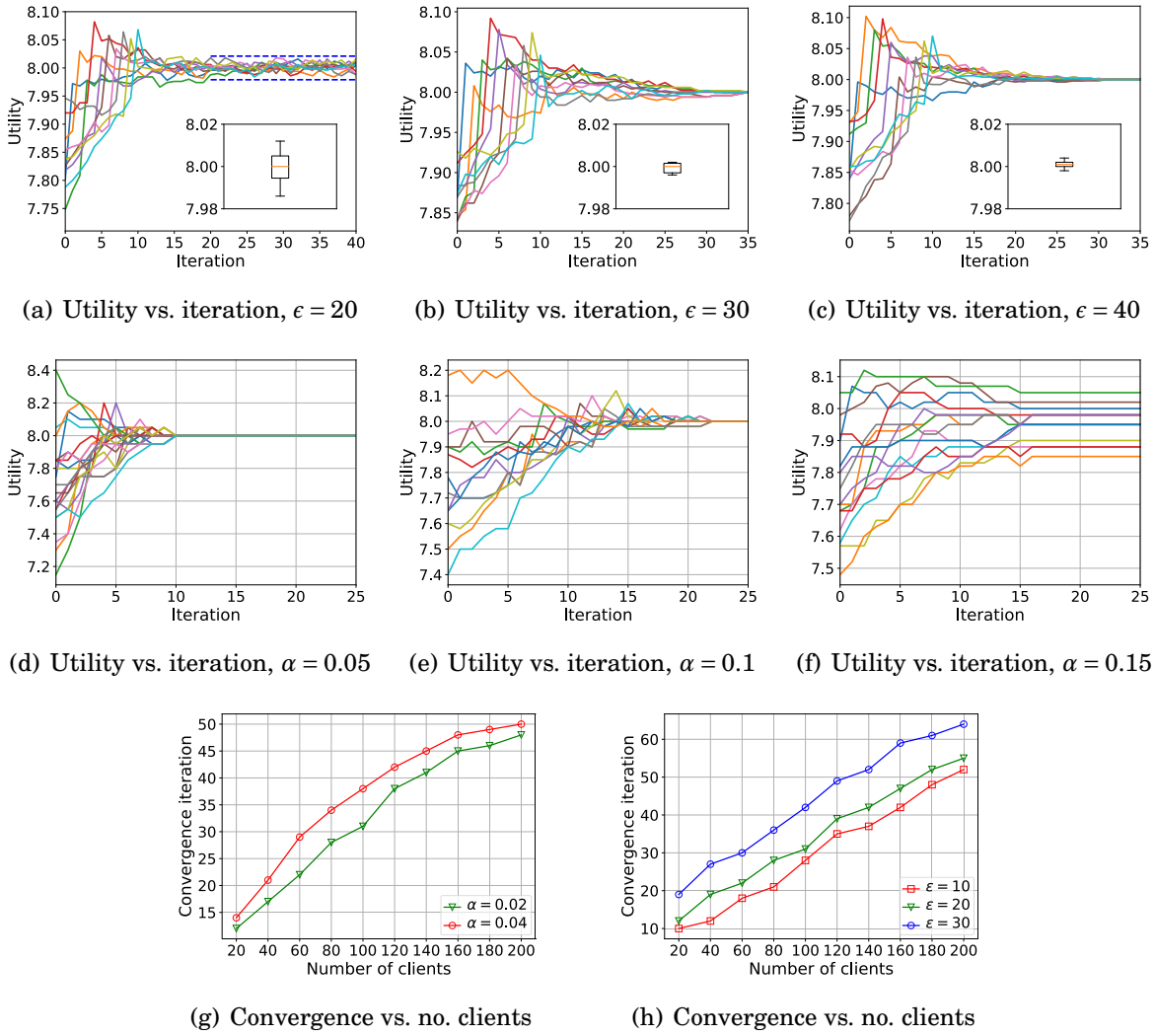
4.5 Experimental results

4.5.1 Experimental results for the game module

4.5.1.1 The influence of ϵ on client's utility

We first investigated the influence of private counters on client utility. In this experiment, we used $n = 50$ clients. The dataset consists of $m = 10$ classes, with $l = 5$ levels of quality.

In addition, the reward function f in Equation 4.7 was chosen as $f(x) = -x/50 + 1$, and the parameter α for clients to make a best response was set to 0.02, namely 1/50. For a better presentation, we used large ϵ to clearly show the detailed convergence tendency of client's utility around the approximate equilibrium point. The theoretical utility at the equilibrium point was 8.0 without normalization.



(a) Utility vs. iteration, $\epsilon = 20$ (b) Utility vs. iteration, $\epsilon = 30$ (c) Utility vs. iteration, $\epsilon = 40$

(d) Utility vs. iteration, $\alpha = 0.05$ (e) Utility vs. iteration, $\alpha = 0.1$ (f) Utility vs. iteration, $\alpha = 0.15$

(g) Convergence vs. no. clients (h) Convergence vs. no. clients

Figure 4.2: The game-theoretic properties of the reward game.

We set different privacy budgets for the private counters, randomly selected 10 clients, and tracked the change in their utility. The results are shown in Figures 4.2(a) to 4.2(c). Initially, client utilities increased because they were allowed to adjust their strategy as long as the improvement was greater than $\alpha = 0.02$. However, with more iterations, and more clients adjusting their strategies, utility for some clients suffered

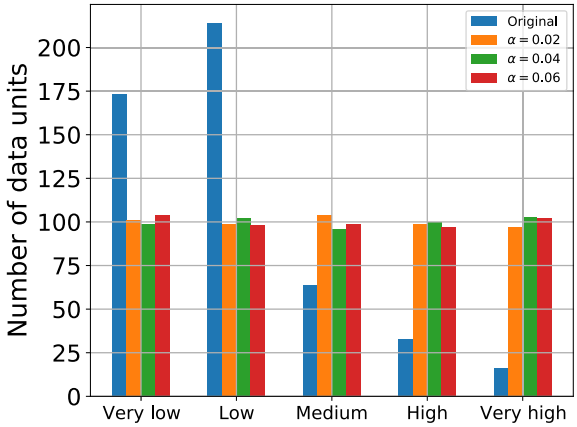


Figure 4.3: The effectiveness on data quality improvement of the game.

as a result of adjustments by others. As the iterations progressed, no client could find a more preferable strategy, and the reward game converged to equilibrium.

In Figure 4.2(a), the counters are initialized with a smaller privacy budget, and thus the counts for each quality level are not very accurate. In this situation, some clients may be able to slightly improve their utilities, but the increase will soon be offset by the movements of others. Consequently, the client utility curves fluctuate around the equilibrium point. The blue dashed lines demonstrate the theoretic upper and lower bounds of the approximate Nash equilibrium, which is given in Equation 4.32. The boxplot in each figure illustrates the distribution of client utilities when the game reaches equilibrium. With a larger privacy budget, the private counters output more accurate counts; therefore, client utility converges to a narrower interval in Figure 4.2(b) and 4.2(c). That is why the box in each boxplot shrinks as the privacy budget increases.

4.5.1.2 The influence of α

The parameter α indicates whether a client can make a best response in their round, which influences the convergence of the reward game. In this experiment, we set different values of α with $n = 20$ clients. The reward function is $f(x) = -x/20 + 1$, and the results are shown in Figure 4.2(d) to 4.2(f). In Figure 4.2(d), we see that, with $\alpha = 0.05$, the reward game reaches equilibrium after 10 iterations, and the utility of each client at equilibrium is about 8.0. In comparison, in Figure 4.2(e), when α increases to 0.1, the reward game needs more iterations to converge. This is because a small α allows clients to adjust their strategy whenever they can improve their utility. By contrast, with a larger α, one client has to wait until when a new choice can improve their utility to at

least α. In this scenario, clients' utility in the equilibrium is still around 8.0. Notably, in Figure 4.2(f), when α is large enough, the reward game will reach a deadlock after some iterations and it will not converge anymore. The reason is that clients cannot reach the threshold α in their iterative strategy adjustment, so they cannot make any new move. However, the reward game still reaches the equilibrium as no clients have incentive to deviate from their current strategy under the requirement of α-best response.

4.5.1.3 The influence of client number

With our next experiment, we set out to explore the relationship between the number of iterations to convergence and the number of participating clients. Intuitively, the reward game should require more iterations to reach equilibrium with a larger number of clients. Figures 4.2(g) and 4.2(h) confirm this conjecture. Figure 4.2(g) illustrates the change in iterations to converge when the number of clients grows. In addition, the game with a larger α converges on an equilibrium more slowly than that with small α. A similar tendency can also be observed in Figure 4.2(h). This figure shows that the game with a smaller ϵ reaches equilibrium faster. This is because the range of the approximation interval (i.e., Equation 4.32) computed with a small ϵ is wider than that with a large ϵ. Although client utility fluctuates within the approximation interval, equilibrium establishes earlier with a small ϵ. This phenomenon can also be observed by comparing the convergence iteration in Figure 4.2(a) (about 15 rounds to converge), Figure 4.2(b) (about 25 rounds to converge), and Figure 4.2(c) (about 30 rounds to converge).

4.5.1.4 The effect of the game and related discussion

One of the main goals of the game module is to incentivize more clients to use high quality data in their local training. The effect of the reward game is shown in Figure 4.3. In the experiment, we presumed $l = 5$ levels of data quality, i.e., {very low, low, medium, high very high}. Without playing the game, clients tend to use low quality data in the training, as illustrated by the blue bars. After the game reaches its equilibrium, the distribution of data contributed by clients is almost uniform on different data qualities, and the amount of high quality data is significantly increased. However, we note that the game itself is only a way of monetary implementation, the actual factor that incentivizes clients to use higher quality data is payment. It is worth noting that the solution at the equilibrium point is *not* optimal, which, obviously, can be observed from Figure 4.3 – the game only pushes the data quality from a low-quality-oriented distribution to a near-uniform distribution, whereas in the optimal situation, most of clients should

contribute their high quality data. The phenomenon is caused by the nature of the Nash equilibrium, which is only a quiescent status where no rational clients have incentive to unilaterally change their strategy. To get optimal solution, more reward should be introduced to break the balance between the clients' gain and their private valuation about the data.

4.5.2 Experimental results for private learning module

In this section, we examined the proposed private learning module in terms of the both parameter update scenarios. The privacy budgets were calibrated by $\epsilon = \epsilon_L = 0.1\epsilon_C$, following the intuition that the server would make the most of the data it buys.

4.5.2.1 Adding noise on model weights

We denote p_c the proportion of clients who use the centralized differential privacy proxy to protect their privacy, i.e.,

$$p_c = \frac{\text{no. clients who use the privacy proxy}}{\text{no. participating clients}}. \tag{4.37}$$

Figure 6.4 shows the change in the global model's accuracy when clients submit model weights to the server. In this experiment, we set $n = 100$ clients who participate in the training. Figure 4.4(a) illustrates how accuracy changes during the training process, given a fixed privacy budget $\epsilon = 1.5$. The model accuracy increases with the growth of training epoch. A significant improvement in accuracy can be observed when more clients choose the centralized differential privacy proxy. For example, in the case $p_c = 0$, i.e., all the clients add noise locally to their weights, the accuracy is around 96.1%. In comparison, when every client adopts the centralized privacy proxy, i.e., $p_c = 100\%$, the accuracy is improved to 98.6%. A detailed variation tendency of accuracy against p_c is demonstrated in Figure 4.4(b). In Figure 4.4(b), we see that the increase of p_c leads to an increase of the global model's accuracy, which means that the proposed framework actually improves the performance of the global model. Further, the privacy budget also influences the model accuracy. As shown in Figure 4.4(c), a larger privacy budget results in higher accuracy. In addition, the gaps of accuracy among each p_c level become more significant when the privacy budget is small. For example, given $\epsilon = 0.4$, the accuracy with $p_c = 0$ is about 58.5%, which is improved to 81.1% when p_c increases to 75%. This shows that, the proposed framework leads to better results when clients have more rigorous privacy requirement on their parameter updates. However, to keep a high p_c,

the server has to spend more monetary compensation to encourage clients to use the centralized privacy proxy, which is in accordance with the money-accuracy tradeoff.

Figures 4.4(d) to 4.4(f) show similar tendencies as Figures 4.4(a) to 4.4(c), which are obtained using different dataset. In Figure 4.4(d), the accuracy of the global model improves 5.2% if half of the clients choose to use the centralized differential privacy proxy, and this improvement will raise to 10.9% if all the clients are involved in using it. These experiments demonstrate the effectiveness of the proposed framework when differentially private noise is added to the model's weight updates.

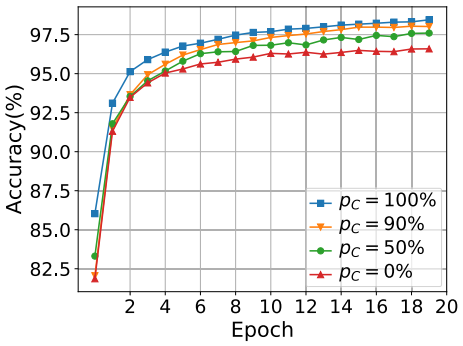
4.5.2.2 Adding noise on the gradients

Figure 4.5(a) illustrates the change in model accuracy with respect to training epoch, where the privacy budget was set $\epsilon = 0.5$. Overall, the accuracy of the global model goes up with the increase of training epoch. Specifically, when all the clients add noise locally to the gradients, the model accuracy is around 87.8%. By contrast, the accuracy is improved to 92.1% when half of the clients choose the centralized differential privacy proxy, and it reaches 94.7% if everyone is involved. Figure 4.5(b) shows the detailed variation tendency of model accuracy with the change of p_c. From this figure, we observe that, with a fixed privacy budget, larger p_c always results in higher accuracy. In other words, the proposed framework indeed improves the performance of the global model. Further, Figure 4.5(c) shows the influence of privacy budget on model accuracy. The figure demonstrates a significant improvement in model accuracy when ϵ is small. Therefore, the proposed framework is more advantageous under a stronger privacy requirement.

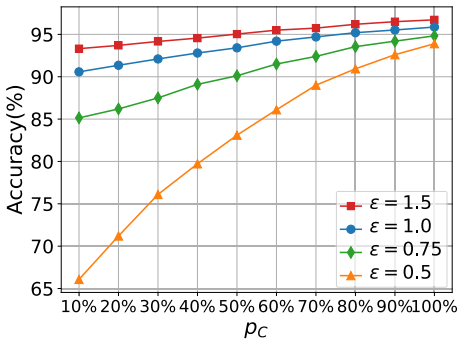
What Figures 4.5(d) to 4.5(f) show are the results obtained on CIFAR-10 dataset. The result patterns are similar as those in Figures 4.5(a) to 4.5(c). Figure 4.5(d) describes the change in model accuracy during the training process, given $\epsilon = 4$. Figure 4.5(e) and 4.5(f) illustrate the influence of p_c and ϵ respectively, where larger p_c and ϵ lead to better model accuracy. These experiments demonstrate the effectiveness of the proposed framework when differentially private noise is added to the model's gradient updates.

4.5.2.3 Discussion and comparison of proposed framework

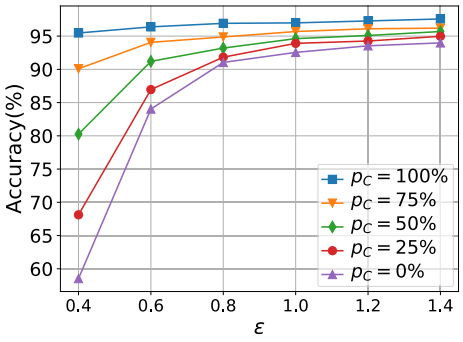
To show the outperformance of the proposed framework, we used a traditional federated learning framework FedAvg as baseline, and adopted the proposed framework in other differentially private federated learning methods to make detailed comparisons. The results are shown in Table 4.1.



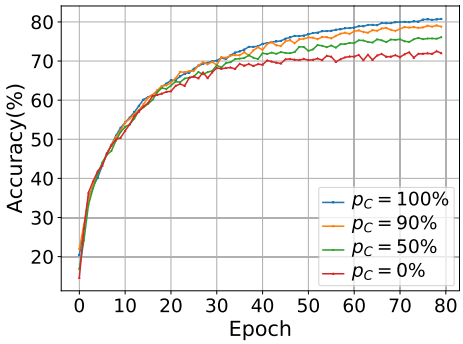
(a) Acc. vs. epoch, MNIST



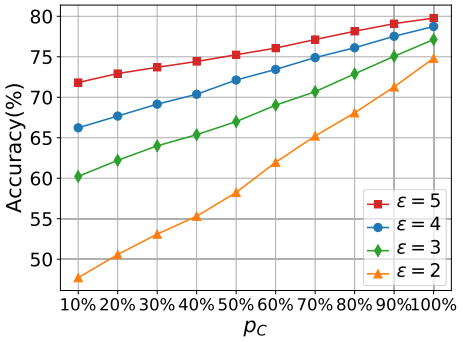
(b) Acc. vs. p_c, MNIST



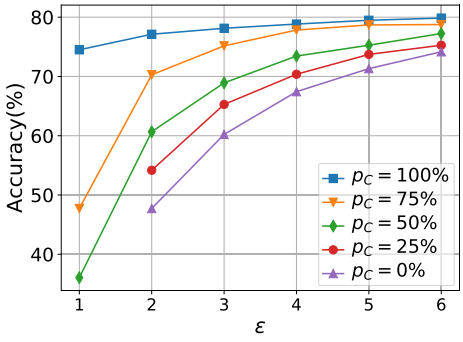
(c) Acc. vs. ϵ, MNIST



(d) Acc. vs. epoch,CIFAR-10.

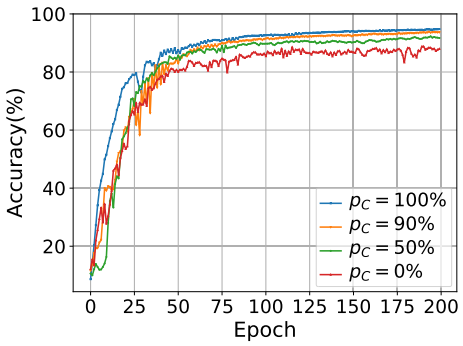


(e) Acc. vs. p_c, CIFAR-10.

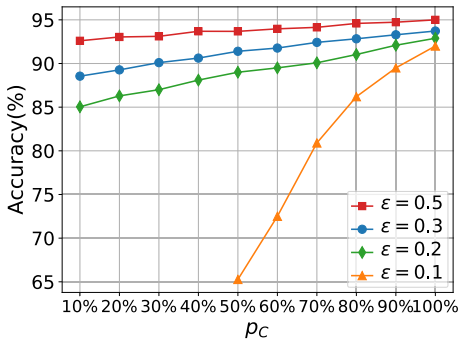


(f) Acc. vs. ϵ, CIFAR-10.

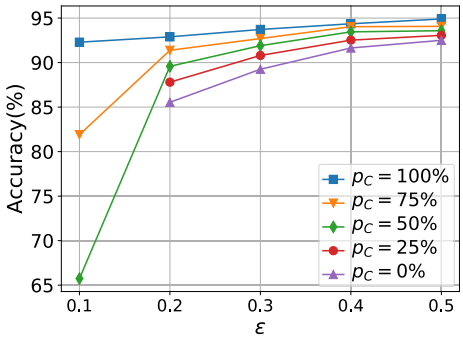
Figure 4.4: The change of model accuracy when differentially private noise is added to the model's weight updates.



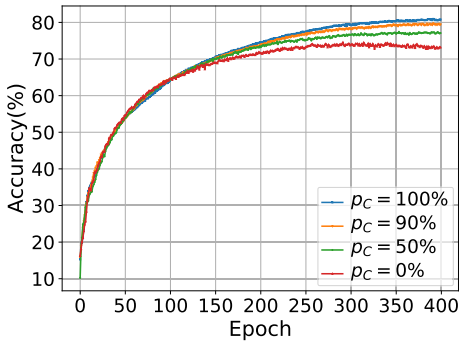
(a) Acc. vs. epoch, MNIST



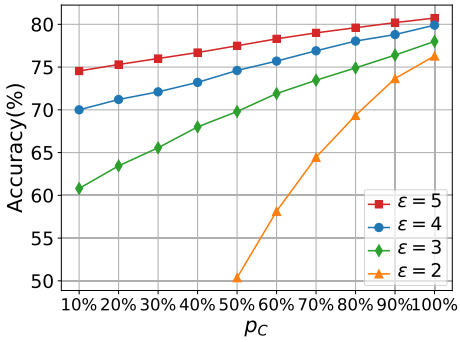
(b) Acc. vs. p_c, MNIST



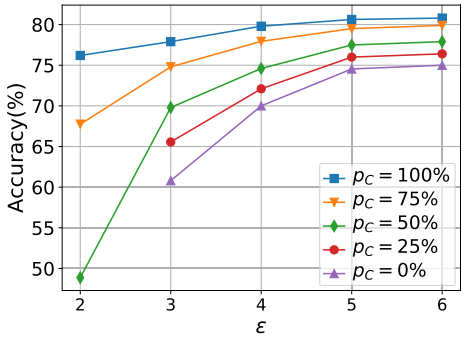
(c) Acc. vs. ϵ, MNIST



(d) Acc. vs. epoch, CIFAR-10



(e) Acc. vs. p_c, CIFAR-10



(f) Acc. vs. ϵ, CIFAR-10

Figure 4.5: The change of model accuracy when differentially private noise is added to the model's gradient updates.

Table 4.1 presents the results of the proposed framework in comparison to other similar works. Clearly, the proposed framework results in better model accuracy. However, we note that the advantage is established based on the server's expenditure, which provides incentive for clients to use the centralized privacy proxy. Meanwhile, some properties may not holds by introducing incentives. For example in [2], the Rényi divergence between Skellam distributions may change, and thus the Skellam mechanism may fail. Ref [127] adopts contract theory to encourage clients to add less noise in their local perturbation, while our proposed framework allows clients to choose the centralized privacy proxy directly, so that our framework achieves a higher accuracy.

Finally, we emphasize that a balance between money and privacy should be made in real-world application. The proposed framework leads to more significant improvement when the privacy budget is small, but in such scenario, clients have more rigorous requirement on their privacy, and thus the server needs to pay more to encourage clients to use the centralized privacy proxy. That demonstrates the intuition of "money for privacy" throughout this chapter.

Table 4.1: Comparison of the proposed framework with others

Methods	Dataset	Accuracy (↑)	Incentive	Properties
FedAvg	MNIST	94% (**2.3%**)	Not used	-
FedAvg	CIFAR10	73% (**6.2%**)	Not used	-
FedAvg	CIFAR100	72% (**3.6%**)	Not used	-
ref [148]	FMNIST	89% (**4.1%**)	Not used	Not hold
ref [2]	EMNIST	80% (**2.4%**)	Not used	Not hold
ref [127]	MNIST	95% (**1.2%**)	Used	Hold

4.6 Summary

This chapter proposes a private reward game to motivate clients to contribute their high-quality data to federated learning. The reward game employs a mediator to give suggestions to each client, and good behavior of clients will form an *ex-post* Nash equilibrium of the incomplete information game. We provided detailed proofs and analyses of the game's properties, along with an examination of the privacy guarantee. Additionally, we conducted simulations with real-world datasets to validate the framework's performance. The results demonstrate that, under our scheme, model accuracy is better than it is under traditional federated learning.

CHAPTER 5

APT RIVALRY GAME: DEFENDING AGAINST ADVANCED PERSISTENT THREATS

Advanced persistent threats (APTs) are one of today's major threats to cyber security. Highly determined attackers along with novel and evasive exfiltration techniques mean APT attacks elude most intrusion detection and prevention systems. The result has been significant losses for governments, organizations, and commercial entities. Intriguingly, despite greater efforts to defend against APTs in recent times, frequent upgrades in defense strategies are not leading to increased security and protection. This chapter demonstrates this phenomenon in an appropriately designed APT rivalry game that captures the interactions between attackers and defenders. What is shown is that the defender's strategy adjustments actually leave useful information for the attackers, and thus intelligent and rational attackers can improve themselves by analyzing this information. Hence, a critical part of one's defense strategy must be finding a suitable time to adjust one's strategy to ensure attackers learn the least possible information. Another challenge for defenders is determining how to make the best use of one's resources to achieve a satisfactory defense level. This chapter figures out the optimal timings of a player's strategy adjustment in terms of information leakage, which form a family of Nash equilibria. Moreover, two learning mechanisms are proposed to help defenders find an appropriate defense level and allocate their resources reasonably. Experimental results show the optimality of the equilibria and demonstrate the necessity of specifying optimal strategy adjustment timings when defending against APTs.

5.1 Introduction

With the fast evolution and frequent innovation in cyber attack/defense techniques, APTs [27, 129] have become a prominent threat to cyber security [77]. Different from traditional cyber attacks, today's APTs are long-term, stealthy attacks performed by proficient and well-funded adversaries. The goal of an APT attack is not only limited to sabotaging critical infrastructures and/or exfiltrating critical information [95], but also includes staying undetected for further exfiltration and/or sabotage [155]. Moreover, one salient characteristic of APT attacks is that the threats keep changing. An increasing security budget does not effectively protect an organization from compromise [22, 117]. The key reason is that, through frequent interactions, the attackers and defenders learn from each other, making the attackers more sophisticated and stronger than ever before.

It is important to note that the defenders are not incompetent, it is continuous learning through a long and determined APT campaign means the attackers become more and more competitive with every interaction. More importantly, the attackers continuously collect information about their target, adapting to the defender's resistance efforts by sustained analysis and learning [91, 136]. Although the defenders are able to successfully detect/prevent some attempts at intrusion by dynamically adjusting their defense strategies and periodically upgrading their defense policies, these adjustments may leave useful information for the attackers [135, 154]. For example, if an attacker gets detected and prevented from today's defense method, they may start to analyze the situation and develop a novel attack variant to break through that limitation[3].

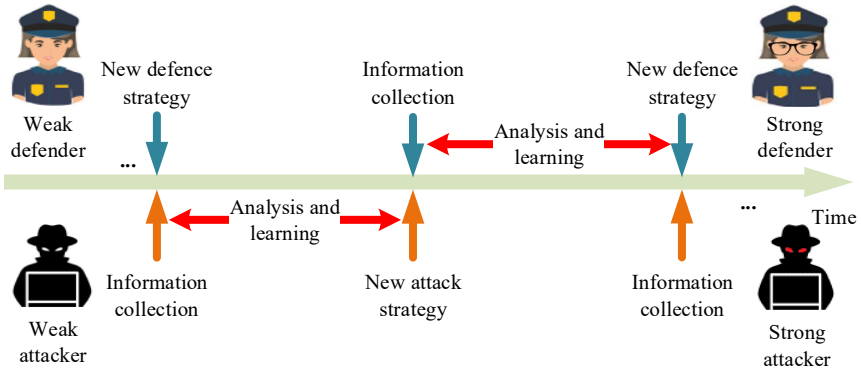


Figure 5.1: The evolution of attacker and defender in APT attacks.

Figure 5.1 demonstrates the process. In the beginning, the attacker has limited knowledge about the defender. When the defender employs a new defense strategy, some

pieces of information are left for the attacker. This provides the attacker with a chance to learn more about the defender's defense strategy – even if that new defense strategy leads to a better defense level. In turn, developments in the attacker's infiltration techniques will motivate the defender to devise more advanced methods of protection. From this standpoint, it is important for the defender to choose a time to make their strategy adjustment when they can ensure the least information will be learned by the attacker.

Today, many techniques have been adopted to model the frequent interactions between attackers and defenders in APT campaigns. Of these, game theory is one of the most powerful choices [31, 166]. Game theory provides an efficient way to investigate how to defend against APT attacks, and, as such, many game-theoretic models have been devised that describe competition between an attacker and a defender [87, 118, 168]. However, existing solutions generally use a single, highly-abstracted game to characterize the interactions between the two opponents, and such games do not tend to reflect the attack-defense processes in real APT scenarios. First, most of existing games have pre-specified strategy set for players, which does not consider the notion that attackers and defenders can learn. Second, many existing models have a clear criterion of winning where each player tries to win during the course of the game. However, in long-term attack and defense interactions in APTs, it is hard to specify who actually wins. Thus far, it has been challenging to quantify how much a player can learn during the course of an APT attack. Additionally, how to take advantage of past information and thus better defend against APT attacks is another challenge.

In this chapter, we proposed a two-player APT rivalry game to analyze the interactions between attackers and defenders. Rather than attempting to characterize the players' behaviors through a single game, we adopted a game-theoretic model to capture how much information each player can learn from their opponent's strategy adjustments and policy upgrades. Different from previous time-based game models, we consider that time itself imposes a cost on both players, even they do not take any new actions. In addition, the players in our model do not adjust their strategy at arbitrary time, they must consider the trade-off between time cost and information leakage. Our assumptions profoundly reflect today's attack-defense campaigns in APTs. That is, the adversary cannot be eliminated; instead, it can only be suppressed by more advanced techniques [22]. We also propose two reinforcement learning methods that help the defender to learn from past experiences and to select optimal defense levels for the future. These methods prescribe that the defender should quickly respond to the detected sabotage activities and set optimal defense levels. When faced with multiple defense points, the defender

must determine a reasonable plan for allocating the resources available. Our proposed mechanisms are capable of addressing these issues.

The main contributions of this chapter therefore include:

- An APT rivalry game that considers the information disclosed from a player's strategy adjustments. The defender and attacker are able to figure out the best timing for strategy adjustments based on the solution of the game.
- Two learning mechanisms based on reinforcement learning to help defenders find the optimal defense levels as well as appropriate resource allocations during a game against attackers.
- Theoretical proofs and analyses of the proposed game and mechanisms. We derived the necessary conditions for which the game has a family of equilibria with respect to the players' rationality. We also derive the regret bounds of our learning mechanism.
- Experimental simulations that show the existence and optimality of the equilibria under different game settings. Further experiments demonstrate the effectiveness of the proposed learning mechanisms.

In the rest of this chapter, we first introduce basic background knowledge of APTs and then detail the design and rationale of the proposed APT rivalry game. Theoretical analyses and experimental results are also given.

5.2 Preliminaries

5.2.1 The game model and Nash equilibrium

In this chapter, we take advantage of a dynamic timing game called the *war of attrition* [11]. This game characterizes the timing of strategy adjustments noting that the attacker and defender are trying to reduce information leaks to their opponent. Originally formulated to study animal conflicts and genetic evolution [88], the classic war of attrition game works as follow. Suppose players 1 and 2 are competing for a single resource. Both players can drop out at any time, if one player drops, the other player wins (i.e., obtains the resource) and the game ends immediately. The valuations of player 1 and player 2 are v_1 and v_2 respectively, representing the benefit they will obtain if they win the resource. Besides, each player suffers a cost c for fighting per unit of time. We

consider the mixed-strategy Nash equilibrium of the game, where the strategy of each player corresponds to a distribution T_i ($i = 1,2$) over drop-out times. Each player decides their drop-out time according to distribution T_i.

Definition 5.1. *(Nash equilibrium [102]) An action profile a^* is a Nash equilibrium if each a_i^* is the best response of a_{-i}^*. That is, $\forall i \in [n]$, $a' \in A$,*

$$u_i(a_i^*, a_{-i}^*) \geq u_i(a_i', a_{-i}^*). \tag{5.1}$$

For the war of attrition game, a player's utility is the difference between their valuation (if wins) and the cost for fighting. Namely, if the competition ends at time t, the winner gets $u_i = v_i - c \cdot t$, and the other player gets $-c \cdot t$. In a Nash equilibrium, both players are indifferent between dropping out at some time t and waiting to drop out at $t + \mathrm{d}t$. That is, at an equilibrium point, for player i,

$$v_i \cdot \frac{\mathrm{d}T_j(t)}{1 - T_j(t)} - c \cdot \mathrm{d}t = 0, \tag{5.2}$$

where $v_i \cdot \mathrm{d}T_j(t)/(1 - T_j(t))$ is the possible gain of player i when it persists in the game and $c \cdot \mathrm{d}t$ is the cost of staying in. The ratio $T_j(t)/(1 - T_j(t))$ represents the probability that player j drops out at time t, which is also known as the hazard rate of player j in the context of evolution theory.

By integrating, the unique mixed-strategy equilibrium of the war of attrition game is

$$T_i(t) = 1 - e^{-ct/v_j}, \quad i = 1,2. \tag{5.3}$$

In the context of APTs, a strategy adjustment of the attacker/defender corresponds to a player's drop out, which will leave a piece of information that values v_i to their opponent. We use the war of attrition model to figure out the optimal strategy adjustment timing in terms of information leakage. However, the valuations of players are private information of the attacker/defender in APT campaigns. Therefore, we consider a more complex scenario where the valuations of players are drawn from different distributions, and discuss player's expected utility when forming a Nash equilibrium. In addition, we extend the game to the situation where players are inertial (i.e., irrational with some probability), making it comparable to an APT attack. We also derive the necessary conditions for which a family of equilibria exists.

5.3 Problem definition and game design

5.3.1 Problem statement

This research mainly addresses two issues associated with APT attacks. The first is to develop a game-theoretic model for APT attacks to help attackers and defenders find the optimal times to adjust their strategies – with optimal being defined as the time when the least information possible will be leaked to avoid developing a strong opponent. The second is an effective method for the defender to select an optimal defense level, as well as find a reasonable resource allocation.

To defend against APTs, it is essential to find an optimal timing scheme for strategy adjustment in terms of information leakage. On the one hand, frequently making strategy adjustment not only results in huge resource consumption but may also impart information to one's attackers. On the other hand, an outdated defense strategy will make the system more vulnerable to novel attack variants. Therefore, a balance between the two is required.

We note that the term "information leakage" in this chapter refers to "what the defender/attacker can know by observing their opponent's new strategy adjustment", which enables them to improve their strategy or policy. We do not use information leakage to portray players' previously unseen cognition. The reason is that the players do not fully know each other, they cannot tell which information is new to their opponent.

In addition, knowing when to make a strategy adjustment, it is also necessary to explore ways for determining what is an appropriate defense level. There should be a tradeoff between resource consumption and system security, so determining an proper resource allocation is another factor to consider.

Formally, we consider the continuous-time rivalry game on a cyber network between an APT attacker and a defender. The attacker tries to invade a network and stay undetected, while the defender endeavors to secure every terminal point within the network and minimize the losses caused by the APT attack. c_a denotes the average cost to an attacker per unit of time, i.e., the cost generated by the attacker analyzing, moving around, and perpetrating malicious activity. Similarly, the defender also consumes time, money, and computational resources in defending against an APT attack via activities such as log monitoring or anomaly detection. c_d denotes the average cost to a defender per unit of time. To characterize the information leaks incurred in a strategy adjustment, v_a and v_d denote the value of the information learned by an attacker/defender from a strategy adjustment by their opponent.

The goal of the game is to find the optimal timing for strategy adjustments that leaks the least information. The aim of the learning method is to find an effective way for the defender to select an optimal defense strategy from past losses and experience.

5.3.2 Design rationale of the APT rivalry game

The course of the APT rivalry game is illustrated in Figure 5.2. There are two players $P = \{\text{attacker, defender}\}$ in the game. From the starting point $T = 0$ (e.g., the reconnaissance stage of the attack), the attacker and defender consume resources to perform attack/defense activities. The cost per unit of time for each player is c_a and c_d, respectively. At each instant, the players decide whether to adjust or change their strategy. One player's strategy adjustment leaves a piece of useful information to its opponent, which can be used in the opponent's subsequent strategy making. For example, at time $T = T_0$, the defender updates a blacklist of URLs. The attacker, whose list is outdated, will soon realize this fact and so she learns new information. The value of this information is v_a. During the course of the game, each player can freely choose the timing of when they adjust their strategy. After one player's strategy adjustment, the APT rivalry game goes to the next stage and thus the cost c_a and c_d also change. This is because attacking or defending against different strategies of the opponent possibly leads to different resource consumption.

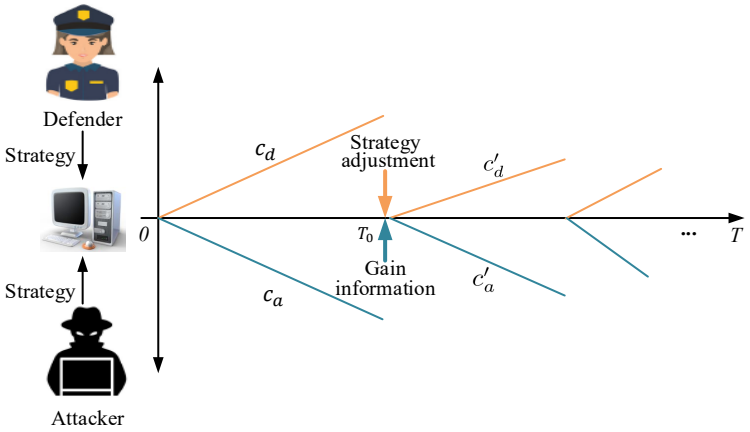


Figure 5.2: The APT rivalry game between attacker and defender.

In the proposed game, we consider the optimal timing of players' strategy adjustment in terms of information leakage. Here, the strategy adjustment refers to the players' routine strategy upgrades but does not necessarily involve emergency actions. For

example, if a defender finds an unknown program running within his system, he should take remedial action *immediately*, even if these actions will leak some side information to the attacker.

In this chapter, we assume that the attacker and defender are risk neutral and aim to maximize their utility with respect to an optimal timing scheme for strategy adjustment.

5.4 The solution of the game and learning mechanisms

5.4.1 The evaluation of players' information leakage

In the APT rivalry game, we use v_a and v_d to capture the value of information learned by an attacker/defender from a strategy adjustment by their opponent. However, in APT campaigns, it is challenging to evaluate the actual values v_a and v_d due to the uncertainty and asymmetry [105] of APTs. Sometimes, even the players themselves cannot foresee the effect of their new tactic [113]. To address this issue, we employ metrics from information theory to quantify the information leakage incurred during player's strategy adjustment, which serves as the basis for the computing v_a and v_d.

From the defender's perspective, let X_b be the random variable that captures the attacker's strategy adjustment detected by the defender, and X_r represents the risk that the cyber system is compromised. Then the mutual information between X_b and X_r is defined as

$$I(X_r;X_b) = H(X_r) + H(X_b) - H(X_r,X_b), \tag{5.4}$$

where $H(X) = -\sum_x \Pr[X=x]\cdot\log(\Pr[X=x])$ is the Shannon entropy of X.

The mutual information $I(X_r;X_b)$ measures the reduction of entropy in X_r caused by knowing X_b. In other words, it represents the defender's reduced uncertainty about the system's risk by observing the attacker's new strategy adjustment.

To estimate the value of v_d, let random variable $V \sim L(v)$ denote the loss of the defender if the cyber system is compromised. Then the value of information learned by the defender is modeled to be

$$v_d = V \cdot I(X_r;X_b). \tag{5.5}$$

We consider X_b and X_r to be random variables because the adjustment of attacker vectors involves changes on multiple defense points, the defender cannot capture all of

them exactly. In practice, X_b is formulated by the defender's observation of the attacker's movement, and the corresponding risk can be estimated from cyber security authorities like FireEye and NIST. Moreover, the potential loss V is also treated as a random variable as the defender is unaware of the attacker's stealthy goal. The distribution of $L(v)$ is formulated by the defender based on the value of the cyber system.

Similarly, from the attacker's perspective, the value of information v_a can be estimated in the same way, with each random variable has different interpretations, e.g., X_r is the success rate of the attacker, $V \sim L(v)$ is the potential gain if the targeted system is infiltrated.

Given $I(X_r; X_b)$ and distribution $L(v)$, the value of information v_d in fact follows a distribution $I \circ V$, where I is a scalar. To improve legibility, we denote $F = I \circ V$ and assume that the values v_a and v_d are drawn from distributions F_a and F_d, with density denoted as f_a and f_d. F_a and F_d are assumed to be differentiable from $[0, h] \to [0, 1]$.

5.4.2 The equilibrium strategy adjustment timing

In this section, we formulate the utility of attacker and defender, as well as deriving the sufficient conditions for the existence of equilibria in the APT rivalry game.

We first normalize the players' costs incurred in the game to $c_a = c_d = 1$, which is without loss of generality, since for nonnegative cost c and valuation v, it is only the ratio c/v that influences the equilibria. When players have different costs and valuations, we can interpret v as players' cost-valuation ratio. The result of the game will not change. Then, the equilibria $T_i(v_i)$ $(i = a, d)$ that specifies when the players adjust their strategies is a function of valuation v_i.

Lemma 5.1. *(Monotonicity and differentiability of $T_i(v_i)$) For $T_i(v_i)$ to be an equilibrium of the game, $T_i(v_i)$ should be strictly increasing and continuously differentiable.*

The intuition of Lemma 5.1 is that the player with larger v_i sticks to their current strategy longer as waiting longer may yield more valuable information about their opponent. In addition, it cannot be the case that T_i is constant on some interval $[v_i', v_i'']$ and then increasing after v_i''. Assuming not, for some $v_i' \le v_i''$, $T_0 = T_i(v_i') = T_i(v_i'') > 0$. That means $\Pr[T_i(v_i) = T_0] > 0$, i.e., player i would make an adjustment with a discrete probability at T_0. But, if this were the case, the opponent player j would never make an adjustment within $[T_0 - \epsilon, T_0]$ given a small ϵ, which contradicts $T_j(v_j)$'s continuity. Therefore, T_i must be strictly increasing. For a complete proof, we refer readers to the study of concession games [41, 97].

According to Lemma 5.1, we can ignore the possibility that both players will adjust their strategies simultaneously. Assume that the attacker follows the equilibrium $t_a \sim T_a(v_a)$ to make a strategy adjustment. Then, if the defender adjusts his strategy at time z, the expected utility of the defender will be

$$u_d(z;v_d) = \int_0^z (v_d - t_a)\mathrm{d}(F_a(\varphi_a(t_a))) - z(1-F_a(\varphi_a(z))), \tag{5.6}$$

where $\varphi_i(\cdot) = T_i^{-1}(\cdot)$ is the inverse function of $T_i(\cdot)$. So $\varphi_i(t)$ represents the valuation of a player who makes a strategy adjustment at time t. The second term can also be written as $(z+v_a)(1-F_a(\varphi_a(z)))$, which incorporates the attacker's valuation, both formulations leads to the same equilibria for the defender as long as the attacker is assumed to follow her equilibrium strategy. To simply the notation, we sometimes write $\varphi_i(t)$ as φ_i.

In Equation 5.6, the first term and second term are the defender's expected utility when the attacker makes a strategy adjustment at $t_a < z$ and $t_a > z$, respectively. Differentiating Equation 5.6 with respect to z, we have the following equation

$$\frac{\partial u(z;v_d)}{\partial z} = v_d F_a'(\varphi_a(z))\varphi_a'(z) - (1-F_a(\varphi_a(z))). \tag{5.7}$$

Set the derivative to zero, and we have

$$v_d F_a'(\varphi_a(z))\varphi_a'(z) = 1 - F_a(\varphi_a(z)). \tag{5.8}$$

Similarly, from the perspective of the attacker, we have

$$v_a F_d'(\varphi_d(z))\varphi_d'(z) = 1 - F_d(\varphi_d(z)). \tag{5.9}$$

Let $\langle \varphi_a(t), \varphi_d(t) \rangle$ denote the pair of valuation functions for the attacker and defender, then we have the main theorem of this chapter, as follows:

Theorem 5.1. *(The equilibria of the APT rivalry game) Suppose $\langle \varphi_a(t), \varphi_d(t) \rangle$ is a solution to the differential equation*

$$\begin{cases} \varphi_d(t) F_a'(\varphi_a(t))\varphi_a'(t) = 1 - F_a(\varphi_a(t)), \\ \varphi_a(t) F_d'(\varphi_d(t))\varphi_d'(t) = 1 - F_d(\varphi_d(t)). \end{cases} \tag{5.10}$$

with conditions $\lim_{t\to\infty} \varphi_a(t) = \lim_{t\to\infty} \varphi_d(t) = h$ and $\min\{\varphi_a(t), \varphi_d(t)\} = 0$, then $\langle \varphi_a(t), \varphi_d(t) \rangle$ forms an equilibrium of the APT rivalry game.

Theorem 5.1 characterizes the conditions where the APT rivalry game has a family of equilibria. In practice, the solution of the system is determined by different parameter

estimations of the players. We consider the defender and attacker's long-term competition. Every time they make a strategy adjustment, the parameters in the equation change, as does the solution of the next equilibrium timing.

We leave the complete proof of the theorem to Section 5.5.

5.4.3 The equilibrium with inertial players

In this section, we explore the change of equilibria when the attacker and defender are inertial. In the context of APTs, it is common that the attacker and/or defender cannot get complete information about the movements of their opponents during the reconnaissance, monitoring, etc. The asymmetry of information may influence the rationality of a player in APT rivalry games [120, 133], whom we refer to as an inertial player.

Hence, consider there is a probability $p > 0$ such that a player is inertial and thus will never make any attack/defense strategy adjustments during the game. The player's strategy adjustment time, viewed from its opponent's perspective, would then satisfy

$$\hat{F}_i(\varphi_i(t)) = (1-p)F_i(\varphi_i(t)). \tag{5.11}$$

Simply replace F_a and F_d with $\hat{F}_a$ and $\hat{F}_d$ in Theorem 5.1, and we have the equilibrium $\langle \hat{\varphi}_a(t), \hat{\varphi}_d(t) \rangle$ for inertial players.

Theorem 5.2. *(The equilibrium of APT game with inertial players) There exist a unique equilibrium of the APT rivalry game when players are inertial with probability $p > 0$.*

To see this, note that in the settings where there players are inertial, there is no longer a set of equilibrium. To see this, we rearrange Equation 5.10 and have

$$\frac{\hat{F}'_a(\varphi_a)}{\varphi_a(1-\hat{F}'_a(\varphi_a))} = \frac{\hat{F}'_d(\varphi_d)}{\varphi_d(1-\hat{F}'_d(\varphi_d))}. \tag{5.12}$$

And by integrating, we have

$$Y_a(\varphi_a) = Y_d(\varphi_d) + k, \tag{5.13}$$

where Y_i satisfies

$$\begin{aligned} Y_i(\varphi) &= \int_{\varphi}^{h} \frac{\hat{F}'_i(x)}{x(1-\hat{F}_i(x))}\mathrm{d}x < -\int_{h}^{\varphi} \frac{\hat{F}'_i(x)}{\varphi(1-\hat{F}_i(x))}\mathrm{d}x \\ &= \frac{1}{\varphi}\ln\frac{1-\hat{F}_i(\varphi)}{1-\hat{F}_i(h)}. \end{aligned} \tag{5.14}$$

According to Equation 5.11, $\hat{F}_i(\cdot) < 1$ and $Y_i(\varphi)$ converges as $\varphi \to h$. Based on the fact that $Y_i(\varphi)$ is a strict monotone function, Equation 5.13 has a unique solution. In other words, there is a unique equilibrium when players become inertial.

5.4.4 A case study of equilibrium computation

In this section, we presented a case study to illustrate how to find the equilibria in real-world APT scenario, including the evaluation of parameters and the solve of Nash equilibria.

Suppose a cyber system has three risk levels, namely X_r = {low, medium, high}. The attacker makes frequent queries on the target's publicly accessible repositories in an effort to collect domain/routing information and locate the websites that have high-risk vulnerabilities. The potential threats related to the attacker's strategy are cross-site scripting (XSS), SQL injections (SQLI) and DNS tunneling attacks (DNST), X_b = {XSS, SQLI, DNST}. The defender estimates the joint distribution of risks and potential attacks based on the system and well-known vulnerability databases like NIST National Vulnerability Database (NVD) [91]. The joint distribution of X_b and X_r are given in Table 5.1.

Table 5.1: The joint distribution of X_b and X_r.

	Low	Medium	High	Pr[X_b]
XXS	0.1	0.1	0.1	**0.3**
SQLI	0.1	0.2	0.1	**0.4**
DNST	0.1	0.1	0.1	**0.3**
Pr[X_r]	**0.3**	**0.4**	**0.3**	**1.0**

Based on the table, the defender can compute the mutual information $I(X_r; X_b) = 0.02$. Consider a simple case where the loss of the defender when the system is compromised follows a uniform distribution on interval $[0, 50]$, $V \sim U(0, 50)$. Then we have $F_d(v) = v$. Similarly, suppose $F_a(v) = v$, and the differential equation system in Theorem 5.1 becomes

$$\begin{cases} \varphi_d(t)\varphi_a'(t) = 1 - \varphi_a(t), \\ \varphi_a(t)\varphi_d'(t) = 1 - \varphi_d(t). \end{cases} \tag{5.15}$$

Rearrange the order, and we have

$$\frac{\mathrm{d}\varphi_a}{\mathrm{d}\varphi_d} = \frac{\varphi_a \cdot (1 - \varphi_a)}{\varphi_d \cdot (1 - \varphi_d)}. \tag{5.16}$$

Integrate the above equation, and we have the general solution

$$\ln(1/\varphi_d - 1) = \ln(1/\varphi_a - 1) + k, \tag{5.17}$$

The above equation defines a set of equilibria with a constant k. When $k = 0$, by symmetry, we have $\varphi_a = \varphi_d$, therefore, Equation 5.15 boils down to $\varphi_i \varphi_i' = 1 - \varphi_i$. By integrating and inverting, we have, for both the attacker and the defender,

$$T_i(v_i) = -v_i - \ln(1 - v_i), \tag{5.18}$$

which is the symmetric equilibrium of the game.

5.4.5 Resource allocation with reinforcement learning

In previous sections, we discussed when to make an attack/defense strategy adjustment in terms of information leakage and derived the strategy adjustment timing that forms a set of equilibria for both players. In the following sections, we further explore how to make a strategy adjustment when the equilibrium timing is known. Without loss of generality, the methods are presented from the perspective of the defender.

5.4.5.1 Learning problem formulation

To defend against APT attacks, multiple defense methods should be implemented, such as firewalls and anomaly detection. It is not always reasonable to exhaust every defense method to its utmost level as that would lead to significant consumption of the defender's resources. The defender generally distributes its recourses across these "battlefields", leading to a comprehensive defense level over the full cyber system. In this chapter, we assume that there are $P = \{P_i, \ldots, P_n\}$ vulnerable defense points. $\rho_i^d \in [0,1]$ is the specific defense level with respect to P_i, which represents the defender's resource commitment on P_i. For example, if P_i is the anti-virus scanning frequency, then $\rho_i^d = 1$ indicates that the defender sets the highest possible scanning frequency to protect his system. Similarly, the attacker also has attack level $\rho_i^a \in [0,1]$ on these defense points, which reflects the attacker's resource distribution over P. If $\rho_i^d \geq \rho_i^a$, we say that the defender successfully defends against one attack attempt in terms of P_i.

The defender's overall defense level is represented by k discrete numbers $L = \{l_1, \ldots, l_K\}$, where $l_k = \{k \in \mathbb{Z} | k \cdot n/K \leq \frac{1}{n} \sum_{i=1}^{n} \rho_i^d < (k+1) \cdot n/K\}$. Each strategy adjustment on a single defense point will eventually be reflected in the overall defense level.

The strategy adjustment of the defender follows. At any time t that forms an equilibrium of the APT rivalry game, the defender can modify or change a series of his defense

methods, such as the firewall level, the frequency of anomaly pattern detection, etc. These adjustments will change the defense level ρ_i^d on each defense point P_i, and further change the overall defense level of the cyber system, say from l_i to l_j with $i, j \in [K]$. Different overall defense levels will result in different losses to the defender, which influences the defender's next strategy adjustment in return.

In this chapter, we formulated the defender's strategy adjustment problem as a learning problem. We proposed two reinforcement learning based mechanisms to select the optimal defense level as well as specifying how to allocate the defender's resources across these different defense points.

5.4.5.2 The adversarial bandit solution

A natural solution to the strategy adjustment problem is to interpret it as an adversarial bandit problem, where an adversary (the attacker) has a complete control of the loss suffered by the defender and the defender only gets feedback from the defense actions it had taken before.

In the adversarial bandit setting, $l_{i_1}, \dots l_{i_t}$ denote the defense levels selected by the defender from time 1 to t. Let $x_{i_1}(1) \dots, x_{i_t}(t)$ be the sequence of losses suffered by the defender. At time t, the defender computes the loss he has suffered from his previous overall defense level $l_{i_{t-1}}$, and then decides his new defense level based on the knowledge it obtained so far, i.e., the history $\{l_{i_1}, x_{i_1}(1), \dots, l_{i_{t-1}}, x_{i_{t-1}}(t-1)\}$. Intuitively, the loss on defense point P_i is computed by $x_{P_i}(t) = |\rho_i^d(t) - \rho_i^a(t)|$. That is, if $\rho_i^d(t) < \rho_i^a(t)$, the defender has failed to defend P_i and suffers $\rho_i^a(t) - \rho_i^d(t)$. If $\rho_i^d(t) \geq \rho_i^a(t)$, the defender has succeed in his defense but has wasted $\rho_i^d(t) - \rho_i^a(t)$ resources, which might have been used on P_j to further improve the security of the system. The overall loss is therefore $x_{i_t}(t) = \frac{1}{n} \sum_{i=1}^{n} x_{P_i}(t)$. Assume that the K defense levels are common knowledge to the attacker and defender. The attacker specifies a loss vector $(x_1(t), \dots, x_K(t))$ by distributing its resources across the set of defense points P, where $x_k(t) \in [0,1]$ represents the loss of the defender if its overall defense level falls to $k \in [K]$.

For any time $T > 0$ and any mechanism $\mathscr{M}$ that maps from the defender's history $\{l_{i_1}, x_{i_1}(1), \dots, l_{i_{t-1}}, x_{i_{t-1}}(t-1)\}$ to the set of defense levels $L = \{1, \dots, K\}$, the total loss generated by the mechanism along the time horizon T is

$$G_{\mathscr{M}}(T) = \sum_{t=1}^{T} x_{t_j}(t). \tag{5.19}$$

In addition, for a best single defense level j^*, the loss it leads to is

$$G_{min}(T) = \min_{j} \sum_{t=1}^{T} x_{j^*}(t). \tag{5.20}$$

In this chapter, we consider the (weak) regret of the mechanism $\mathscr{M}$, which is the gap between $G_{\mathscr{M}}(T)$ and $G_{min}(T)$,

$$R(T) = G_{\mathscr{M}}(T) - G_{min}(T). \tag{5.21}$$

Regret is a good way to measure the performance of a learning algorithm. It represents the loss suffered by the defender relative to the optimal fixed defense level.

We assume that the defender is risk-neutral and, thus, the objective of the defender is to obtain a bounded expected regret. The exponential-weight algorithm for exploration and exploitation (Exp_3) [4] with losses is used to find the best single defense level. The original Exp_3 algorithm was developed with respect to players' rewards. However, we reformulated this algorithm to derive the regret bound in terms of the defender's losses. The details are shown in Algorithm 9.

The algorithm takes as its input the set of defense levels L. The parameter γ controls the trade-off between exploration (trying out a new defense level to find the optimal one) and exploitation (playing the best defense level so far to have minimized loss). The algorithm starts by assigning each defense level a weight $w_i(1) = 1$ (Line 1-3). At each time step t, it computes a probability mass for each defense level and draws a particular one accordingly (Line 5-6). The defender then adjusts his defense strategy following the selected defense level. The next time that it is necessary to make a strategy adjustment, the defender estimate the loss he suffered, and updates the weight for each defense level (Line 7-11). In the weight update process (Line 9), the algorithm uses the estimated loss rather than the exact one, which aims to punish the defense levels that lead to a large loss. A detailed analysis of the regret is presented in Section 5.5.

5.4.5.3 The deep Q-network solution

The previous solution endeavors to find the best fixed defense level, but it does not provide a detailed resource allocation scheme for a specific defense point. This issue is addressed in the second solution.

The second solution takes advantage of deep reinforcement learning, where the knowledge of the defender is stored in a deep neural network. Compared to regular reinforcement learning, deep Q-networks (DQNs) [92] can handle more complex problems

Algorithm 9 Mechanism $\mathscr{M}_1$: Exp_3 algorithm with losses

Input: the defense levels $L = \{l_1, \ldots, l_K\}$, parameter $\gamma \in [0,1]$.
Output: defense levels l_{i_t} for each t.

for $i \in [K]$ **do**
  initialize weight $w_i(1) \leftarrow 1$ for defense level l_i.
end for
for $t = 1, 2, \ldots$ **do**
  Set probability

$$p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K},\ i = 1, \ldots, K.$$

  draw a defense level l_{i_t} randomly according to the distribution of $p_i(t)$.
  observe the loss $x_{i_t}(t)$.
  for $j \in [K]$ **do**
    update the weight for defense level l_j by

$$\hat{x}_j(t) = \frac{\mathbb{I}\{j = i_t\} x_j(t)}{p_j(t)};$$

$$w_j(t+1) = w_j(t) \exp(-\frac{\gamma \hat{x}_j(t)}{K}).$$

  end for
end for

with a large number of states or actions. In regular reinforcement learning, an agent maintains a Q-table to track changes in the state-action pairs $Q(s,a)$. However, there is no limit on how much the Q-table can expand, and so it will become hard to update with a large number of states or actions. By contrast, deep reinforcement learning adopts a neural network to approximate the Q-table, which is much easier to maintain.

In an APT attack, the defender may not be aware of the attack model. As a model-free reinforcement learning technique, a DQN can be employed to derive the optimal defense policy (i.e., the ρ_i^d and l_{i_t}) in a Markov decision process (MDP). To avoid ambiguity, in this section, the strategy of the defender indicate the defender's resource allocation across different defense points, while in previous sections strategy has referred to the actual defense methods the defender has used to protect the system.

In the proposed DQN solution, the defender's strategy in current round t is based on his strategies and losses in the last τ rounds. The state of the defender is $s_t = \{o_{t-\tau}, \ldots, o_{t-1}\}$, where $o_{t-j} = \left(\{\rho_i^d(t-j)\}, \{\rho_i^a(t-j)\}\right)$ for $i \in [K]$, $j \in [\tau]$. That is, a state consists of a series of observations, and each observation is made up of the defender's

specific defense levels over P and the attacker's attack levels over P in a single round. The defender's actions are defined as the defense levels $\rho_i^d(t)$ available for selection. We assume that each defense point P_i has an importance factor I_i, which indicates how important P_i is in protecting the cyber system. The reward of the defender in round t is therefore $r(t) = \sum_{i=i}^{n} I_i \cdot \text{sign}(\rho_i^d(t) - \rho_i^a(t))$.

We also employ a new strategy-based sampling approach to select samples from the experience replay set. The existing sampling approaches, such as random sampling [93] and gradient-based [160] sampling are not suitable for the proposed APT game as the defender may have a large number of available actions. To avoid overfitting, defenders should avoid selecting the samples that have similar defense levels. Therefore, the similarity between strategies must be considered.

Given the experience replay set D, Euclidean distance is used to measure the similarity between strategies. Specifically, the defender first computes the average of the specific defense levels within D, denoted as $\bar{\rho}^d(j)$,

$$\bar{\rho}^d(j) = \frac{1}{|D|} \sum_{j=1}^{|D|} \rho^d(j), \quad \rho^d(j) = (\rho_i^d(j))_{i=1}^K. \tag{5.22}$$

Then, for each candidate defense level $\rho^d(j)$, the defender computes the Euclidean distance between $\rho^d(j)$ and $\bar{\rho}^d(j)$,

$$Ed(\rho_i, \bar{\rho}) = ||\rho^d(j) - \bar{\rho}^d(j)||_2. \tag{5.23}$$

Finally, the defender selects m samples that have the largest distance according to the order of $Ed(\rho^d(j), \bar{\rho}(j))$.

The DQN based learning mechanism is shown in Algorithm 10. In each round t, the defender adopts an ϵ-greedy approach to select a defense strategy (Line 4). The defender then achieves the selected defense levels by adjusting a series of defense methods at each defense point. Here, the reward is also estimated and a new observation is obtained (Line 5-6). The defender then updates the observation for his state (Line 10), and puts the sample with the updated state into the experience replay set (Line 11). Finally, the defender selects m samples with which to train the DQN by minimizing the loss function defined by the difference between target Q-values and learned Q-values (Line 12-17).

5.4.5.4 Discussion and comparison of the two solutions

The adversarial bandit based solution allows the defender to learn a defense level fast by only accessing its incurred losses. However, we note that in such setting the adversary

Algorithm 10 Mechanism $\mathscr{M}_2$: Deep Q-network

Input: the defender's state, weight θ of the deep Q network, experience replay set D.
Output: a trained DQN.

initialize weight θ and Q-values of each defense strategy randomly.
set $D = \varnothing$.
for $t = 1, 2, \ldots$ **do**
    With probability ϵ, randomly select a defense strategy $\rho^d(t) = \{\rho_i^d(t)\}_{i=1}^K$, otherwise select $\rho^d(t) = \arg\max_\rho Q(s_t, \rho; \theta)$.
    adjust defense methods on defense points $P = \{P_i\}_{i=1}^K$ to achieve the selected defense strategy.
    obtain the reward $r(t)$ and new observation o_{t+1}.
    input $(s_t, \rho^d(t))$ to the DQN.
    obtain the output vector $Q(s_t)$ from DQN.
    update state $s_{t+1} \leftarrow s_t \cup \{o_{t+1}\} - \{o_{t-\tau}\}$.
    put sample $(s_t, \rho^d(t), r(t), s_{t+1})$ into D.
    select m samples from D.
    for $j \in [m]$ **do**
        $Q_j \leftarrow r(j) + \eta \cdot \max_\rho Q(s_{j+1}, \rho; \theta)$
    end for
    update weight θ by gradient descent on loss function $\frac{1}{m}\sum_{j=1}^m [Q_j - Q(s_j, \rho^d(t); \theta)]^2$.
end for

(attacker) is generally assumed to be oblivious, i.e., the losses $x_i(t)$ are generated without considering the past actions of the learner (defender). However, the result obtained here in this chapter can be generalized to the non-oblivious setting without changing the main analysis. In comparison, the DQN based solution considers the past τ observations of the defender, which involves the previously detected stealthy movements and activities on the part of the attacker. Therefore, this method may be more suitable for defenders with the ability to monitor or track the attacker's actions.

5.5 Theoretical analysis

This section presents a theoretical analysis of the proposed APT defense framework, including the game-theoretic properties it provides and the regret bounds it guarantees with respect to different defense strategies.

5.5.1 Proofs of theorems

We first present the complete proof for Theorem 5.1, i.e., any solution that satisfies the differential Equation 5.10 forms a Nash equilibrium of the APT rivalry game.

Theorem 5.1. Proof. We first prove the optimality of $\langle \varphi_a(t), \varphi_d(t) \rangle$. Taking Equation 5.10 into Equation 5.7, we have

$$\frac{\partial u(z; v_d)}{\partial z} = \frac{1 - F_a(\varphi_a(z))}{\varphi_d(z)} \cdot (v_d - \varphi_d(z)) \tag{5.24}$$

By previous argument $\varphi_a(t)$ is strictly increasing. Therefore, for any $0 < z < h$, $1 - F_a(\varphi_d(z)) > 0$. That means

$$(v_d - \varphi_d(z)) \frac{\partial u(z; v_d)}{\partial z} \geq 0. \tag{5.25}$$

In addition, the above equality holds if and only if $v_d = \varphi_d(z)$; otherwise, the inequality is strictly positive. Hence, the defender's best response to the attacker is to choose a z_d^* such that $\varphi_d(z_d^*) = v_d$. A symmetric argument demonstrates that $\varphi_a(z_a^*) = v_a$ is indeed the best response for the attacker.

Next, we consider the endpoint of the equilibrium functions. First, observe that it cannot be the case that $\varphi_a(0) = \varphi_d(0) = 0$. Otherwise, both players would have a positive discrete probability of strategy adjustment at $T = 0$. If this were the case, each player would have an incentive to wait infinitesimally so as to gain for extremely small cost. For similar reasons, the attacker and defender must have the same maximum strategy adjustment time t_{max}. That is, at the upper endpoint,

$$\varphi_d(t_{max}) = \varphi_a(t_{max}) = h. \tag{5.26}$$

To complete the proof, we show that, at the upper endpoint, $\varphi_d(t_{max}) = \varphi_a(t_{max}) = h$ for some $t_{max} \in (0, \infty)$.

Suppose this were not the case, which implies that $\varphi_a(t)$ and $\varphi_d(t)$ are bounded away from h. Then, consider any interval $[v, w] \subset (0, \infty)$. Since $\varphi_a(t)$ is increasing, according to Equation 5.10 we have

$$\frac{\varphi_a(v) F_d'(\varphi_d(t)) \varphi_d'(t)}{1 - F_d(\varphi_a(t))} > 1 > \frac{\varphi_d(w) F_d'(\varphi_d(t)) \varphi_d'(t)}{1 - F_d(\varphi_a(t))}. \tag{5.27}$$

Integrating over interval $[v, w]$ it follows that

$$\varphi_a(w) \ln\left(\frac{1 - F_d(\varphi_a(v))}{1 - F_d(\varphi_a(w))}\right) > w - v > \varphi_a(v) \ln\left(\frac{1 - F_d(\varphi_a(v))}{1 - F_d(\varphi_a(w))}\right). \tag{5.28}$$

If $\varphi_a(t)$ and $\varphi_d(t)$ are bounded away from h, then the left expression is bounded, which contradicts the fact that w in the middle expression can be arbitrarily large.

Then suppose $\varphi_d(\bar{t}_d) = h$. From the second inequality, we have that when $w \to \bar{t}_d$, the right expression becomes unbounded. This implies $\bar{t}_d = \infty$. By a symmetric argument for the attacker, we have $\bar{t}_a = \infty$. The above analysis shows

$$\lim_{t\to\infty} \varphi_a(t) = \lim_{t\to\infty} \varphi_d(t) = h, \tag{5.29}$$

Therefore Equation 5.26 holds. This completes the proof. □

We then explore the change of the defender's regret led by mechanism $\mathscr{M}_1$ and derive the upper bound of regret as a function of learning rounds.

Theorem 5.3. *Mechanism $\mathscr{M}_1$ guarantees that*

$$\mathbb{E}[G_{\mathscr{M}_1}] - G_{min} \leq \frac{K \ln K}{\gamma} + (e-2)\gamma T, \tag{5.30}$$

for any $K > 0$, $T > 0$ and $\gamma \in [0,1]$, where $e = 2.718\ldots$ is the Euler's number.

Proof. Let $\Gamma(t) = \sum_{i=1}^{K} w_i(t)$ denote the sum of the weights at time t. Suppose the loss sequence generated by mechanism $\mathscr{M}_1$ is $\{l_{i_1}, \ldots, l_{i_t}\}$. Then let us consider the change of total weight between time t and $t+1$.

$$\begin{aligned}
\frac{\Gamma(t+1)}{\Gamma(t)} &= \sum_{i=1}^{K} \frac{w_i(t+1)}{\Gamma(t)} \\
&= \sum_{i=1}^{K} \frac{w_i(t)}{\Gamma(t)} \exp(-\frac{\gamma \hat{x}_i(t)}{K}) \\
&= \sum_{i=1}^{K} \frac{p_i(t) - \gamma/K}{1-\gamma} \exp(-\frac{\gamma \hat{x}_j(t)}{K}) \\
&\leq \sum_{i=1}^{K} \frac{p_i(t)}{1-\gamma} \left[1 - \frac{\gamma \hat{x}_j(t)}{K} + (e-2)\left(\frac{\gamma \hat{x}_j(t)}{K}\right)^2 \right] \\
&\leq 1 - \frac{\gamma/K}{1-\gamma} x_{i_t}(t) + \frac{(e-2)\gamma^2/K^2}{1-\gamma} \sum_{i=1}^{K} \hat{x}_i(t).
\end{aligned} \tag{5.31}$$

The first three equations are obtained from the definitions of $w_i(t)$ and $p_i(t)$. The first inequality uses the fact that $e^{-x} \leq 1 - x + (e-2)x^2$ for $x \in [0, \infty)$. The last inequality is based on observations that

$$\sum_{i=1}^{K} p_i(t)\hat{x}_i(t) = p_{i_t}(t)\frac{x_{i_t}(t)}{p_{i_t}(t)} = x_{i_t}(t); \tag{5.32}$$

$$\sum_{i=1}^{K} p_i(t)\hat{x}_i(t)^2 = p_{i_t}(t)\frac{x_{i_t}(t)}{p_{i_t}(t)}\hat{x}_{i_t}(t) \leq \hat{x}_{i_t}(t) = \sum_{i=1}^{K} \hat{x}_i(t). \tag{5.33}$$

We then take the logarithms of Equation 5.31, and use the inequality $e^x \geq 1+x$ to have

$$\ln \frac{\Gamma(t+1)}{\Gamma(t)} \leq -\frac{\gamma/K}{1-\gamma} x_{i_t}(t) + \frac{(e-2)\gamma^2/K^2}{1-\gamma} \sum_{i=1}^{K} \hat{x}_i(t). \tag{5.34}$$

By induction, we can write

$$\ln \frac{\Gamma(t+1)}{\Gamma(1)} \leq -\frac{\gamma/K}{1-\gamma} G_{\mathscr{M}_1} + \frac{(e-2)\gamma^2/K^2}{1-\gamma} \sum_{t=1}^{T} \sum_{i=1}^{K} \hat{x}_i(t). \tag{5.35}$$

On the other hand, for any defense level l_j, we have

$$\ln \frac{\Gamma(t+1)}{\Gamma(1)} \geq \ln \frac{w_j(t+1)}{\Gamma(1)} = -\frac{\gamma}{K} \sum_{t=1}^{T} \hat{x}_j(t) - \ln K. \tag{5.36}$$

Combining the two bounds in Equation 5.35 and Equation 5.36, we have

$$G_{\mathscr{M}_1} - \frac{(e-2)\gamma}{K} \sum_{t=1}^{T} \sum_{i=1}^{K} \hat{x}_i(t) \leq (1-\gamma) \sum_{t=1}^{T} \hat{x}_j(t) + \frac{(1-\gamma)K \ln K}{\gamma}.$$

Note that $\hat{x}_i(t)$ is an unbiased estimator of $x_i(t)$, namely

$$\mathbb{E}_{i \sim p_i(t)}[\hat{x}_i(t)] = \mathbb{E}\left[p_i(t) \cdot \frac{x_i(t)}{p_i(t)} + 0 \right] = x_i(t). \tag{5.37}$$

So, taking the expectation of both sides in Inequality 5.37, we have

$$\begin{aligned} \mathbb{E}[G_{\mathscr{M}_1}] - G_{min} &\leq \frac{(1-\gamma)K \ln K}{\gamma} + \frac{(e-2)\gamma}{K} \sum_{t=1}^{T} \sum_{i=1}^{K} x_i(t) \\ &\leq \frac{(1-\gamma)K \ln K}{\gamma} + \frac{(e-2)\gamma}{K} \cdot TK \\ &\leq \frac{K \ln K}{\gamma} + (e-2)\gamma T. \end{aligned} \tag{5.38}$$

This complete the proof. □

Based on the above theorem, we can actually derive the upper bound of the expected regret by minimizing the two error terms in Equation 5.38.

Theorem 5.4. *The expected regret of mechanism $\mathscr{M}_1$ is at most $2\sqrt{e-2}\sqrt{TK \ln K}$.*

Proof. There are two error terms in Equation 5.38. The first one corresponds to the learning inaccuracy – the larger the γ, the higher the γT. The second term relates to the learning overhead – the smaller the γ, the higher the $\frac{K \ln K}{\gamma}$. Choosing $\epsilon = \sqrt{\frac{K \ln(K)}{(e-2)T}}$ to equalize the two error terms, we have

$$\mathbb{E}[G_{\mathscr{M}_1}] - G_{min} \leq 2\sqrt{e-2}\sqrt{TK \ln K}. \tag{5.39}$$

That is, the average loss of mechanism $\mathscr{M}_1$ approaches the average loss of the best fixed defense level at a rate of $\sqrt{T}$. This completes the proof. □

5.5.2 Discussions and the security insights of the theorems

Theorem 5.1 captures the nature of informational asymmetry in APT rivalry game. It proves the existence of a family of equiilbria characterized by differential equations. From a security standpoint, this theorem demonstrates that players' strategy adjustment timing does matter in terms of information leakage, which helps to explain why constantly upgrading security policies may not always adequately defend an organization from being compromised [22, 117]. Theorem 5.1 provides suggestions for the defender in their routine practical strategy upgrades by indicating the optimal strategy adjustment timing in the light of Nash equilibrium.

Meanwhile, Theorem 5.3 and 5.4 state that the proposed defending algorithm leads to an expected regret bound of $O(\sqrt{T})$. Regret measures how much the defender regrets, in hindsight, of not having followed the algorithm's advice. The security interpretation of this regret bound is that it limits the gap between the real loss suffered by the defender and the ideal loss in theory. The regret will grow at a logarithmic rate if the defender distributes his or her resources following the algorithm's recommendation. These theorems provides a theoretical guarantee for the defender to dynamically adjust defending strategy against APTs.

5.6 Experiment and analysis

In this section, we evaluate the performance of the proposed methods. We first examine the game-theoretic properties of the APT rivalry game, investigating the existence of Nash equilibria with different valuation distributions and rationalities for each of the players. Then we look into the learning mechanisms to see whether the defender can learn a near optimal defense strategy using the proposed learning scheme. The experimental results as well as corresponding analyses are presented in each subsection.

5.6.1 Experiments for the APT game

5.6.1.1 Experimental setup

We conducted the experiments with three representative distributions from which the attacker and defender draw their valuations independently. The three distributions are given in Equation 5.40, where F_1 is a uniform distribution in support $[0,1]$, F_2 is an exponential distribution with the parameter $\lambda = 1$ in $[0,\infty)$, and F_3 is a distribution with a probability density of $f_X(x) = 2x$, $x \in [0,1]$. The distributions F_2 and F_3 are monotone

decreasing and monotone increasing in their support, which represent the players' different beliefs in the probability of getting low/high-value information.

$$\begin{aligned} &F_1(v) = v; \quad v \in [0,1] \\ &F_2(v) = 1 - e^{-v}, \quad v \in [0,\infty); \\ &F_3(v) = v^2, \quad v \in [0,1]. \end{aligned} \tag{5.40}$$

For a clear and simple presentation, we only considered symmetric equilibria in the experiment. The symmetric equilibria derived from Theorem 5.1 with respect to the three distributions are given in Equation 5.41.

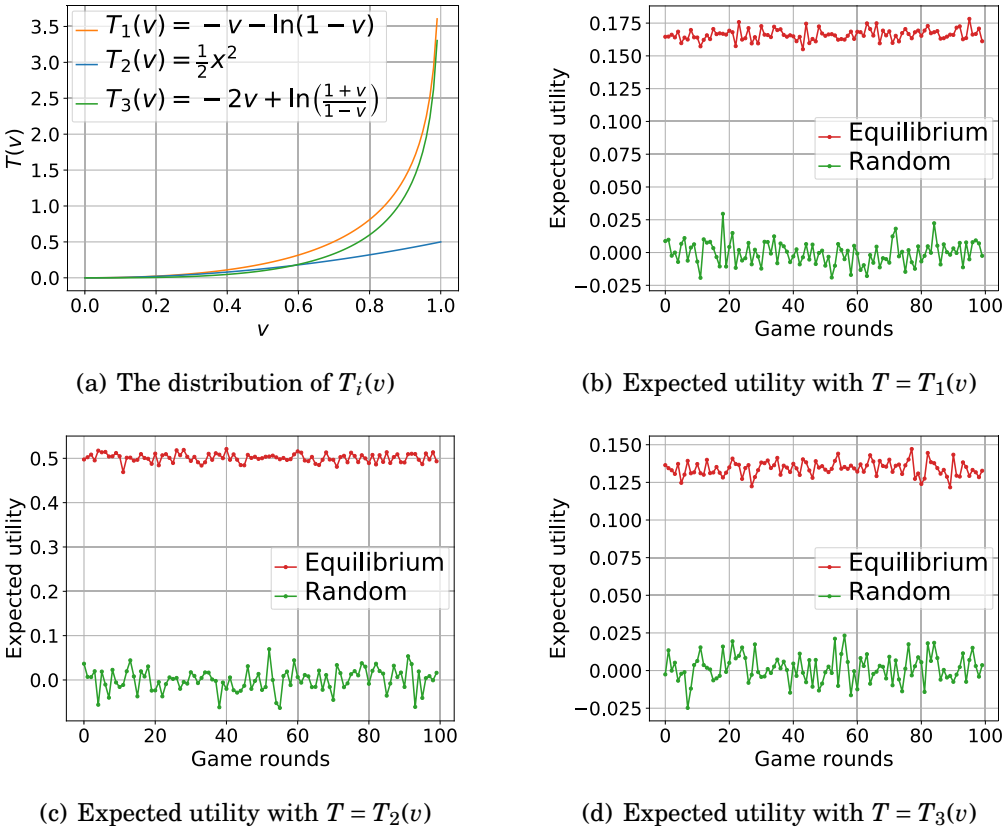
$$\begin{aligned} &T_1(v) = -v - \ln(1-v); \quad v \in [0,1] \\ &T_2(v) = \frac{1}{2}v^2, \quad v \in [0,\infty); \\ &T_3(v) = -2v + \ln\left(\frac{1+v}{1-v}\right), \quad v \in [0,1]. \end{aligned} \tag{5.41}$$

The results are shown in Figures 5.3 to 5.5.

5.6.1.2 Results and analyses

Figure 5.3 illustrates the change of the defender's expected utility when the attacker and defender are fully rational. Figure 5.3(a) depicts the distributions of strategy adjustment timing indicated by the Nash equilibria. Overall, the three distributions increase in their support, which is consistent with the properties of $T_i(v)$ in Lemma 5.1. Figures 5.3(b) to 5.3(d) show the defender's expected utility in 100 rounds of the APT rivalry game, given the players' valuations were drawn from $T_1(v)$, $T_2(v)$, $T_3(v)$, respectively. From the figures, we can see that the defender's expected utility in the equilibrium is always higher than that with the random strategy. The expected utility with the random strategy fluctuates around 0, while the utility in the equilibrium strategy is strictly positive. In addition, although the exponential distribution $F_2(v)$ decreases in $[0,\infty)$, this provides a chance for the defender to get high-value information from his opponent's strategy adjustment. Therefore, the defender's expected utility when the players' valuations are drawn from the exponential distribution is higher than that with other two distributions.

Figure 5.4 demonstrates the defender's expected utility when the players are partially rational, given their valuations were drawn from the uniform distribution $F_1(v)$. Figure 5.4(a) shows the variation tendency of the equilibrium timing $T_1(v)$ with different parameters p. From the figure, we observe that when players are fully rational ($p = 0$),



(a) The distribution of $T_i(v)$

(b) Expected utility with $T = T_1(v)$

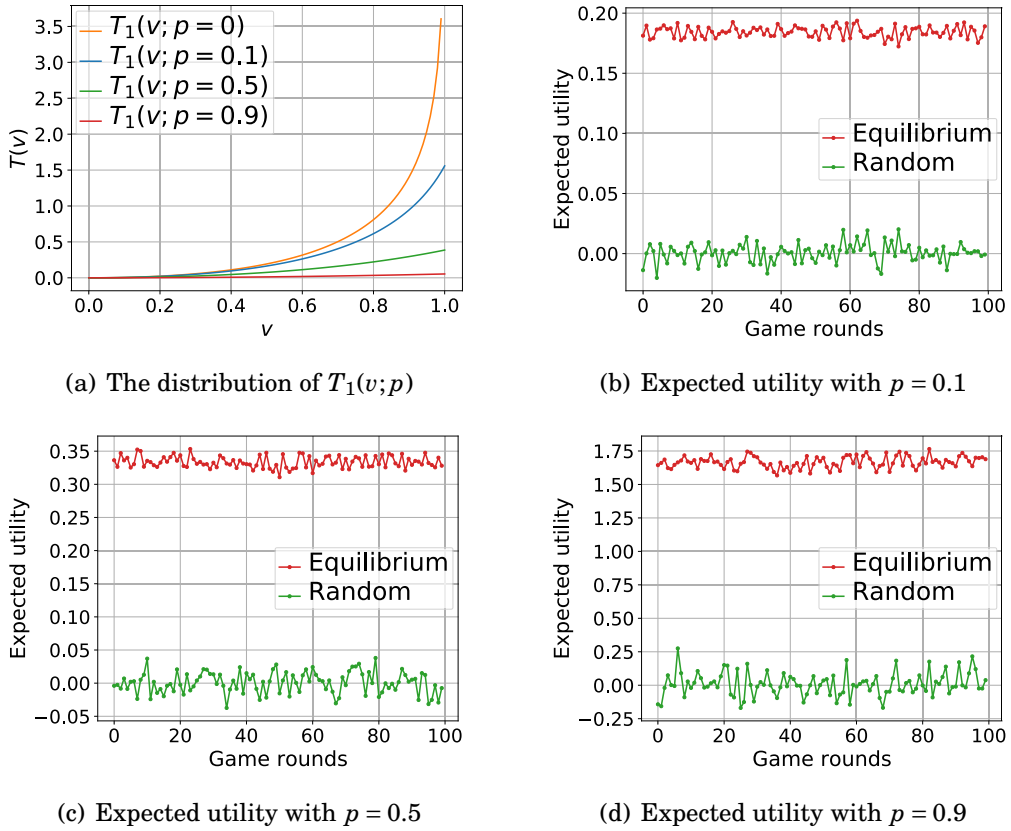
(c) Expected utility with $T = T_2(v)$

(d) Expected utility with $T = T_3(v)$

Figure 5.3: The fully rational ($p = 0$) defender's expected utility under equilibrium with respect to different valuation distributions.

the equilibrium timing goes to infinity as v approaches 1. In the symmetric equilibrium, this indicates that, if the defender thinks the next strategy adjustment of its opponent will leave valuable information, then he will wait for a longer period of time before making this strategy adjustment. However, as players become inertial, the probability that they never make a strategy adjustment increases. As a result, the strategy adjustment timing in the equilibrium goes down as p increases. Note that when $p = 0.1$, 0.5 and 0.9, the value of $T_1(v;p)$ at $v = 1$ is no longer infinity. This indicates that if players are inertial, they may also make a strategy adjustment immediately even if waiting for a while is likely to provide them with valuable information about their opponent.

Figures 5.4(b) to 5.4(d) present the expected utility of the defender under different rationality settings. In these figures, the expected utility in the equilibrium strategy is still higher than that in the random strategy. In addition, although the value of the expected utility in equilibrium increases with an increase of p, these values can only be obtained under the circumstance that players happen to be rational. Therefore, the



(a) The distribution of $T_1(v;p)$

(b) Expected utility with $p = 0.1$

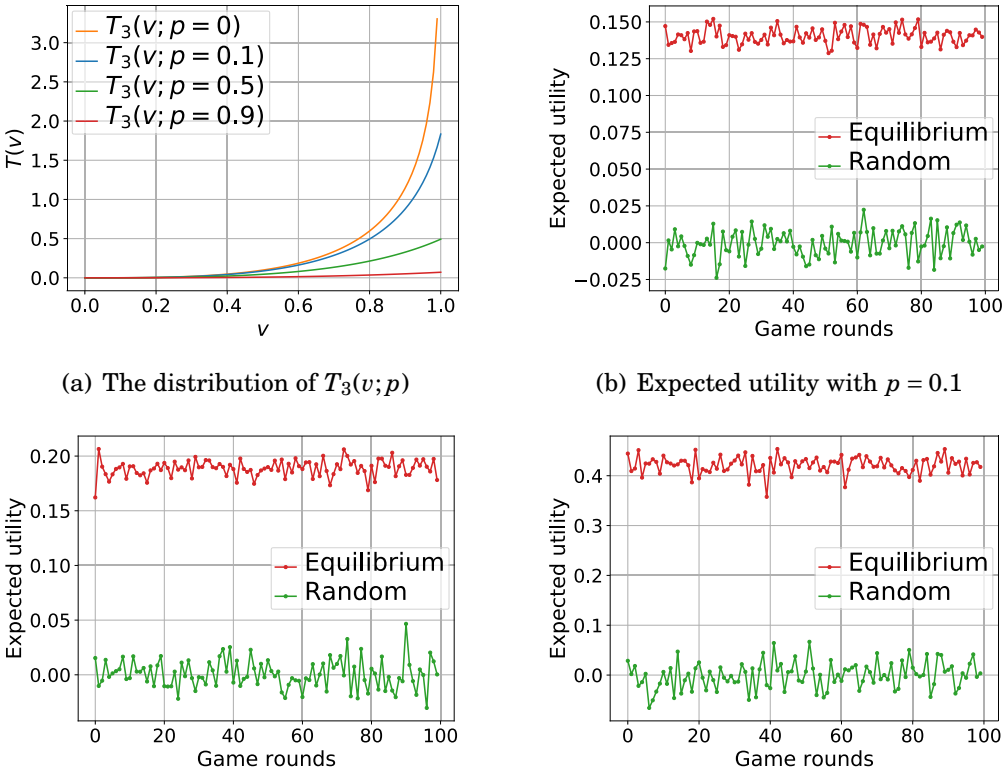
(c) Expected utility with $p = 0.5$

(d) Expected utility with $p = 0.9$

Figure 5.4: The inertial ($p = 0.1, 0.5, 0.9$) defender's expected utility under equilibrium with respect to valuations sampled from distribution F_1.

overall expected utility still decreases as the players become more and more inertial.

Figure 5.5 shows the results when the players' valuations were drawn from an increasing distribution $F_3(v)$. Figure 5.5(a) describes the change of the equilibrium timing $T_3(v;p)$ with respect to different values of p. Figures 5.5(b) to 5.5(d) illustrate the detailed expected utility of the defender after 100 rounds of the ATP rivalry game. The result patterns are similar to those in Figure 5.4. From the figures, we can see that, although distributions $F_1(v)$, $F_2(v)$, $F_3(v)$ vary in their support, the equilibrium timing $T_1(v)$, $T_2(v)$, $T_3(v)$ fall into similar patterns. In addition, the defender's expected utility is always higher with Nash equilibria, showing that the Nash's solution leads to the optimal result in the proposed APT rivalry game.



(a) The distribution of $T_3(v;p)$

(b) Expected utility with $p = 0.1$

(c) Expected utility with $p = 0.5$

(d) Expected utility with $p = 0.9$

Figure 5.5: The inertial ($p = 0.1, 0.5, 0.9$) defender's expected utility under equilibrium with respect to valuations sampled from distribution F_3.

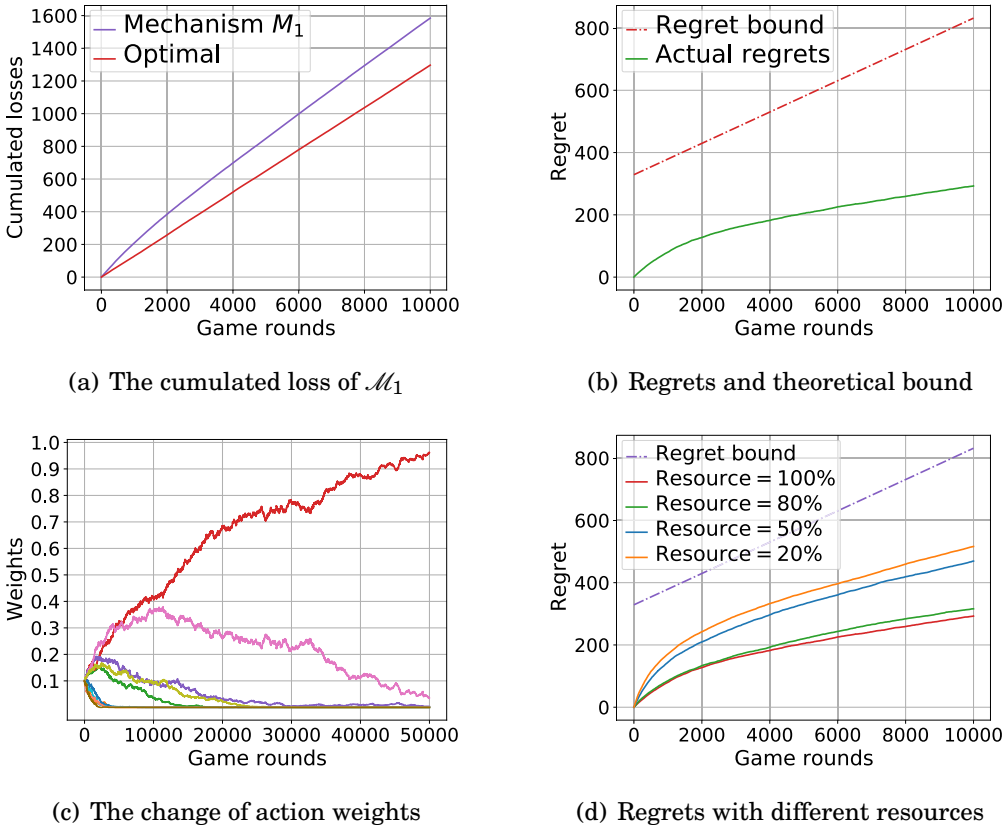
5.6.2 Experiments for the learning mechanisms

5.6.2.1 Experimental setup

We also conducted experiments to investigate whether the defender can learn a near optimal defense strategy through the proposed learning mechanisms. For mechanism $\mathscr{M}_1$, we assumed that there are $|P| = 10$ defense points, across which the defender and attacker can allocate their defense/attack resources. The total amount of their resources was fixed, and the resources spent on each defense point was represented as a real numbers $\rho_i^a, \rho_i^d \in [0,1]$. In the experiment, we set different defense levels $|L|$ as well as changing the amount of defense resources used by the defender. We then looked into how the defender's regret changed when mechanism $\mathscr{M}_1$ was used to select the defense levels, and we also examined the robustness of mechanism $\mathscr{M}_1$ when the defender had limited resources with which to defend against the attacker.

For mechanism $\mathscr{M}_2$, we assumed that the defender and the attacker had a certain

amount of defense/attack resources, which could be freely allocated to each defense point. In this scenario, the defender not only cared about the overall defense level of the system, but also needed to pay attention to the specific resource allocations for each defense point. To this end, we varied the defender's resources to see the corresponding performance in the proposed mechanism. We also varied the number of defense points and the distribution of importance factor to check the mechanism's robustness. The results are presented in Figures 5.6 to 5.8.



(a) The cumulated loss of $\mathscr{M}_1$

(b) Regrets and theoretical bound

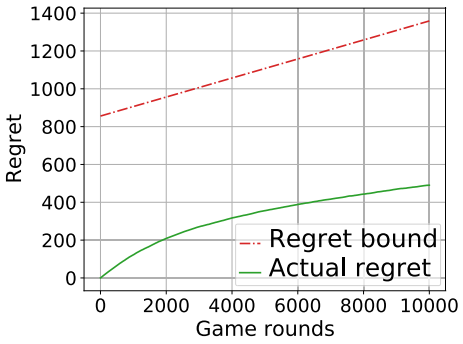
(c) The change of action weights

(d) Regrets with different resources

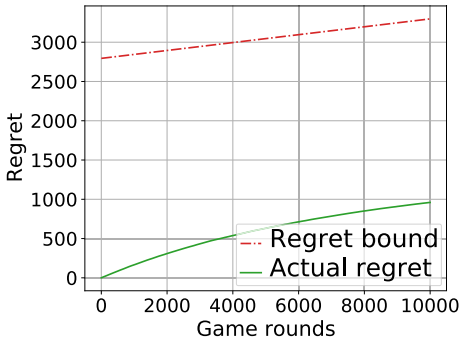
Figure 5.6: The performance of mechanism $\mathscr{M}_1$ with $|L| = 10$ actions and different defense resources of the defender.

5.6.2.2 Results and analyses

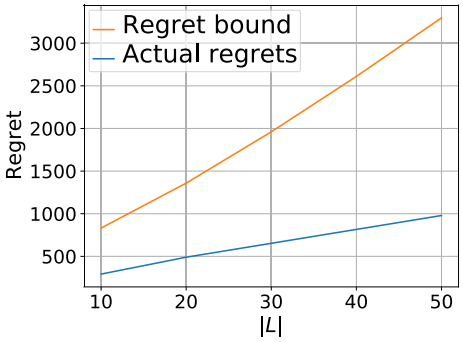
Figure 5.6 illustrates the rationale of mechanism $\mathscr{M}_1$ with the number of defense level set to $|L| = 10$. Figure 5.6(a) depicts the defender's cumulated loss incurred by using mechanism $\mathscr{M}_1$ and the optimal cumulated loss in hind-sight. From the figure, we can see that the regret gap between mechanism $\mathscr{M}_1$ and the optimal strategy increases as more rounds are added to the game. However, the rate of increase drops, as shown in



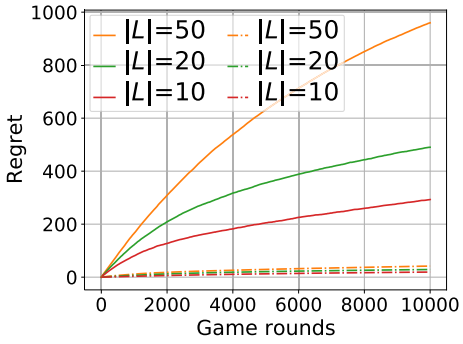
(a) The regret with $|L| = 20$



(b) The regret with $|L| = 50$



(c) The change of regrets with $|L|$



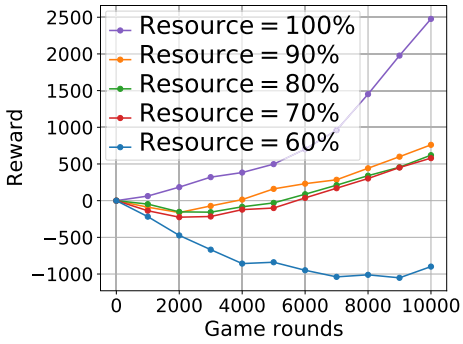
(d) The regret and averaged regret

Figure 5.7: The performance of mechanism $\mathscr{M}_1$ with varying defense levels L.

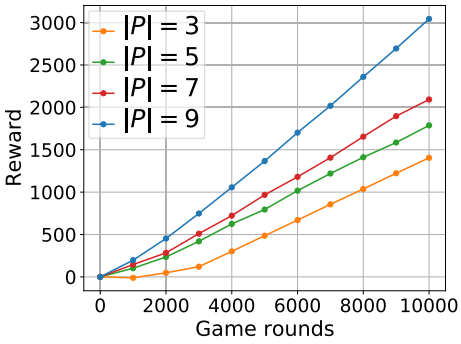
Figure 5.6(b). Figure 5.6(b) describes the change of the defender's regret as well as the theoretical regret bound. The theoretical regret bound is a linear function of the game round T, while the actual regret incurred by mechanism $\mathscr{M}_1$ is strictly lower within this bound. In addition, the rate of increase of the actual regret level is slower than the slope of the linear function.

To thoroughly explore this mechanism, we traced the variation tendency of each defense level's weight during the course of mechanism $\mathscr{M}_1$. As shown in Figure 5.6(c), at first, the weights of the 10 defense levels are all equal and normalized to $[0,1]$. As the game goes on, the mechanism iteratively updates the weight of each defense level, calibrating their weights according to the loss led by choosing them. After $50,000$ rounds, the weight of the best fixed defense level approaches 1, while the weights of other defense levels decrease to 0.

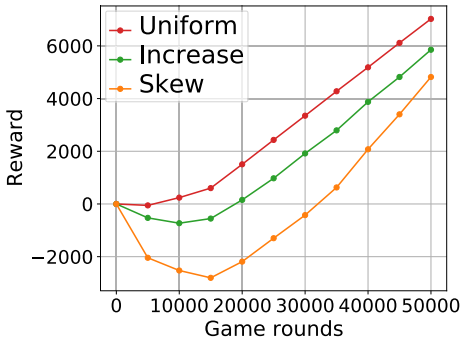
In Figure 5.6(d), we changed the resource of the defender to see whether the proposed mechanism still worked in such a circumstance. As APT attackers are always well-prepared and well-funded, we fixed the attacker's resource and gradually reduce the



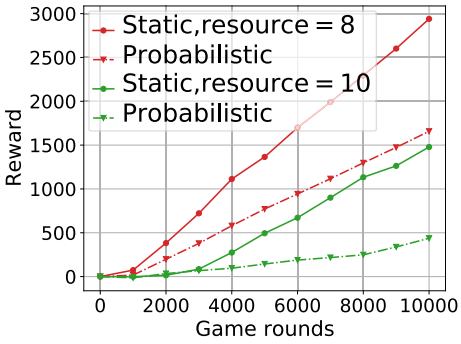
(a) The rewards with different defender's resources



(b) The rewards with different number of defense points



(c) The rewards with different importance values



(d) The rewards with static and probabilistic winning

Figure 5.8: The performance of mechanism $\mathscr{M}_2$ with varying defense resources, defense points and importance values.

resources available to the defender. Figure 5.6(d) shows the regret incurred with the defender's resource at 100%, 80%, 50%, and 20% proportional to the attacker. The figure tells that with less available defense resources, the regret of the defender increases, meaning that the defender suffers more due to a lack of resources. Note that with only 20% resources of the attacker, the defender reaches a barely satisfactory regret, which is twice as much as that with 100% resources. This is due to the oblivious assumption of the attacker in adversarial bandit setting. If the attacker keeps track of the defender's strategy and estimates the importance he has attached to each defense point, the defender will suffer heavily as a result of reduced resources (Figure 5.8(a)).

Figures 5.7 demonstrates how regret changes when the number of defense level increases. In APT attacks, there are multiple vulnerable points that need to be protected, which may leads to many defense levels being chosen. Therefore, we increased the level of defense $|L|$ to see if mechanism $\mathscr{M}_1$ still worked. Figure 5.7(a) and Figure 5.7(b) present

the regret incurred when $|L| = 20$ and $|L| = 50$ respectively, showing that increasing the defense level will not influence the relationship between theoretical regret bound and the actual regret. However, as the defense levels grow, the defender's actual regret increases. Figure 5.7(c) captures the increasing tendency of the regret bound and actual regret at $T = 10,000$ as $|L|$ increases from 10 to 50. The increase in actual regret can be explained by the fact that the mechanism explores more defense levels given more possible choices. In addition, the regret bound grows at a rate of $O(|L|\log|L|)$, which agrees with the theoretical analysis in Equation 5.38. Figure 5.7(d) shows the trends of actual regret incurred and the averaged regret for each defense level. From the figure, we can observe that although the actual regret doubled or tripled (solid lines), the average regret on each specific defense level increases quite slowly (dashed lines). This again tells us that the increase of actual regret is caused by an increase in defense levels, demonstrating the robustness of the proposed mechanism.

The experimental results relating to the second mechanism $\mathscr{M}_2$ are presented in Figure 5.8. Figure 5.8(a) illustrates the defender's accumulated reward as the number of game rounds grows. In the figure, the defender's reward increases as the game goes on, showing the effectiveness of mechanism $\mathscr{M}_2$. In this experiment, we assumed that the attacker employs a memory to record the past resource allocations of the defender, and that she has the ability to infer which defense points are more important to her opponent. As shown in Figure 5.8(a), reducing the defender's resources drastically influences his reward, which provides a stark contrast to Figure 5.6(d). When the defender's resource sit at 60% of the attacker's, it seems that the defender cannot competently defend against the attacks launched by his opponent. Figure 5.8(b) shows the change in the defender's reward when the number of defense points varies. With an increase in the number of defense points, the defender's reward also grows. This is because the enlarged "battlefields" provides more opportunities for the defender to win and gain. Note that this experiment was conducted based on the defender having enough defense resources. Without this assumption, the defender's reward may not increase.

Figure 5.8(c) considers the scenario where the defender may attach different importance values on different defense points. In this experiment, we changed the distribution of the importance factors used to compute the rewards, and examined the learning ability of the proposed mechanism. We set $|P| = 5$ defense points, and the total amount of importance factors was 15. In the uniform condition, the importance factor for each defense point is $I = \{3,3,3,3,3\}$, while that in the increased and skewed conditions were $I = \{1,2,3,4,5\}$ and $I = \{1,1,1,6,6\}$. From the figure, we can see that the mechanism

learns fast when the importance factors are uniformly distributed. In the other two conditions, the mechanism took some rounds to figure out the important defense points before starting to win. In addition, in the uniform condition, the defender received the highest reward. This is because, in the other conditions, the attacker and defender allocated more resources to important defense points. This made for stiff competition, thus reducing the defender's chance to win.

In Figure 5.8(d), we considered the fact that winning or losing a specific defense point may not be deterministic; it could possibly be probabilistic depending on the resources spent by both players. For example, if the defender allocated 3 resources to a defense point and the attacker spent 7, then the defender had a probability of 0.3 to win. Figure 5.8(d) shows that the defender receives more rewards in the deterministic setting. This is because, in the deterministic setting, the mechanism tries to improve the selected strategy when players tie with each other on a specific defense point, while, in the probabilistic setting, the same strategy may lead to a positive reward and thus deceives the training process. We also observed that the reward when players have 8 resources increases faster than that with 10 resources. The reason is that given fixed defense points, the network with 8 resources has less combinations of output.

5.7 Summary

This chapter develops an APT rivalry game to describe the interactions between attackers and defenders. The proposed game considers the information leaks incurred when players adjust their strategies. This chapter derives the necessary conditions for which a family of equilibria holds and showed that these equilibria actually imply the optimal timing for each player to make a strategy adjustment. The experimental results show that the equilibria of the game leads to higher utility and the proposed mechanisms can indeed learn from the players' past experiences. In the current game setting, the defender estimates their potential loss relying on public vulnerability databases provided by authoritative sources. Practically, a fresh attack may not have been previously recorded in these databases, making it difficult to determine the extent of the damage. Consequently, it is necessary to investigate novel relationships, such as cooperation, between the defender and attacker. Additionally, exploring new game formulations, such as bargaining games, might offer valuable perspectives on strategies to defend against APTs.

CHAPTER 6

DIFFERENTIAL PRIVACY AND MACHINE UNLEARNING: REMOVING THE IMPACT OF CLIENTS IN FEDERATED LEARNING

Over the past decades, the abundance of personal data leads to the rapid development of machine learning models and important advances in artificial intelligence (AI). However, alongside all the achievements, there are increasing privacy threats and security risks that may cause significant losses for data providers. Recent legislation requires that the private information about a user should be removed from a database as well as machine learning models upon certain deletion requests. While erasing data records from memory storage is straightforward, it is often challenging to remove the influence of particular data samples from a model that has already been trained. Machine unlearning is an emerging paradigm that aims to make machine learning models "forget" what they have learned about particular data. Nevertheless, the unlearning issue for federated learning has not been completely addressed due to its special working mode. First, existing solutions crucially rely on retraining based model calibration, which is likely unavailable and can bring new privacy risk for federated learning frameworks. Second, today's efficient unlearning strategies are mainly designed for convex problems, which are incapable of handling more complicated learning tasks like neural networks. To overcome these limitations, this chapter takes advantage of differential privacy and develops an efficient machine unlearning algorithm named FedRecovery. The FedRecovery erases the impact of a client by removing a weighted sum

of gradient residuals from the global model, and tailors the Gaussian noise to make the unlearned model and retrained model statistically indistinguishable. Furthermore, the algorithm neither requires retraining based fine-tuning nor needs the assumption of convexity. Theoretical analyses show the rigorous indistinguishability guarantee. Additionally, the experiment results on real-world datasets demonstrate that the FedRecovery is efficient and is able to produce a model that performs similarly to the retrained one.

6.1 Introduction

Machine unlearning is an emerging challenge that both the data science and policy communities are trying to grapple with. It stems from the increasingly urgent requirement of individuals for data protection, and is one of the centrepieces of machine learning research [49, 106]. Machine unlearning aims to post-process a trained model to remove the influence of specific training sample(s), such that the output model "looks as if it has never seen the unlearned data before". In fact, the development of machine unlearning is not only motivated by privacy and security issues, but also driven by legislation. For example, the European Union's General Data Protection Regulation (GDPR) [140] and the previous *Right to Be Forgotten* [39] entails the right for people to withdraw their consent to any processing operation on their data in certain contexts. Similar statements can be found in Canada's Consumer Privacy Protection Act (CPPA) [70] and California's Consumer Privacy Act (CCPA) [52], which obligate companies and organizations to delete individuals' information from a trained model upon request.

Federated learning [48] is a powerful working mode that provides privacy-preserving machine learning solutions. The core idea of federated learning is to train a machine learning model on separate datasets that are distributed across different devices or parties. During the training process, only the model's parameters or the gradients are shared, each client's data is kept invisible to other parties. Therefore, the model can be trained without disclosing the clients' raw data, and thus their data privacy can be preserved [84].

Machine unlearning plays an important role in addressing privacy and security issues in federated learning. Theoretically, unlearning an individual from a model is an ideal way to prevent information leakage from model inversion attacks [40] and membership inference attacks [124]. Moreover, unlearning techniques are also instrumental to eliminating the influence of data poisoning attacks [7] performed by malicious clients. For the same reason, machine unlearning can be used to update models if the previous

training data is outdated or of low quality [149]. Intuitively, a naive way to remove an individual's influence from a model is to retrain the model from scratch on the remaining data. However, retraining from scratch leads to prohibitively expensive computation costs [44]. Worse still, in federated learning, it is even impossible to perform retraining.

Achieving machine unlearning in federated learning is rather challenging due to its unique working mode. Firstly, the clients in federated learning are continuously changing. The server can hardly recall previous clients to perform an unlearning operation, let alone retraining from scratch. Secondly, the server has no access to the data samples to be unlearned. Therefore, existing centralized unlearning algorithms [50, 60] that are parameterized by unlearned data will not work in federated learning. Finally, the communication overhead between the server and clients is limited. As a result, the unlearning algorithms [46, 122] derived from the second-order optimization like the L-BFGS [151] are no longer efficient.

In federated learning, the former updates of clients have implicit and increasing influence on subsequent model updates during the training process, which is known as the incremental effect [83]. Therefore, unlearning algorithms have to disentangle the dependence of the unlearned client from the global model with the least damage to the contributions of other clients [139]. To address this issue, existing works [81, 143, 149] roughly remove the gradients or model weights concerning the unlearned client, and crucially rely on post-processed fine-tuning to repair the damage caused by unlearning operations. However, the fine-tuning process is in fact impractical and illegal in real-world scenarios. Firstly, fine-tuning also leads to considerable computation and communication costs. Secondly, as the server's computing power is limited, distributing the model for fine-tuning may bring new privacy and security risks to the unlearned client, which is strictly prohibited by the previously mentioned legislation. Therefore, this chapter explores a new question: *Is it possible to efficiently find a model that performs similarly to the retrained one?*

This chapter proposes a differentially private machine unlearning algorithm, FedRecovery, which takes advantage of clients' historical submissions to reproduce a model that is almost irrelevant to the client to be unlearned. The FedRecovery mathematically quantified the incremental effect by introducing the concept of gradient residual. It eliminates the influence of the unlearned client by removing a weighted sum of gradient residuals from the global model, where the weights are evaluated based on clients' actual contributions to decreasing the global loss. To provide a rigorous unlearning guarantee, this chapter adopts the notion of approximate statistical indistinguishability to limit the

difference between the unlearned model and the unavailable retrained model. Specifically, this chapter first derives the upper bound of the distance between the unlearned model and the retrained model, and then leverages the Gaussian mechanism to mask this gap in parameter space. As modern federated learning tasks generally start with pre-trained models given by the server [134], the discrepancy between models will be small, and the amount of noise can be precisely calibrated. FedRecovery neither requires the convexity assumption on loss functions nor needs any retraining based fine-tuning that may bring extra communication and computation costs. Therefore, it is suitable for today's federated learning framework and is component to handle complex unlearning tasks.

The main contributions of this chapter are summarized as follows.

- We mathematically quantified the incremental effect in federated learning, which provides a theoretical guideline to remove the influence of a participating client.
- We proposed an efficient unlearning algorithm FedRecovery, which reproduces a model that is indistinguishable from the retrained one by only exploring clients' historical submissions.
- We gave theoretical proofs and analyses of the proposed unlearning algorithm. We derived the upper bound of distance between the unlearned model and the retrained model, and achieved a rigorous unlearning guarantee by adding carefully calibrated noises.
- We conducted experimental simulations to show the indistinguishability between the models generated by FedRecovery. Further experiments demonstrate the effectiveness of the proposed unlearning algorithm.

The rest of this chapter first introduces basic background knowledge of machine unlearning and then details the design of the private learning and unlearning algorithms. Theoretical analyses and experimental results are also given.

6.2 Preliminaries

This section introduces the necessary background knowledge and basic concepts in differential privacy and indistinguishability. The formal definition of the unlearning problem to be addressed is also presented.

6.2.1 Differential privacy and (ϵ, β)-indistinguishability

Differential privacy [32, 90] is a rigorous mathematical privacy model that limits the influence each individual can make in statistical data analysis. The aim of differential privacy is to bound the differences in a query between two datasets that differ in a single entry [173]. Recently, differential privacy has also been proven to provide useful properties in the area of machine unlearning [44, 50].

Similarly, the (ϵ, β)-indistinguishability is a notion of approximate statistical indistinguishability developed from differential privacy literature, which requires that two random variables are indistinguishable for reasonable values of ϵ and β.

Definition 6.1. *((ϵ, β)-indistinguishability [103]) Let X and Y be random variables over domain $\mathscr{R}$. If for every possible subset of outcome $S \subseteq \mathscr{R}$, X and Y satisfy*

$$\begin{aligned} \Pr[X \in S] &\leq \exp(\epsilon) \cdot \Pr[Y \in S] + \beta, \\ \Pr[Y \in S] &\leq \exp(\epsilon) \cdot \Pr[X \in S] + \beta, \end{aligned} \tag{6.1}$$

then the variables X and Y are (ϵ, β)-indistinguishable.

One of the most widely used mechanisms that can achieve indistinguishability is the Gaussian mechanism, which adds Gaussian distributed noise to the statistical output.

Definition 6.2. *(Gaussian mechanism [13]) Given random variables $X \sim \mathcal{N}(\mu_1, \sigma^2 \mathbb{I}_d)$ and $Y \sim \mathcal{N}(\mu_2, \sigma^2 \mathbb{I}_d)$ that satisfy $\|\mu_1 - \mu_2\| \leq \Delta$, then for any $\beta > 0$, X and Y are (ϵ, β)-indistinguishable if*

$$\epsilon = \frac{\Delta^2}{2\sigma^2} + \frac{\Delta}{\sigma}\sqrt{2\log(1/\beta)}. \tag{6.2}$$

6.2.2 Problem definition

This chapter aims to find a model that performs similarly to the retrained model, given a model that has already been trained by n clients and the intermediate statistics cached during training. In other words, consider that there are $C = \{c_1, \ldots, c_n\}$ clients that collaboratively trained a global model $\mathscr{M}$ under the schedule of the central server S. After the model is trained, the i_u-th client submits an unlearning request to withdraw his consent to the use of his data. Therefore, the server S has to remove the influence of client c_{i_u}'s data from the trained model M. Ideally, the performance of the unlearned model should be comparable to that of the retrained one.

Without retraining and fine-tuning, it is generally not possible to recover an unlearned model from M that is completely independent of client c_{i_u}'s influence. Therefore, this

chapter turns to a compelling alternative: *is it possible to design an unlearning algorithm that produces a (distribution of) model(s) that is statistically indistinguishable from the (distribution of) model(s) that would have arisen from full retraining?* Mathematically, the unlearning guarantee is formulated as the follow definition.

Definition 6.3. *(Client-level (ϵ, β)-machine unlearning) An unlearning algorithm $\mathscr{M}_U$ satisfies (ϵ, β)-machine unlearning with respect to the learning algorithm $\mathscr{M}_L$, if for any possible set of output $W \subseteq \mathbb{R}^d$,*

$$\begin{aligned} \Pr[\mathscr{M}_L(C \backslash \{c_{i_u}\}, \epsilon) \in S] \leq e^{\epsilon} \cdot \Pr[\mathscr{M}_U(w, \Omega, \epsilon) \in S] + \beta, \\ \Pr[\mathscr{M}_U(w, \Omega, \epsilon) \in S] \leq e^{\epsilon} \cdot \Pr[\mathscr{M}_L(C \backslash \{c_{i_u}\} \in S] + \beta, \end{aligned} \tag{6.3}$$

where w represents the parameter of model M, and Ω is the set of cached statistics the server obtained from each client, such as gradients and intermediate model parameters.

This chapter follows the real-world assumption that the server S cannot access any data of any clients. In addition, the server's computing ability is limited, such that it cannot perform any training work.

For the properties of loss functions, it is assumed that the local loss function $f_i(w)$ of each client is L-smooth.

Assumption 6.1. *(L-Smoothness) A function $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth if it is differentiable and its gradient function $\nabla f : \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz continuous with constant L, i.e., $\forall w_1, w_2 \in \mathbb{R}^d$,*

$$\|\nabla f(w_1) - \nabla f(w_2)\| \leq L \|w_1 - w_2\|. \tag{6.4}$$

Without loss of generality, each client employs the gradient descent method to find the optimal parameter, which is a popular choice in optimizing complex tasks such as neural networks. If the stochastic gradient descent method is used, there is a further assumption about the stochasticity.

Assumption 6.2. *(The norm of stochastic gradients) The expected squared norm of stochastic gradients is uniformly bounded, namely for all $i = 1, \ldots, n$ and $t = 1, \ldots, t$, there exists a $G > 0$ such that*

$$\mathbb{E}_{\xi_t} \|\nabla f_i(w_t)\|^2 \leq G^2, \tag{6.5}$$

where ξ_t represents the sampling distribution over local data.

To improve legibility, it is assumed that the n-th client is the one who wishes to unlearn his data, but the unlearning algorithm works for arbitrary client.

6.3 The FedRecovery algorithm

subsectionDesign rationale and algorithm overview The FedRecovery algorithm has two components: the perturbed learning algorithm $\mathscr{M}_L$ and the unlearning algorithm $\mathscr{M}_U$, as depicted in Figure 6.1. The learning algorithm $\mathscr{M}_L$ employs gradient descent to obtain a trained model, while the unlearning algorithm $\mathscr{M}_U$ removes the influence of the client who submits an unlearning request. Both of them rely on the Gaussian perturbation to achieve (ϵ, β)-machine unlearning.

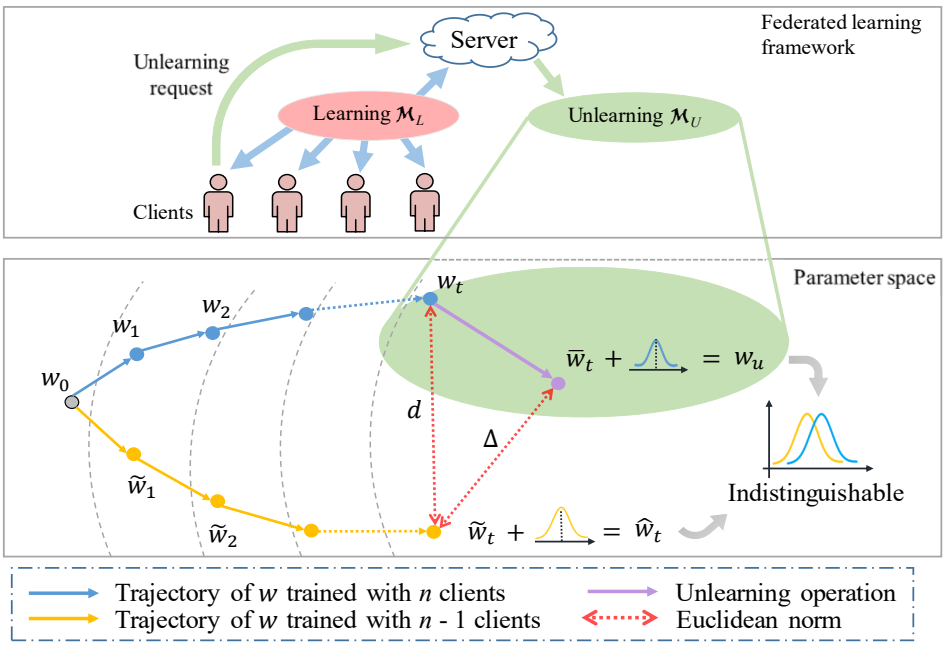


Figure 6.1: The framework of the FedRecovery algorithm and its rationale in the parameter space. The learning algorithm $\mathscr{M}_L$ trains the model using the gradient descent method, which corresponds to the blue and yellow trajectories. The unlearning algorithm $\mathscr{M}_U$ removes the influence of the client who wishes to be unlearned, as shown in green. The Gaussian noise is added to guarantee the unlearned model and retrained model are indistinguishable.

At the beginning of a federated training, the server S initializes the model structure and sets a start point w_0, which are then distributed to n participating clients $C = \{c_1, \ldots, c_n\}$. During the training process, each client computes the gradient $\nabla f_i(w)$ using their local data, and the server iteratively updates the global model's parameter w. The change of w forms a trajectory in parameter space W, as shown by the blue lines. After t iterations, the server S obtains a trained model M, but unfortunately, the n-th client

c_n then requires to withdraw the consent of using his data. Therefore, the server has to unlearn the influence of c_n from M by leveraging the resources at hand, namely the series of parameters $\{w_i\}_{i=0}^{t}$ and the cached gradients $\{\nabla f_i(w_j)\}$, $i \in [t]$, $j \in [n]$.

In fact, without client c_n's participation, the first $n-1$ clients would also form a descending trajectory $\{\tilde{w}_i\}_{i=1}^{t}$ in the parameter space after t training iterations, as shown by the yellow lines. The aim of FedRecovery is to find a $\bar{w} \in W$ that eliminates c_n' s contribution as much as possible by removing the gradient residuals generated from clients' historical submissions (the green line). To achieve (ϵ, β)-machine unlearning, the algorithm introduces perturbations to $\bar{w}_t$ and $\tilde{w}_t$ based on the upper bound of the distance Δ between them. Specifically, the Gaussian noise calibrated by Δ is added to $\bar{w}_t$ and $\tilde{w}_t$, and thus the unlearning guarantee can be promised by the property of (ϵ, β)-indistinguishability.

Theoretically, if the loss functions are convex, it is easy to bound the distance Δ between $\bar{w}_t$ and $\tilde{w}_t$ as the optimal point for convex functions is unique. Without the assumption of convexity, the FedRecovery algorithm derives an upper bound of $d \geq \Delta$ by exploring the smoothness of loss functions, which serves as an upper bound of Δ.

It is worth noting that the parameters $\{\tilde{w}_i\}_{i=1}^{t}$ are unknown throughout the course of the FedRecovery algorithm; only the distance between $\bar{w}_t$ and $\tilde{w}_t$ is estimated. In addition, to achieve (ϵ, β)-machine unlearning, the result of the learning algorithm is also randomized, which differs from traditional federated learning frameworks. Finally, there are extreme cases where $\{\tilde{w}_i\}_{i=1}^{t}$ stay close to w_0, i.e., the training would have failed with $n-1$ clients. The FedRecovery algorithm also works in such special scenarios, but the amount of introduced noise will be consequently large.

The detailed algorithms and related concepts are explained in the subsequent sections.

6.3.1 Gradient residual and the perturbed learning algorithm

This section first introduces the concept of gradient residual to theoretically demonstrate the incremental effect of parameters in federated learning, and then presents the perturbed federated learning algorithm that is helpful in achieving machine unlearning.

Without loss of generality, in the t-th round of model update, the server collects gradients from n participating clients and aggregates all the updates to obtain a new global model w_t.

$$w_t = w_{t-1} - \eta \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_{t-1}), \quad i \in [n]. \tag{6.6}$$

Suppose the n-th client submits an unlearning request to the server after training. Consider the case where the n-th client does not participate in the t-th aggregation, then the parameter update at time step t will become

$$\tilde{w}_t = \tilde{w}_{t-1} - \eta \cdot \frac{1}{n-1} \sum_{i=1}^{n} \nabla f_i(\tilde{w}_{t-1}), \quad i \in [n]. \tag{6.7}$$

To investigate the influence of the last client, we can subtract Equation 6.7 from Equation 6.6, and have

$$\begin{aligned} & w_t - \tilde{w}_t - (w_{t-1} - \tilde{w}_{t-1}) \\ = & -\eta \cdot [\frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_{t-1}) - \frac{1}{n-1} \sum_{i=1}^{n-1} \nabla f_i(\tilde{w}_{t-1})] \\ = & -\frac{\eta}{n-1} \sum_{i=1}^{n-1} [\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}_{t-1})] + \delta_{t-1}, \end{aligned} \tag{6.8}$$

where δ_{t-1} is the gradient residual that contains the information submitted by the last client in round $t-1$, which is derived from all the n clients' updates and can be computed exactly by the server.

$$\delta_{t-1} = \frac{\eta}{n} [\frac{1}{(n-1)} \sum_{i=1}^{n-1} \nabla f_i(w_{t-1}) - \nabla f_n(w_{t-1})]. \tag{6.9}$$

Note that although $\{\delta_i\}_{i=1}^{t}$ contains the information from the client who wants to unlearn his influence, it is not reasonable to subtract $\sum_{i=1}^{t} \delta_i$ directly from w^*. This is because the influence of each δ_t extremely differs along the course of the training process. To see this, we apply the triangular inequality in Equation 6.8.

$$\begin{aligned} & \|w_t - \tilde{w}_t\| - \|w_{t-1} - \tilde{w}_{t-1}\| \\ \leq & \frac{\eta}{n-1} \sum_{i=1}^{n-1} \|\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}_{t-1})\| + \|\delta_{t-1}\| \\ \leq & \eta L \|w_{t-1} - \tilde{w}_{t-1}\| + \|\delta_{t-1}\|, \end{aligned} \tag{6.10}$$

where the last inequality is derived by introducing the smoothness of the client's loss function.

Then, we sum Equation 6.10 iteratively from training round 1 to t, and with the fact $\|w_0 - \tilde{w}_0\| = 0$, we have

$$\begin{aligned} \|w_t - \tilde{w}_t\| &\leq \gamma^{t-1} \|\delta_0\| + \gamma^{t-2} \|\delta_1\| + \cdots + \|\delta_{t-1}\| \\ &= \sum_{i=1}^{t} \gamma^{t-i} \|\delta_{i-1}\|, \end{aligned} \tag{6.11}$$

where $\gamma = 1 + \eta L$ is jointly determined by the nature of the loss function (i.e., smoothness L) and the learning rate η.

Equation 6.11 demonstrates that the distance between w_t and $\tilde{w}_t$ is upper-bounded by the series $\{\delta_i\}_{i=0}^{t-1}$, where the influence of δ_i grows exponentially along the training process. Meanwhile, it also explains the incremental effect of parameters generated in federated learning, namely the preceding parameters have more influence than the later ones.

Theoretically, to limit the upper bound of $\|w_t - \tilde{w}_t\|$, it is necessary to set $\eta < \frac{1}{tL}$, such that $\lim_{t\to\infty}(1+\eta L)^t = 1$ and $\|w_t - \tilde{w}_t\| \leq \sum_{i=1}^{t} \|\delta_{i-1}\|$. However, it is not possible or practical to run federated learning for infinitely many rounds, and thus the series $\{\gamma^{t-i}\}$ may not converge.

Intuitively, the best way of eliminating the n-th client's influence is to remove the weighted sum of $\{\delta_i\}_{i=0}^{t-1}$ from w_t with weights γ^{t-i}, which will ideally produce a $\hat{w}_t$ that is closer to $\tilde{w}_t$. However, this is not feasible in practice. First and foremost, computing the Lipschitzness and smoothness of a neural network is generally an NP-hard problem [121]. Therefore, we cannot precisely obtain the constant L in polynomial time (assuming that P≠NP). Second, the upper bound in Equation 6.11 ignores the implicit interactions among clients. In fact, the trajectory of the global model's parameter would likely change without the participation of the n-th client. These $\{\delta_i\}_{i=0}^{t-1}$ are derived from n clients' current gradients, but the gradients of the first $n-1$ clients may be implicitly biased.

Further, the incremental effect means that even if the influence of the n-th client is removed, there is no guarantee that we can obtain a model that is *exactly* the same as the one without the n-th client's participation. This is because the stochasticity incurred during training will definitely change the possible trajectory of parameters generated in gradient descent. Therefore, the best thing we can do is to make w_u and $\tilde{w}_t$ indistinguishable, such that an observer cannot identify if the model is trained by $n-1$ or n clients. According to the rationale of the Gaussian mechanism, to achieve indistinguishability, the output model of the federated learning algorithm should be perturbed by Gaussian noise. The perturbed learning algorithm is presented in Algorithm 11.

Following the above intuition, we employed the gradient flow [85] method to quantify the influence of the n-th client on the global model's performance. In addition, we derived a tighter upper bound of $\|w_t - \tilde{w}_t\|$ using the theory of gradient descent in non-convex optimization [74], and adopted differentially private mechanism to make w_u and $\tilde{w}_t$ indistinguishable.

Algorithm 11 The perturbed federated learning algorithm $\mathscr{M}_L$

Input: the initial parameter $w_0 \in \mathbb{R}^{\mathrm{d}}$ trained by federated learning, the privacy budget ϵ for the Gaussian mechanism, the confidence parameter β.

Output: the perturbed global model $\hat{w}$.

1: **train** the model from w_0 by t rounds federated learning and obtains parameter w_t.

2: **set**

$$\sigma = \frac{1}{\sqrt{2}} \cdot \frac{d}{\sqrt{\log(1/\beta)+\epsilon} - \sqrt{\log(1/\beta)}},$$

where d is the upper bound of the distance in Equations 6.30 to 6.32 based on the gradient descent method used.

3: **sample** a noise $z \in \mathbb{R}^{\mathrm{d}}$ from a Gaussian distribution, $z \sim \mathscr{N}(0, \sigma^2 \mathbb{I}_{\mathrm{d}})$.

4: **return** $\hat{w}_t = w_t + z$.

6.3.2 The machine unlearning algorithm

The unlearning algorithm $\mathscr{M}_U$ is established based on gradient flow and differential privacy. The gradient flow studies the dynamics of gradients in continuous time, which is essentially the limit of the gradient descent method when the learning rate η approaches to 0.

When $\eta \to 0$, the gradient descent dynamic Equation 6.6 can be reformulated as

$$\lim_{\eta \to 0} \frac{w_t - w_{t-1}}{\eta} = -\frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_{t-1}) \tag{6.12}$$

Denote $F(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$ the objective function of the server, we obtain the following differential equation,

$$\frac{\mathrm{d}w_t}{\mathrm{d}t} = -\nabla F(w_t). \tag{6.13}$$

Therefore, the variation tendency of the objective function $F(w)$ with respect to t can be derived using the chain rule.

$$\frac{\mathrm{d}F(w_t)}{\mathrm{d}t} = \frac{\mathrm{d}F(w_t)}{\mathrm{d}w_t} \cdot \frac{\mathrm{d}w_t}{\mathrm{d}t} = -\|\nabla F(w_t)\|^2. \tag{6.14}$$

The above equation characterizes the sensitivity of the objective function along the trajectory of the global model's parameter. As $-\|\nabla F(w_t)\|^2 \leq 0$, the equation also guarantees that, in general, if $F(w)$ is bounded from negative infinity, the federated learning process will converge.

As the server has easy access to the gradients of clients and has to compute $\nabla F(w_t)$ according to the protocol of federated learning, $\|\nabla F(w_t)\|^2$ is a natural way to weight the contribution of clients' gradients in each iteration. In comparison, using $\|\nabla F(w_t)\|^2$

rather than γ^t brings another advantage in that $\nabla F(w_t)$ incorporates the influence of the first $n-1$ clients' gradients with regard to $\nabla f_n(w_t)$. For example, the norm $\|\nabla f_n(w_t)\|$ itself may be large due to the landscape of the local loss function f_n, but the norm of the aggregated gradient $\|\nabla F(w_t)\|$ may stay small because the influence of the other $n-1$ clients' gradients is taken into consideration during the aggregation process. According to Equation 6.14, the global model's performance change is reflected by $\|\nabla F(w_t)\|$. Therefore, it is reasonable to use $\|\nabla F(w_t)\|^2$ as the weights to scale the influence of the n-th client from round 1 to t.

We use p_i to denote the weight of δ_i in the i-th iteration of federated learning.

$$p_i = \frac{\|\nabla F(w_i)\|^2}{\sum_{j=1}^{t-1} \|\nabla F(w_j)\|^2}, \quad i \in [t]. \tag{6.15}$$

When the n-th client submits an unlearning request, the unlearning algorithm $\mathscr{M}_U$ removes the n-th client's influence by computing

$$\bar{w}_t = w_t - \sum_{i=1}^{t-1} p_i \cdot \delta_i, \quad i \in [t]. \tag{6.16}$$

To make w_u and $\tilde{w}_t$ indistinguishable, it is necessary to introduce randomizations to mask the gap between $\bar{w}_t$ and $\tilde{w}_t$. Differentially private mechanisms convert the distances in parameter space to the distance between distributions, which is a natural choice. The algorithm $\mathscr{M}_U$ employs the Gaussian mechanism to achieve indistinguishability, where the noise is carefully calibrated by the Euclidean norm $\|\bar{w}_t - \tilde{w}_t\|$. Therefore, for $\bar{w} \in \mathbb{R}^{\mathrm{d}}$, we have the unlearned model parameter

$$w_u = \bar{w}_t + z, \quad z \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_{\mathrm{d}}). \tag{6.17}$$

The unlearning guarantee of the FedRecovery algorithm is given by the following theorem.

Theorem 6.1. *The unlearning algorithm $\mathscr{M}_U$ satisfies (ϵ, β)-machine unlearning with respect to the learning algorithm $\mathscr{M}_L$. For any possible set of output $W \subseteq \mathbb{R}^d$,*

$$\begin{aligned} &\Pr[w_u \in W] \leq \exp(\epsilon) \cdot \Pr[\hat{w}_t \in W] + \beta, \\ &\Pr[\hat{w}_t \in W] \leq \exp(\epsilon) \cdot \Pr[w_u \in W] + \beta. \end{aligned} \tag{6.18}$$

Our key technical insight that leads to successful machine unlearning by introducing indistinguishability is based on the following theorems.

Theorem 6.2. *The upper bound of the distance between $\bar{w}$ and $\tilde{w}_t$ is smaller than the upper bound of the distance between w_t and $\tilde{w}_t$. Namely,*

$$\sup \|\bar{w} - \tilde{w}_t\| \leq \sup \|w_t - \tilde{w}_t\|. \tag{6.19}$$

Theorem 6.3. *The expected distance between w_t and $\tilde{w}_t$ is upper-bounded by the following inequality for all iteration t,*

$$\mathbb{E}_\xi[\|w_t - \tilde{w}_t\|] \leq \sqrt{\sum_{t=0}^{t-1} \eta_t [F(w_0) - F(w^*) + \frac{LG^2}{2} \sum_{t=0}^{t-1} \eta_t^2]} + D_t, \tag{6.20}$$

where w^* is the optimal solution of the objective function $F(w)$, and D_t is a constant relating to the global model's training trajectory. The expectation is taken with respect to the stochasticity of gradient descent $\xi = (\xi_1, \ldots, \xi_t)$.

The above theorems enable us to quantify the upper bound of distance between the parameter $\bar{w}_t$ and the parameter retrained without the n-th client $\tilde{w}_t$, which provides a chance to achieve (ϵ, β)-indistinguishability by introducing randomizations. The basic idea is as follows, for both the retrained parameter $\tilde{w}_t$ and the parameter $\bar{w}_t$ that removes the n-th client's effect, we use the Gaussian mechanism to form two distributions, $\hat{w}_t \sim \mathcal{N}(\tilde{w}_t, \sigma^2 \mathbb{I}_\mathrm{d})$ and $w_u \sim \mathcal{N}(\bar{w}_t, \sigma^2 \mathbb{I}_\mathrm{d})$, and differential privacy guarantees the statistical indistinguishability between the two random variables.

The complete unlearning algorithm is presented in Algorithm 12. The algorithm takes as input the unlearning request of client i_u, the existing trained model w_t and the gradients incurred during training. The parameter ϵ controls the indistinguishability between the unlearned model w_u and the actual model $\tilde{w}_t$ trained without the participation of client i_u. The algorithm starts by computing the aggregated gradients and the gradient residuals relating to client i_u (Line 1-7). For each time step t, it assigns a probability mass that will be used to weight the gradient residuals $\{\delta_i\}_{i=0}^{t-1}$ (Line 8). The weights p_i are derived based on the norm of the aggregated gradients. Then the influence of the i_u-th client is eliminated by removing the weighted sum of δ_is from the trained model w_t (Line 9). Finally, the algorithm samples a noise vector from a Gaussian distribution and adds it to the previously deduced parameters $\bar{w}_t$ to achieve indistinguishability (Line 10-11). We defer the detailed proofs and related analyses of the algorithm to Section 6.4.

Algorithm 12 Unlearning algorithm $\mathscr{M}_U$ for federated learning

Input: the unlearning request from the i_u-th client, the existing global model: $w_t \in \mathbb{R}^{\mathrm{d}}$, the gradients submitted by each client: $\nabla f_i(w_j)$ for $i \in [n]$ and $j \in [t]$, the privacy budget for the Gaussian mechanism: ϵ.

Output: the unlearned global model w_u.

1: **set**

$$\sigma = \frac{1}{\sqrt{2}} \cdot \frac{d}{\sqrt{\log(1/\beta)+\epsilon} - \sqrt{\log(1/\beta)}},$$

where d is the upper bound of the distance in Equation 6.30 to 6.32 based on the gradient descent method used.

2: **for** $i \in [t]$ **do**

3: **aggregate** the gradients for the global model,

$$\nabla F(w_i) = \frac{1}{n} \sum_{j=1}^{n} \nabla f_j(w_i), \quad j \in [n].$$

4: **end for**

5: **for** $i \in [t]$ **do**

6: **compute** the gradient residuals $\{\delta_i\}_{i=1}^{t}$ of the i_uth client by Equation 6.9.

$$\delta_i = \frac{\eta}{n}[\frac{1}{(n-1)} \sum_{j \neq i_u} \nabla f_j(w_i) - \nabla f_{i_u}(w_i)].$$

7: **end for**

8: **compute** the weights p_i for each gradient residual δ_i.

$$p_i = \frac{\|\nabla F(w_i)\|^2}{\sum_{j=1}^{t-1} \|\nabla F(w_j)\|^2}, \quad i \in [t].$$

9: **subtract** a weighted sum of δ_i from w.

$$\bar{w}_t = w_t - \sum_{i=1}^{t-1} p_i \cdot \delta_i, \quad i \in [t].$$

10: **sample** a noise $z \in \mathbb{R}^{\mathrm{d}}$ from a Gaussian distribution, $z \sim \mathscr{N}(0, \sigma^2 \mathbb{I}_{\mathrm{d}})$.

11: **return** $w_u = \bar{w}_t + z$.

6.3.3 Discussion and analysis

In this section, we present discussions and related analyses of the FedRecovery algorithm, including running time, memory usage, early stopping strategy, etc.

• **Running time.** The FedRecovery algorithm only needs simple calculations to achieve unlearning, e.g., subtraction and sampling. Let t denote the training round, the time consumption of subtracting gradient residuals is $O(t)$. The computation of model distance and the sampling of Gaussian noise are in $O(1)$. In comparison to previous algorithms [81, 143, 149], FedRecovery does not require any retraining or fine-tuning based calibration to improve model's performance, and thus saves $O(t_{extra})$ time for extra calibration. Moreover, the unlearning algorithm only runs on the server's side, without additional collaboration with the previous or new clients. Therefore, the communication overhead of FedRecovery is 0, which is also significantly reduced.

• **Memory usage.** The FedRecovery algorithm does not require memory-intensive checkpoint storage for models like the SISA [9] and RecEraser [18]. However, it does need the server to keep clients' historical submissions, which costs $O(tn)$ in memory space. Given a fixed training round, the memory storage of FedRecovery grows linearly with the number of clients in federated learning. In comparison, the storage of SISA and RecEraser increases with the number of submodels and checkpoints. To further reduce memory usage, the server can adopt an early stop strategy in the unlearning process. The Gaussian mechanism will guarantee the indistinguishability between the early-stopped model and the retrained model.

• **Early stopping.** For neural networks, the NTK theory [76] indicates that in an overparameterization regime, the gradient-based methods converge exponentially fast to zero training error, with the model's parameters hardly varying (aka. "lazy training" phenomenon [21]). Therefore, for the purpose of saving memory, the server can store clients' submissions for only the first $t_0 < t$ rounds according to the global model's performance, and then run the FedRecovery to achieve machine unlearning.

• **Subsequent training.** The retraining-based unlearning algorithms [130, 143] suggest performing subsequent training on the remaining dataset to "heal" the damage caused by the unlearning operation. In FedRecovery, we followed another line of work that directly uses the Gaussian noise to mask the gap between parameters. We note that it is not necessary to perform further subsequent training on the output of FedRecovery, because the output of the algorithm is already guaranteed to be indistinguishable from retrained models.

• **Gradient disclosure.** Existing research has demonstrated that submitting raw gradients to the server can still reveal clients' information [56, 171]. Current federated learning frameworks typically employ differentially private techniques or encryption-based methods to protect clients' submissions. However, in the context of machine unlearning, the server must verify the legitimacy of each unlearning request. For local differential privacy-based federated learning frameworks, the server can continue to regard each client's noisy submission as their contribution and use the proposed mechanism to eliminate their impact. In the scenario where the model parameter is updated under homomorphic encryption. It is impossible to perform an unlearning operation because the server cannot validate and distinguish the contributions of each client. In addition, there is no need for clients to submit an unlearning request in this context, as the encryption procedure has already severed the relationship between clients and their submissions.

• **Migration to centralized unlearning.** It is natural to consider the migration of FedRecovery to the centralized machine unlearning scenario. In fact, differentially private randomization has been widely adopted in recent research to achieve a rigorous unlearning guarantee [50, 103, 122]. The centralized unlearning scenario offers the advantage of granting the model owner access to the training data, allowing for the caching of more useful information such as Hessian matrices. By contrast, with federated learning, the only data accessible is the gradients submitted by clients. Hence, the incorporation of additional information in FedRecovery could enhance its performance in a centralized unlearning scenario.

6.4 Theoretical analysis

In this section, we present detailed proofs of previous theorems. We first show that the proposed unlearning algorithm satisfies (ϵ, β)-indistinguishability (Theorem 6.1), and then give an upper bound on the distance between w_t and $\tilde{w}_t$ (Theorems 6.2 and 6.3).

Theorem 6.1 guarantees the (ϵ, β)-indistinguishability of the proposed unlearning algorithm, which is essentially an implementation of the Gaussian mechanism.

Proof of Theorem 6.1. Denote $C = \{c_1, \ldots, c_n\}$ the set that contains n clients in a federated learning, and $C' = C \backslash \{c_{i_u}\}$ the set that contains $n-1$ clients excluding the i_u-th one. In addition, let $\hat{w}_t = \mathscr{M}_L(C', \epsilon)$ be the model obtained by retraining from scratch and $w_u = \mathscr{M}_U(w_t, \nabla F, \epsilon)$ be the model produced by the unlearning algorithm. Note that w_t is the parameter trained with clients in C without adding noise.

According to Algorithm 11 and Algorithm 12, we know that $\hat{w}_t$ and w_u are computed as $\hat{w}_t = \tilde{w}_t + z$ and $w_u = \bar{w}_t + z$ respectively, where the noise $z \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_{\mathrm{d}})$. Therefore, the random variables $\hat{w}_t$ and w_u follow distributions $\hat{w}_t \sim \mathcal{N}(\tilde{w}_t, \sigma^2 \mathbb{I}_{\mathrm{d}})$ and $w_u \sim \mathcal{N}(\bar{w}_t, \sigma^2 \mathbb{I}_{\mathrm{d}})$.

In addition, according to Theorem 6.2, we know that the upper bound of $\|\bar{w}_t - \tilde{w}_t\|$ is bounded by the upper bound of $\|w_t - \tilde{w}_t\|$, where the latter is given in Theorem 6.3.

Finally, we invoke the Gaussian mechanism (Definition 6.2) to show that variables $\hat{w}_t$ and w_u are indistinguishable, and thus the unlearning algorithm $\mathscr{M}_U$ satisfies (ϵ, β)-unlearning guarantee with

$$\sigma = \frac{1}{\sqrt{2}} \cdot \frac{d}{\sqrt{\log(1/\beta) + \epsilon} - \sqrt{\log(1/\beta)}}, \tag{6.21}$$

where the distance d is taken following Equations 6.30 to 6.32 based on the specific gradient descent method used.

That completes the proof. □

Theorem 6.2 provides an insight about the upper bound of distance between w_u and $\tilde{w}_t$, which enables us to quantify the amount of noise used to achieve (ϵ, β)-indistinguishability.

Proof of Theorem 6.2. Note that Equation 6.8 illustrates the change of the global model's parameters between two successive training rounds. We hence sum Equation 6.8 over rounds 1 to t. Based on the fact $w_0 \equiv \tilde{w}_0$, we can obtain the difference between parameters $\tilde{w}_t$ and w_t.

$$\tilde{w}_t - w_t = \frac{\eta}{n-1} \sum_{j=1}^{t} \sum_{i=1}^{n-1} [\nabla f_i(w_{j-1}) - \nabla f_i(\tilde{w}_{j-1})] - \sum_{j=1}^{t} \delta_{j-1}. \tag{6.22}$$

According to Equations 6.7 and Equation 6.16, the difference between parameter $\tilde{w}_t$ and the parameter without the n-th client's influence $\bar{w}_t$ is given by

$$\begin{aligned} \tilde{w}_t - \bar{w}_t = \frac{\eta}{n-1} \sum_{j=1}^{t} \sum_{i=1}^{n-1} [\nabla f_i(w_{j-1}) - \nabla f_i(\tilde{w}_{j-1})] \\ - [\sum_{j=1}^{t} \delta_{t-1} - \sum_{j=1}^{t} p_j \delta_{t-1}]. \end{aligned} \tag{6.23}$$

Note that the first item in Equation 6.22 and Equation 6.23 are the same, denoted as A. By applying the triangle inequality and the fact $p_j \leq 1$, we have

$$\begin{aligned} \sup \|\bar{w}_t - \tilde{w}_t\| &\leq \|A\| + \sum_{j=1}^{t} (1 - p_j) \|\delta_{j-1}\| \\ &\leq \|A\| + \sum_{j=1}^{t} \|\delta_{j-1}\| \\ &= \sup \|w_t - \tilde{w}_t\| \end{aligned} \tag{6.24}$$

Therefore, the Euclidean norms between parameters satisfy

$$\sup \|\bar{w}_t - \tilde{w}_t\| \leq \sup \|w_t - \tilde{w}_t\|. \tag{6.25}$$

That completes the proof. □

Theorem 6.3 states that for a properly chosen learning rate, the expected distance between parameters w_t and $\tilde{w}_t$ grows at a rate of $O(\sqrt{t})$. Therefore, we can bound the distance between $\bar{w}$ wand $\tilde{w}_t$ by bounding the distance between w_t and $\tilde{w}_t$.

Before proofing Theorem 6.3, we first show the smoothness of the global objective function $F(w)$, which is guaranteed by the smoothness of each local objective function $f_i(w)$.

Lemma 6.1. *(Smoothness of function F) If each function $f_i : \mathbb{R}^d \to \mathbb{R}$ is L-smooth, the function $F(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$ is also L-smooth.*

Proof. The result follows by the definition of L-smoothness and the triangle inequality.

$$\begin{aligned} \|\nabla F(w_1) - \nabla F(w_2)\| &= \frac{1}{n}\|\sum_{i=1}^{n} \nabla f_i(w_1) - \sum_{i=1}^{n} \nabla f_i(w_2)\| \\ &\leq \frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(w_1) - \nabla f_i(w_2)\| \\ &\leq L\|w_1 - w_2\|. \end{aligned} \tag{6.26}$$

Now we present the proof of Theorem 6.3 based on lemma 6.1.

Proof of Theorem 6.3. Denote $G(w_t, \xi_t)$ the gradient of $F(w_t)$ with sampling distribution ξ_t. By the smoothness of function F and the stochastic gradient descent update rule $w_{t+1} = w_t - \eta_t G(w_t, \xi_t)$, we have

$$\begin{aligned} &\mathbb{E}_{\xi_t}[F(w_{t+1})] \\ &\leq \mathbb{E}_{\xi_t}[F(w_t) + \nabla F(w_t)^T(w_{t+1} - w_t) + \frac{L}{2}\|w_{t+1} - w_t\|_2] \\ &= \mathbb{E}_{\xi_t}[F(w_t) - \eta_t \nabla F(w_t)^T G(w_t, \xi_t) + \frac{L}{2}\eta_t^2 \|G(w_t, \xi_t)\|^2] \\ &\leq \mathbb{E}_{\xi_t}[F(w_t)] - \eta_t \|\nabla F(w_t)\|^2 + \frac{1}{2}\eta_t^2 L G^2, \end{aligned} \tag{6.27}$$

where the last inequality follows from the assumption that $G(w_t, \xi_t)$ is an unbiased estimation of $\nabla F(w_t)$, and $\mathbb{E}_{\xi_t}\|\nabla f_i(w_t)\|^2 \leq G^2$.

Rearrange these terms, we have

$$\eta_t \|\nabla F(w_t)\|^2 \leq \mathbb{E}_{\xi_t}[F(w_t)] - \mathbb{E}_{\xi_t}[F(w_{t+1})] + \frac{L}{2}\eta_t^2 G^2. \tag{6.28}$$

Summing over t iterations and taking the overall expectation over $\xi = (\xi_1, \ldots, \xi_t)$, we have the following inequalities.

$$\begin{aligned}\sum_{t=0}^{t-1} \eta_t \|\nabla F(w_t)\|^2 &\leq \mathbb{E}_\xi[F(w_0)] - \mathbb{E}_\xi[F(w_{t-1})] + \frac{L}{2}\sum_{t=0}^{t-1}\eta_t^2 G^2 \\ &\leq F(w_0) - F(w^*) + \frac{1}{2}\sum_{t=0}^{t-1}\eta_t^2 L G^2.\end{aligned} \tag{6.29}$$

Finally, by applying the triangle inequality and the linearity of expectation, we have

$$\begin{aligned}&\mathbb{E}_\xi[\|w_t - \tilde{w}_t\|] \\ &= \mathbb{E}_\xi[\|w_t - w_0\|] + \mathbb{E}_\xi[\|\tilde{w}_t - w_0\|] \\ &= \|\sum_{t=0}^{t-1}\eta_t \mathbb{E}_{\xi_t}[G(w_t, \xi_t)]\| + D_t \\ &\leq \sum_{t=0}^{t-1}\eta_t \|\nabla F(w_t)\| + D_t \\ &\leq \sqrt{(\sum_{t=0}^{t-1}\eta_t)(\sum_{t=0}^{t-1}\eta_t \|\nabla F(w_t)\|^2)} + D_t \\ &\leq \sqrt{\sum_{t=0}^{t-1}\eta_t [F(w_0) - F(w^*) + \frac{LG^2}{2}\sum_{t=0}^{t-1}\eta_t^2]} + D_t,\end{aligned} \tag{6.30}$$

where the penultimate inequality follows from the Cauchy–Schwarz inequality. □

Based on Theorem 6.3, we immediately have the following corollary.

Corollary 6.1. *If each client adopts batch gradient descent in their local training, the distance between w_t and $\tilde{w}_t$ satisfies the following inequality for all iteration $t \in [T]$:*

$$\|w_t - \tilde{w}_t\| \leq \sqrt{2\sum_{t=0}^{t-1}\eta_t [F(w_0) - F(w^*)]} + D_t. \tag{6.31}$$

Further for a constant learning rate $\eta_k = \eta \leq 1/L$, we have

$$\|w_t - \tilde{w}_t\| \leq \sqrt{2t\eta[F(w_0) - F(w^*)]} + D_t. \tag{6.32}$$

Proof. (Sketch) In batch gradient descent, we do not consider the bias of gradient estimation, so that the second term in Equation 6.29 disappears and the result follows as Equation 6.31. If the learning rate is a constant η, $\sum_{i=0}^{t-1}\eta_t = t\eta$, which implies the result in Equation 6.32. □

6.5 Experimental results

In this section, we evaluate the performance of the proposed FedRecovery algorithm. We first examine the statistical indistinguishability it provides, investigating the difference between the unlearned model and the retrained model. Then we look into the performance of the unlearning algorithm to see whether the influence of the targeted client is removed. The experimental results and corresponding analyses are presented in each subsection.

6.5.1 Experimental setup

6.5.1.1 Datasets and models

We used four real-world datasets in the experiments, including MINST [75], CIFAR10 [73], SVHN [104], and USPS [58].

- **MNIST.** The dataset contains $60,000$ training images and $10,000$ test images for hand written digit recolonization. Each image is grayscale and normalized in 28×28 pixels.

- **CIFAR10.** The dataset consists of $60,000$ 32×32 color images in 10 classes, $50,000$ of them are training images, and $10,000$ are test images.

- **SVHN.** It contains $600,000$ 32×32 color images from pictures of house number plates. The images are centered and the background distractors are kept in the image.

- **USPS.** The dataset contains $9,300$ 16×16 gradyscale images scanned from envelopes by the U.S. Postal Service. The images are centered, normalized, and show a broad range of font styles.

In the experiments, we adopted convolutional neural network (CNN) models to perform image classification tasks on these datasets, which are generally non-convex. To limit the injected noise and guarantee a good performance, we started federated learning from pre-trained models, such that the distance between the unlearned model and retrained model will not change drastically. The pre-training strategy is commonly used in computer vision and natural language processing, and it does not violate the principle of federated learning [134].

6.5.1.2 Evaluation and comparison metrics

Intuitively, an effective unlearning algorithm should force the model to "forget" what it has learned about the unlearned data, but retain its "knowledge" of the remaining ones. In our experiment, we adopted the widely accepted criteria to evaluate the proposed algorithm.

- **Accuracy.** We investigated the overall accuracy of the unlearned model and the retrained model. Meanwhile, we also examined the accuracy of the unlearned model on unlearned class and remained classes.
- **Running time.** We looked into the running time needed to perform the proposed unlearning algorithm, and made comparisons with benchmark methods like retraining from scratch.
- **Attack success rate.** We performed membership inference attack (MIA) on the unlearned model to see if the influence of the targeted client was really removed by the proposed unlearning algorithm.

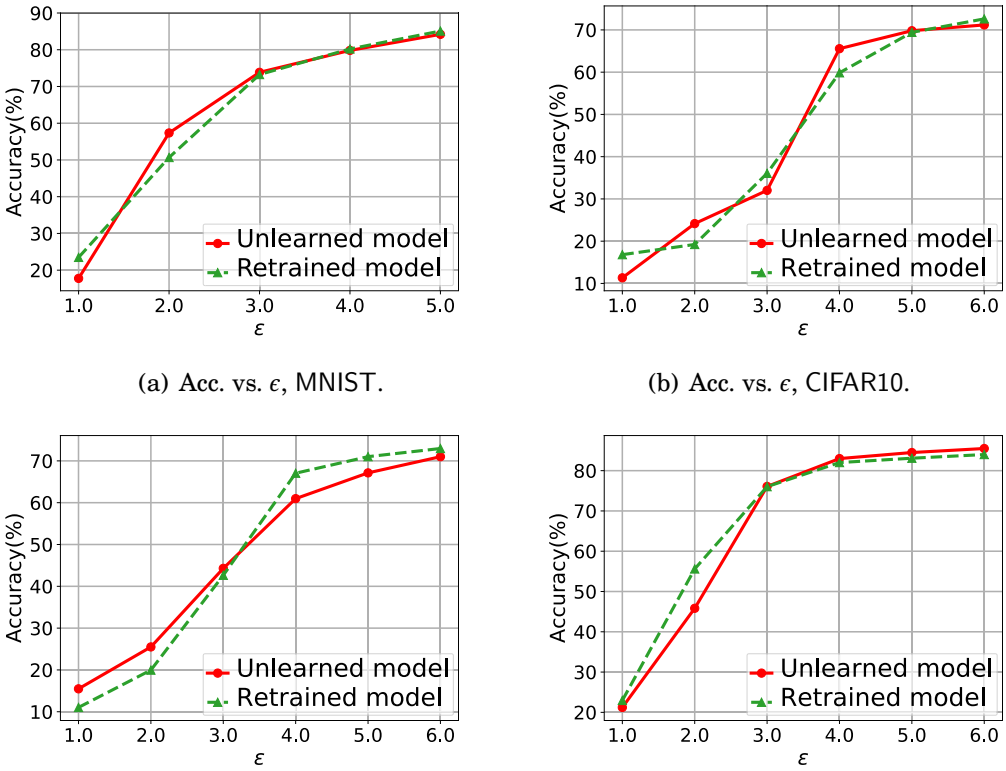
Verifying the success of machine unlearning is not an easy task, because the verification method is usually tailored based on certain unlearning strategies. In our experiment, we made comparisons with the state-of-the-art unlearning algorithms [81, 83, 143] in federated learning.

The experiments were conducted on a server with Red Hat Enterprise Linux 7.9 OS, Intel(R) Xeon(R) E-2288G 3.70GHz CPU, and NVIDIA Quadro RTX6000 GPU with 24G RAM.

6.5.2 Results and analyses

6.5.2.1 The influence of the privacy budget

The privacy budget controls the indistinguishability between the unlearned model and the retrained model. Therefore, our first experiment is designed to investigate the influence of the privacy budget on model's performance. In this experiment, we used $n = 10$ participating clients, each of whom holds a local database that is randomly sampled from the original one. The privacy budget was set in the range $1.0 \sim 6.0$, and the performance was assessed in terms of the global model's accuracy. The results are shown in Figure 6.2.



(a) Acc. vs. ϵ, MNIST.

(b) Acc. vs. ϵ, CIFAR10.

(c) Acc. vs. ϵ, SVHN.

(d) Acc. vs. ϵ, USPS.

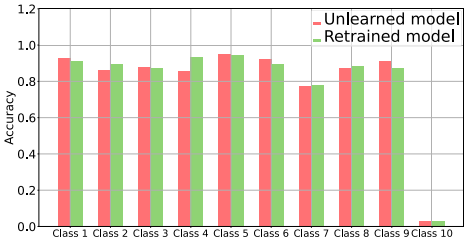
Figure 6.2: The accuracy of unlearned model and retrained model with different privacy budgets.

Figure 6.4(a) shows the change of model accuracy with respect to different privacy budgets on the MNIST dataset. In this figure, the accuracy of both models increases from around 20% to 90% with the privacy budget grows from 1.0 to 5.0. With a larger privacy budget, the requirement of indistinguishability is weaker, so that less noise is injected in the model's parameters. As a result, the accuracy of models increases. The result patterns were similar for the other three datasets. Note that in some cases, the accuracy of the unlearned model is higher than that of the retrained model. This is caused by the randomness incurred when sampling the Gaussian noise. From the figures, we can see that the performance of the unlearned model is similar to that of the retrained model, which verifies the indistinguishability achieved by the proposed FedRecovery algorithm.

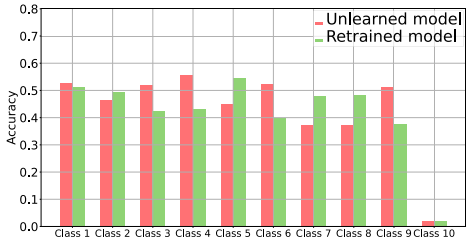
6.5.2.2 The effectiveness of unlearning

Our next experiment aims to examine the unlearning ability of the proposed FedRecovery algorithm. In this experiment, we took $n = 10$ clients, each of whom holds a local database

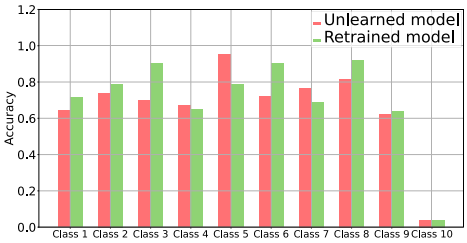
that corresponds to a specific class of data in the original database. The clients first collaboratively trained a model that was able to recognize data in 10 classes, and then we ran FedRecovery to remove the influence of the last client, who contributed to the last class of data. The results are shown in Figure 6.3.



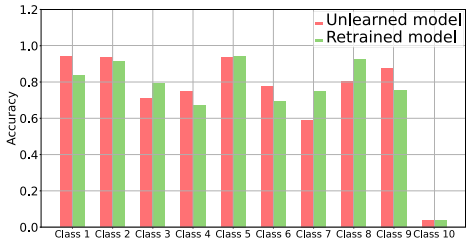
(a) Model accuracy on each class, MNIST dataset.



(b) Model accuracy on each class, CIFAR10 dataset.



(c) Model accuracy on each class, SVHN dataset.



(d) Model accuracy on each class, USPS dataset.

Figure 6.3: The accuracy of unlearned model and retrained model on different classes.

Figure 6.3(a) shows the accuracy of the unlearned model with respect to different classes on the MNIST dataset. Intuitively, the unlearned model should still be able to recognize the remaining 9 classes of data, but forget what it has learned about the last one. Figures 6.3(a) to 6.3(d) confirm this conjecture. In Figure 6.3(a), we observe that the unlearned model keeps high performance on Class1 to Class9, while the model's accuracy drops significantly on Class10. The reason is that the unlearning operation produces a model that is within a small distance of the retrained model, where the distance gap is then masked by differential privacy. Therefore, the performances of the unlearned model and the retrained model are guaranteed to be similar. A similar variation tendency can also be observed in Figures 6.3(b) to 6.3(d), which were obtained using different datasets and different models. This experiment demonstrates that the proposed FedRecovery is effective in removing client's influence in federated learning environment.

6.5.2.3 The efficiency of unlearning

The time consumption of the FedRecovery algorithm mainly comes from the computation of gradient residuals and the storage/readout of cached statistics. In this experiment, we evaluated the unlearning efficiency of the FedRecovery in terms of running time, and made comparisons with current unlearning solutions [81, 83, 143]. The retraining time of these methods varies depending on the specific federated learning framework they use. For a clear comparison, we reported the averaged retraining time here. The results are presented in Table 6.1.

From the table, we can observe that retraining from scratch is the most time-consuming unlearning strategy, which explains the motivation of current unlearning efforts. In comparison, other methods reduce the unlearning execution time to some extent. For example, reference [83] adopts the Fisher information matrix to approximate the Hessian, and thus accelerates the retraining procedure. In [81], the retraining update only happens in specific rounds, which reduces the model reconstruction time. Despite using speed-up techniques, references [83] and [81] require clients to perform local training for certain rounds. In addition, reference [143] requires clients to download the pruned model and conduct further fine-tuning. These unlearning strategies rely on subsequent training, which still takes execution time and incurs communication costs between clients and the server. Compared to these unlearning techniques, the FedRecovery algorithm avoids retraining and fine-tuning based model calibration. As a result, the running time is significantly reduced.

Table 6.1: The running time of unlearning methods, comparison

Datasets	Running time (s)				
	Retrain	ref [83]	ref [81]	ref [143]	**Ours**
MNIST	679.1	182.6	385.1	129.7	$< \mathbf{1.0}$
CIFAR10	2484.5	879.2	1082.3	683.9	$< \mathbf{2.0}$
SVHN	1380.6	587.1	622.4	311.4	$< \mathbf{1.0}$
USPS	285.2	99.4	137.8	64.7	$< \mathbf{1.0}$

6.5.2.4 Comparison with existing methods

Our next experiment involves a comparison of the model's performance between the proposed FedRecovery and existing machine unlearning algorithms. In this experiment, we

took $n = 10$ clients, each holding a partition of the original database. For references [83] and [81], the data are independent and identically distributed across clients due to the rationale of their methods. For reference [143], each client holds a specific class of data in the original database. For comparison purposes, we set the client deletion ratio to 0.1 and the data deletion ratio to 1.0 when applying their unlearning strategy. The local update epoch in their retraining and fine-tuning is set to 1, with learning rate ranges from 0.01 to 0.05 and batch size of 128. The results are shown in Table 6.2.

Table 6.2: The accuracy of unlearning methods, comparison

Datasets	Global model's accuracy (%)				
	Retrain	ref [83]	ref [81]	ref [143]	**Ours**
MNIST	97.4	97.2	94.1	96.8	**90.9**
CIFAR10	84.9	83.2	80.1	83.8	**71.2**
SVHN	84.6	82.3	80.4	75.3	**70.4**
USPS	90.3	89.4	84.7	90.1	**83.0**

From the table, we can observe that retraining from scratch achieves high accuracy on the four datasets. In comparison, the methods in [81, 83, 143] (the 2nd-4th columns) produce similar results that are comparable to retraining. The reason is that these methods were developed based on the idea of retraining and fine-tuning, which can be regarded as accelerated retraining variations. As for FedRecovery, it does not lead to accuracy as high as the previous methods. This is because the injection of the Gaussian noise perturbs the model's parameters, and thus inevitably decreases the model's performance. However, we note that this despondency is caused by the design rationale of the unlearning strategy: the FedRecovery algorithm relies on differentially private noise to achieve indistinguishability, which is elaborated for the scenario where retraining is unavailable. In addition, the execution time of the FedRecovery is significantly reduced compared to the retraining-based unlearning strategies. Therefore, the FedRecovery algorithm is competent at removing the impact of a particular client and is capable of guaranteeing a competitive performance for federated learning frameworks.

6.5.2.5 The performance of MIA on unlearned models

One of the fundamental purposes of machine unlearning is to protect the privacy of clients in federated learning. In this experiment, we conducted a membership inference attack to estimate how much information about the unlearned client is still contained in

the unlearned model. The aim of MIA is to infer whether the targeted sample was used to train the original model. Therefore, a lower attack precision means a more complete removal of a client's influence. We adopted the MIA strategy in [124] towards the target client's data to build an attack classifier. The results are presented in Table 6.3.

Table 6.3: Attack precision of MIA on unlearned models, comparison

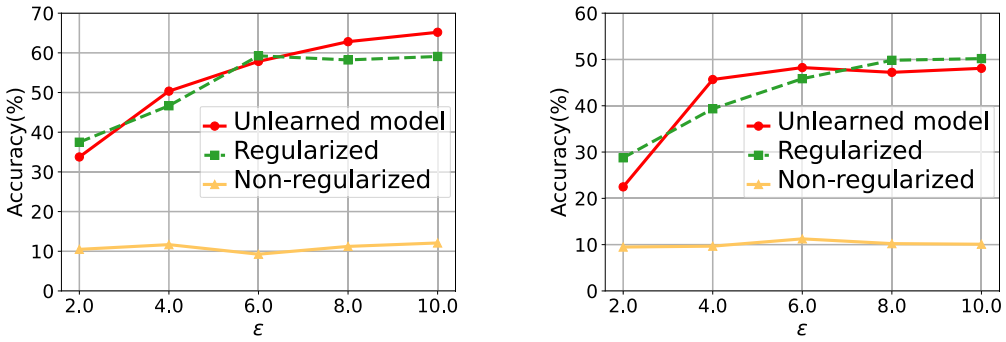
Datasets	Attack precision (%)					
	Origin	Retrain	ref [83]	ref [81]	ref [143]	**Ours**
MNIST	96.8	50.5	50.1	53.4	52.3	**50.9**
CIFAR10	72.7	49.8	51.8	51.5	53.5	**51.2**
SVHN	98.5	50.1	54.9	56.7	49.9	**50.4**
USPS	97.6	49.5	52.2	68.9	50.1	**53.0**

As can be seen from Table 6.3, the MIA has a high attack precision on the original models. In this scenario, the attacker is able to successfully figure out if the target client's data was used during the training. By contrast, after unlearning, the attack precision of MIA is reduced, which means that the attacker cannot decide if the targeted client's data was in the training dataset. Comparing the first and last columns, the drop in attack precision indicates that the impact of the target client is successfully removed by the proposed FedRecovery algorithm. Moreover, the results produced by FedRecovery are similar to those of the retrained model, which are also comparable to existing solutions [81, 83, 143]. This shows that even if FedRecovery does not rely on retraining or fine-tuning, it still yields effective unlearning results. This experiment demonstrates that the proposed unlearning algorithm indeed has the ability to remove a client's influence from a trained global model.

6.5.2.6 Discussions

Throughout the experiments, we start with pre-trained models because the distance between models before and after training will not change significantly. Therefore, according to Equations 6.20 and 6.21, the amount of noise added to models is small, and thus the models' performances can be guaranteed. For a model that begins from an arbitrary initial point, there is no assurance that the distance D_t in Equation 24 is small. As a result, the injected noise may significantly reduce the model's performance. Theoretically, this issue cannot be avoided because throughout the chapter we do not

presume convexity of loss functions. Equation 6.30's triangular inequality is the optimal distance approximation under the assumption of Lipschitzness.



(a) Acc. vs. ϵ, MNIST. (b) Acc. vs. ϵ, CIFAR10.

Figure 6.4: The accuracy of the unlearned model with regularization.

To circumvent this problem, we conducted additional experiments in which a regulation term was added to the loss function. In this experiment, the models were trained from random starting points without any pre-training. Regularization seeks to restrict the D_t distance between the trained model and the initial model. The outcomes are depicted in Figure 6.4. From the figure, we can observe that if the models are trained without regularization, the injected noise will substantially reduce the performance of the unlearned model. With a regularization term, the distance between the initial model and the rained model is constrained, enhancing the accuracy of the models. Therefore, introducing a regularization term is an effective way to improve the performance of models without pre-training. It is important to note, however, that regularization reduces the accuracy of both the unlearned and retrained models.

In this section, we investigated the influence of differential privacy in achieving statistical indistinguishability between models, which verified the fundamental idea of the proposed unlearning strategy. Furthermore, we also examined the unlearning ability of the proposed algorithm, along with its execution time and the model's performance after unlearning operations. The experimental results show that the proposed FedRecovery can efficiently remove the influence of a target client. Compared with current unlearning strategies that heavily rely on retraining and fine-tuning, our method is more suitable for real-world federated learning scenarios.

6.6 Summary

This chapter develops a differentially private machine unlearning algorithm for federated learning frameworks. The proposed FedRecovery algorithm removes the influence of the unlearned client from a trained global model, and adopts Gaussian noise to mask the gap between unlearned parameters and retrained parameters. We gave up the widely used convexity assumption, and derived the upper bound of distance between parameters trained with/without the unlearned client. The FedRecovery does not need any retraining-based fine-tuning or calibration. Theoretical analyses show that FedRecovery satisfies the desired unlearning guarantee, and experimental results demonstrate the performance of FedRecovery is competitive with the retrained model. Currently, FedRecovery efficiently supports pretrained models, but its effectiveness diminishes when training models from scratch. In addition, when raw gradients are no longer transmitted in federated learning, the performance of FedRecovery will be affected. In future work, it is possible to explore the exclusion of Lipschitz conditions and develop a more general unlearning algorithm in the context of complex neural networks, which may provide new insight into the efforts for machine unlearning tasks.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

This thesis investigates the intersection of differential privacy and game theory in cybersecurity. The proposed methods and experimental results demonstrate that game-theoretic and differentially private solutions are crucial for addressing privacy and security issues in the modern cyberworld.

To begin with, Chapter 2 reviews basic concepts in differential privacy and game theory. We highlighted the various roles differential privacy can play in solving game-theoretic problems and introduced how different differentially private methods can be implemented to fulfill these roles. Furthermore, we also provided essential background information regarding federated learning, APT attack and defense, machine unlearning, and conducted a comprehensive review of existing literature in relevant domains.

Chapter 3 explores the implementation of joint differential privacy in defending against adversarial clients in federated learning. We proposed two innovative mechanisms rooted in game theory that conceptualize client selection as an auction game. The clients provide their cost information in the form of bids, while the server employs various payment strategies to optimize its objectives. The framework facilitates the pursuit of various objectives, including the maximization of social welfare and the minimization of costs. Among the two mechanisms being considered, the first one possesses the desirable properties of being envy-free, dominant strategy truthful, and individual rational. The second proposition is similarly characterized by truthfulness and individual rationality,

while additionally granting the server the ability to strategically choose clients in order to fulfill its own distinct optimization objective. The overarching framework is characterized by joint differential privacy, thereby mitigating the influence of adversarial clients.

Chapter 4 discusses the feasibility of using games to improve the quality of training data while still protecting clients' sensitive information. The solution was developed based on congestion games. A private reward game was proposed as a means to incentivize clients in contributing high-quality data to federated learning. The reward system utilizes a mediator which provides recommendations to individual clients. In this context, the client's adherence to good behavior establishes an ex-post Nash equilibrium within the game of incomplete information. The reward system additionally guarantees the integrity of the data quality information pertaining to each individual client. The assurance of joint differential privacy is achieved through the utilization of differentially private counters. A novel approach was proposed to facilitate parameter aggregation through a private data trading method. The objective of this method was to encourage a greater number of clients to adopt a centralized differential privacy model, thereby enhancing the accuracy of the model. We presented comprehensive demonstrations and evaluations of the game's characteristics, accompanied by an investigation into the assurance of privacy. The proposed analysis offers a more comprehensive comprehension of the underlying reasoning behind the proposed game. Furthermore, we performed simulations using datasets from the real world in order to verify the efficacy of the framework. The findings indicate that, within the framework proposed, the model's accuracy surpasses that achieved through conventional federated learning methods.

Chapter 5 shows that it is tactical to make strategy adjustment when defending against adversarial persistent threats. We devised an APT rivalry game that serves as a means to depict the dynamics between assailants and protectors. The game under consideration takes into account the information leakage that occurs when players adjust their strategies. This provides an explanation for why players are becoming more powerful throughout the duration of an APT campaign. The necessary conditions for the existence of a family of equilibria were derived, and it was demonstrated that these equilibria are indicative of the optimal timing for each player to make a strategy adjustment. Furthermore, we have put forth two learning mechanisms with the aim of assisting defenders in identifying the most effective defense level and optimal allocation of resources. The empirical findings indicate that the game's equilibria result in increased utility, and the suggested mechanisms demonstrate their efficacy in learning from the players' previous encounters.

Chapter 6 explores how differential privacy can be employed to develop machine unlearning algorithms to remove the influence of a client in federated learning. A novel FedRecovery algorithm for machine unlearning in federated learning frameworks was developed. The algorithm aims to mitigate the impact of unlearned clients on a trained global model. This is achieved by incorporating Gaussian noise, which serves to conceal the discrepancy between unlearned parameters and retrained parameters. The convexity assumption, which is commonly employed, was abandoned in this study. Consequently, we derived an upper bound for the distance between parameters that were trained with and without the unlearned client. The proposed system does not necessitate any retraining-based fine-tuning or calibration. Theoretical analyses indicate that FedRecovery successfully fulfills the desired unlearning guarantee, while experimental results reveal that the performance of FedRecovery is comparable to that of the retrained model.

Differential privacy and game theory have proven to be powerful tools in cybersecurity. With cybersecurity drawing increasingly attention in people's lives, these theories and methods will have much more potential and untapped benefits.

7.2 Future work

Variant differential privacy plays an important role in solving security and privacy problems. After all, differential privacy was initially proposed to protect the privacy of a single individual [34]. It cannot give too much consideration to maintaining game-theoretic properties. Nissim et al. [107] and Xiao [152] have pointed out that the classic differential privacy actually makes every report an approximate dominant strategy, which is the main drawback of employing it in algorithm design. For cybersecurity, such a rigorous requirement may lead to trivial outcomes, making it hard to analyze agents' behaviors. Therefore, relaxed differential privacy is necessary to address this issue. Kearns et al. [69] proposed joint differential privacy that allows the outcome of each agent to be sensitive to their own data. Kannan et al. [66] further relaxed the size of the outcome where differential privacy is defined, called marginal differential privacy. These differential privacy definitions stem from specific cybersecurity problems, providing strong a privacy guarantee while circumventing the impossibility of the original definition. Therefore, it would be worthwhile to propose variant differential privacy definitions with respect to different real-world scenarios.

Local differential privacy [67] has been a booming field, with several representative privacy preserving techniques coming out in the past decade [12, 38, 144]. The

combination of local differential privacy and cybersecurity has great potential. Most existing mechanisms take agents' private information (e.g., valuations, preferences) as input and generate output that fulfills desired properties (e.g., differential privacy, approximate truthfulness). Although these mechanisms themselves are differentially private, it is natural to ask: who is responsible for running these mechanisms? What if the mechanism's executor is malicious or vulnerable? Local differential privacy allows agents to add noise to their data locally before sending it to the executor, which avoids the concern of a privacy breach in the above process. Wang et al. [145] investigated private data trading under ϵ-local differential privacy, but only dealt with binary data variables. From the perspective of privacy, truthfulness and local differential privacy seem contradictory – now that agents submit their noisy data (e.g., bidding information), is it *really* important to generate them using true ones? In addition, it is also unknown how noisy input data influences the outcome of a mechanism. Therefore, it may be interesting to explore the combination of local differential privacy and cybersecurity, as well as re-establishing the relationship between local differentially private properties and game-theoretic desiderata.

There are also privacy and security issues with multi-agent systems [25, 26], differential privacy could be employed, not only for privacy preservation, but also for enhancing robustness and improving performance. Ye et al. [159] developed a differentially private mechanism for multi-agent advising learning by adding Laplace noise to the reward of student agents. In their work, the aim of introducing differential privacy is to avoid malicious agents. In [169], Zhao et al. proposed a privacy-preserving collaborative deep learning scheme by perturbing the objective function of a neural network. In their study, differential privacy is used not only for protecting the privacy of training data but also for reducing the impact of unreliable participants and obtaining improved stability. Such consideration could also be tailored in federated learning [157]. Suppose there is a malicious agent that tries to cause the misclassification of a model by model poisoning. If the server adopts a differentially private weight aggregation method, then the impact of the malicious agent might be limited. Moreover, it is also worth attempting to address the privacy issue for classic algorithms in multi-agent systems. For example, differentially private simultaneous ascending auction (SAA) [53] and differentially private top trading cycle mechanism (TTC) [66].

The application of game theory in cybersecurity provides significant advantages by providing comprehension, prediction, and mitigation of potential security risks. Game theory offers a mathematical framework that enables the modeling of intricate scenarios

involving both conflict and cooperation, thereby allowing its application to the realm of cyber conflicts [114]. It is possible to predict the actions of attackers and defenders by treating them as players, analyzing the potential strategies they can adopt, and evaluating the outcomes they stand to gain. This analytical approach enables the development of optimal defense strategies. In addition, the utilization of game theory enables professionals to enhance the efficacy of defense mechanisms through the examination of possible attack scenarios and the corresponding countermeasures [167]. The anticipated outcomes of players' strategies are related to the extent of damage incurred from the attack and the expenses associated with defense. Through this approach, it becomes possible to discern the most advantageous strategies for both participants involved.

Game theory enables cybersecurity experts to analyze the conduct and tactics employed by potential adversaries, thereby facilitating the identification of potential risks and susceptibilities [165]. This helps in enhancing the understanding of decision-making mechanisms employed by potential attackers and in devising more effective defensive strategies. In addition, the application of game theory extends to the quantification of risks pertaining to various cybersecurity threats [63]. By conceptualizing the interactions between attackers and defenders as a strategic game, it becomes possible to approximate the potential harm and expenses linked to various forms of cyber attacks. For intrusion detection systems, game-theoretic models can has the potential to enhance the resilience of systems against sophisticated evasion techniques [172]. In scenarios characterized by limited resources for cybersecurity, the application of game theory can facilitate the optimal allocation of resources by modeling the different areas of a system that can be defended and the different types of attacks that could be carried out [146].

Game theory offers a resilient and adaptable framework for the modeling and analysis of cybersecurity scenarios. It provides valuable insights into anticipating potential attacks, designing effective defenses, and optimizing resource allocation by taking into account different strategies, payoffs, and information available to each player involved.

Currently, the boom of large language models (LLMs) poses new privacy and security challenges for model owners and users. LLMs are trained on extensive collections of online data, which frequently include copyrighted material. As a result, when interacting with users, the model may produce text derived from sensitive or confidential information it has learned during training, potentially disclosing information that was meant to be private or insecure [125]. Researchers have not thoroughly investigated the effectiveness of differential privacy in addressing these privacy concerns. The existing solutions for balancing utility and privacy tradeoffs, determining budget allocation, and ensuring

scalability are not immediately applicable to LLM cases. Meanwhile, further exploration is needed to understand the roles of game theory in preventing inferences and attacks throughout the LLM interacting process, considering the current capabilities of LLMs to execute strategic decision-making and strategic reasoning.

BIBLIOGRAPHY

[1] M. ABADI, A. CHU, I. GOODFELLOW, H. B. MCMAHAN, I. MIRONOV, K. TALWAR, AND L. ZHANG, *Deep learning with differential privacy*, in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, New York, NY, USA, 2016, Association for Computing Machinery, pp. 308–318.

[2] N. AGARWAL, P. KAIROUZ, AND Z. LIU, *The skellam mechanism for differentially private federated learning*, in Advances in Neural Information Processing Systems, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds., vol. 34, Curran Associates, Inc., 2021, pp. 5052–5064.

[3] A. ALSHAMRANI, S. MYNENI, A. CHOWDHARY, AND D. HUANG, *A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities*, IEEE Communications Surveys & Tutorials, 21 (2019), pp. 1851–1877.

[4] P. AUER, N. CESA-BIANCHI, Y. FREUND, AND R. E. SCHAPIRE, *The nonstochastic multiarmed bandit problem*, SIAM Journal on Computing, 32 (2002), pp. 48–77.

[5] E. BAGDASARYAN, A. VEIT, Y. HUA, D. ESTRIN, AND V. SHMATIKOV, *How to backdoor federated learning*, in Procs of the 23rd International Conference on Artificial Intelligence and Statistics, S. Chiappa and R. Calandra, eds., vol. 108 of Proceedings of Machine Learning Research, Online, 26–28 Aug 2020, PMLR, pp. 2938–2948.

[6] B. BALLE, J. BELL, A. GASCÓN, AND K. NISSIM, *The privacy blanket of the shuffle model*, in Advances in Cryptology – CRYPTO 2019, A. Boldyreva and D. Micciancio, eds., Cham, 2019, Springer International Publishing, pp. 638–667.

[7] A. N. BHAGOJI, S. CHAKRABORTY, P. MITTAL, AND S. CALO, *Analyzing federated learning through an adversarial lens*, in Proceedings of the 36th ICML, vol. 97 of Proceedings of Machine Learning Research, PMLR, 09–15 Jun 2019, pp. 634–643.

[8] R. BOLLAPRAGADA, J. NOCEDAL, D. MUDIGERE, H.-J. SHI, AND P. T. P. TANG, *A progressive batching l-BFGS method for machine learning*, in Proceedings of the 35th International Conference on Machine Learning, vol. 80 of PMLR, PMLR, 10–15 Jul 2018, pp. 620–629.

[9] L. BOURTOULE, V. CHANDRASEKARAN, C. A. CHOQUETTE-CHOO, H. JIA, A. TRAVERS, B. ZHANG, D. LIE, AND N. PAPERNOT, *Machine unlearning*, in 2021 IEEE Symp. on Security and Privacy (SP), 2021, pp. 141–159.

[10] P. BRIEST, P. KRYSTA, AND B. VÖCKING, *Approximation techniques for utilitarian mechanism design*, in Procs of the 37th Annual ACM Symposium on Theory of Computing, STOC '05, New York, NY, USA, 2005, Association for Computing Machinery, pp. 39–48.

[11] J. BULOW AND P. KLEMPERER, *The generalized war of attrition*, The American Economic Review, 89 (1999), pp. 175–189.

[12] M. BUN, J. NELSON, AND U. STEMMER, *Heavy hitters and the structure of local privacy*, ACM Trans. Algorithms, 15 (2019).

[13] M. BUN AND T. STEINKE, *Concentrated differential privacy: Simplifications, extensions, and lower bounds*, in Proceedings, Part I, of the 14th International Conference on Theory of Cryptography - Volume 9985, Berlin, Heidelberg, 2016, Springer-Verlag, pp. 635–658.

[14] Y. CAO AND J. YANG, *Towards making systems forget with machine unlearning*, in 2015 IEEE Symposium on Security and Privacy, 2015, pp. 463–480.

[15] T.-H. H. CHAN, E. SHI, AND D. SONG, *Private and continual release of statistics*, ACM Trans. Inf. Syst. Secur., 14 (2011).

[16] T.-H. H. CHAN, E. SHI, AND D. SONG, *Optimal lower bound for differentially private multi-party aggregation*, in Proceedings of the 20th Annual European Conference on Algorithms, ESA'12, Berlin, Heidelberg, 2012, Springer-Verlag, pp. 277–288.

[17] K. CHAUDHURI, C. MONTELEONI, AND A. D. SARWATE, *Differentially private empirical risk minimization*, J. Mach. Learn. Res., 12 (2011), pp. 1069–1109.

[18] C. CHEN, F. SUN, M. ZHANG, AND B. DING, *Recommendation unlearning*, in Proceedings of the ACM Web Conference 2022, WWW '22, Association for Computing Machinery, 2022, pp. 2768–2777.

[19] L. CHEN, H. WANG, Z. CHARLES, AND D. PAPAILIOPOULOS, *DRACO: Byzantine-resilient distributed training via redundant gradients*, in Procs of the 35th Inter Conf on Machine Learning, J. Dy and A. Krause, eds., vol. 80 of Procs of Machine Learning Research, Stockholm Sweden, 10–15 Jul 2018, PMLR, pp. 903–912.

[20] A. S. CHIVUKULA, X. YANG, W. LIU, T. ZHU, AND W. ZHOU, *Game theoretical adversarial deep learning with variational adversaries*, IEEE Transactions on Knowledge and Data Engineering, 33 (2021), pp. 3568–3581.

[21] L. CHIZAT, E. OYALLON, AND F. BACH, *On lazy training in differentiable programming*, in Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019.

[22] E. COLE, *Advanced Persistent Threat: Understanding the Danger and How to Protect Your Organization*, Syngress Publishing, 1st ed., 2012.

[23] R. CUMMINGS, S. IOANNIDIS, AND K. LIGETT, *Truthful linear regression*, in Proceedings of the 28th Conference on Learning Theory, COLT 2015, Paris, France, 2015, pp. 448–483.

[24] R. CUMMINGS, M. KEARNS, A. ROTH, AND Z. S. WU, *Privacy and truthful equilibrium selection for aggregative games*, in Web and Internet Economics, E. Markakis and G. Schäfer, eds., Berlin, Heidelberg, 2015, Springer Berlin Heidelberg, pp. 286–299.

[25] F. L. DA SILVA AND A. H. R. COSTA, *A survey on transfer learning for multiagent reinforcement learning systems*, J. Artif. Int. Res., 64 (2019), pp. 645–703.

[26] F. L. DA SILVA, R. GLATT, AND A. H. R. COSTA, *Simultaneously learning and advising in multiagent reinforcement learning*, in Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS '17,

Richland, SC, 2017, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1100–1108.

[27] M. K. DALY, *Advanced persistent threat*, Usenix, Nov, 4 (2009), pp. 2013–2016.

[28] P. DELGADO-SANTOS, G. STRAGAPEDE, R. TOLOSANA, R. GUEST, F. DERAVI, AND R. VERA-RODRIGUEZ, *A survey of privacy vulnerabilities of mobile device sensors*, ACM Comput. Surv., 54 (2022).

[29] E. DIANA, M. KEARNS, S. NEEL, AND A. ROTH, *Optimal, truthful, and private securities lending*, arXiv:1912.06202 [cs, q-fin], (2019).

[30] M. DIJK, A. JUELS, A. OPREA, AND R. L. RIVEST, *Flipit: The game of "stealthy takeover"*, J. Cryptol., 26 (2013), pp. 655–713.

[31] C. T. DO, N. H. TRAN, C. HONG, C. A. KAMHOUA, K. A. KWIAT, E. BLASCH, S. REN, N. PISSINOU, AND S. S. IYENGAR, *Game theory for cyber security and privacy*, ACM Comput. Surv., 50 (2017).

[32] C. DWORK, *Differential privacy*, in Automata, Languages and Programming, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, eds., Berlin, Heidelberg, 2006, Springer Berlin Heidelberg, pp. 1–12.

[33] ——, *A firm foundation for private data analysis*, Commun. ACM, 54 (2011), pp. 86–95.

[34] ——, *Differential privacy and the us census*, in Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS '19, New York, NY, USA, 2019, Association for Computing Machinery, p. 1.

[35] C. DWORK, F. MCSHERRY, K. NISSIM, AND A. SMITH, *Calibrating noise to sensitivity in private data analysis*, in Theory of Cryptography, S. Halevi and T. Rabin, eds., Berlin, Heidelberg, 2006, Springer Berlin Heidelberg, pp. 265–284.

[36] C. DWORK, M. NAOR, T. PITASSI, AND G. N. ROTHBLUM, *Differential privacy under continual observation*, in Proceedings of the Forty-Second ACM Symposium on Theory of Computing, STOC'10, New York, NY, USA, 2010, Association for Computing Machinery, pp. 715–724.

[37] C. DWORK AND A. ROTH, *The Algorithmic Foundations of Differential Privacy*, now, 2014.

[38] U. ERLINGSSON, V. PIHUR, AND A. KOROLOVA, *Rappor: Randomized aggregatable privacy-preserving ordinal response*, in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, New York, NY, USA, 2014, Association for Computing Machinery, pp. 1054–1067.

[39] EU, *Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (gdpr)*, May 2016.

[40] M. FREDRIKSON, S. JHA, AND T. RISTENPART, *Model inversion attacks that exploit confidence information and basic countermeasures*, in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015, pp. 1322–1333.

[41] D. FUDENBERG AND J. TIROLE, *A theory of exit in duopoly*, Econometrica, 54 (1986), pp. 943–960.

[42] A. GHOSH AND A. ROTH, *Selling privacy at auction*, in Proceedings of the 12th ACM Conference on Electronic Commerce, EC'11, New York, NY, USA, 2011, Association for Computing Machinery, pp. 199–208.

[43] A. GHOSH, T. ROUGHGARDEN, AND M. SUNDARARAJAN, *Universally utility-maximizing privacy mechanisms*, in Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, STOC'09, New York, NY, USA, 2009, Association for Computing Machinery, pp. 351–360.

[44] A. A. GINART, M. Y. GUAN, G. VALIANT, AND J. ZOU, *Making ai forget you: Data deletion in machine learning*, in Proceedings of the 33rd International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 2019, Curran Associates Inc.

[45] A. GIRGIS, D. DATA, S. DIGGAVI, P. KAIROUZ, AND A. THEERTHA SURESH, *Shuffled model of differential privacy in federated learning*, in Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, A. Banerjee and K. Fukumizu, eds., vol. 130 of Proceedings of Machine Learning Research, PMLR, 13–15 Apr 2021, pp. 2521–2529.

[46] A. GOLATKAR, A. ACHILLE, AND S. SOATTO, *Eternal sunshine of the spotless net: Selective forgetting in deep networks*, in 2020 IEEE/CVF Conf. on Comput. Vision and Pattern Recognition (CVPR), 2020, pp. 9301–9309.

[47] ——, *Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations*, in Computer Vision – ECCV, 2020, pp. 383–398.

[48] GOOGLE-RESEARCH, *Federated learning: Collaborative machine learning without centralized training data*, April 2017.

[49] L. GRAVES, V. NAGISETTY, AND V. GANESH, *Amnesiac machine learning*, Proceedings of the AAAI Conference on Artificial Intelligence, 35 (2021), pp. 11516–11524.

[50] C. GUO, T. GOLDSTEIN, A. HANNUN, AND L. VAN DER MAATEN, *Certified data removal from machine learning models*, in Proceedings of the 37th ICML, ICML'20, JMLR.org, 2020.

[51] M. HAO, H. LI, G. XU, S. LIU, AND H. YANG, *Towards efficient and privacy-preserving federated deep learning*, in ICC 2019 - 2019 IEEE International Conference on Communications (ICC), 2019, pp. 1–6.

[52] E. L. HARDING, J. J. VANTO, R. CLARK, AND S. C. HANNAH JI, L.AND AINSWORTH, *Understanding the scope and impact of the california consumer privacy act of 2018*, Journal of Data Protection & Privacy, 2 (2019), pp. 234–253.

[53] J. HSU, Z. HUANG, A. ROTH, T. ROUGHGARDEN, AND Z. S. WU, *Private matchings and allocations*, in Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, STOC'14, New York, NY, USA, 2014, Association for Computing Machinery, pp. 21–30.

[54] R. HU, Y. GUO, H. LI, Q. PEI, AND Y. GONG, *Personalized federated learning with differential privacy*, IEEE Internet of Things Journal, 7 (2020), pp. 9530–9539.

[55] L. HUANG AND Q. ZHU, *Analysis and computation of adaptive defense strategies against advanced persistent threats for cyber-physical systems*, in Decision and Game Theory for Security, L. Bushnell, R. Poovendran, and T. Başar, eds., Cham, 2018, Springer International Publishing, pp. 205–226.

[56] Y. HUANG, S. GUPTA, Z. SONG, K. LI, AND S. ARORA, *Evaluating gradient inversion attacks and defenses in federated learning*, in NIPS, vol. 34, Curran Associates, Inc., 2021, pp. 7232–7241.

[57] Z. HUANG, J. LIU, AND X. WANG, *Learning optimal reserve price against non-myopic bidders*, in Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18, Red Hook, NY, USA, 2018, Curran Associates Inc., pp. 2042–2052.

[58] J. HULL, *A database for handwritten text recognition research*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 16 (1994), pp. 550–554.

[59] L. HURWICZ, E. S. MASKIN, AND R. B, *Intelligent design*, The Economist, (2007).

[60] Z. IZZO, M. ANNE SMART, K. CHAUDHURI, AND J. ZOU, *Approximate data deletion from machine learning models*, in Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, vol. 130 of PMLR, PMLR, 13–15 Apr 2021, pp. 2008–2016.

[61] A. JACOT, F. GABRIEL, AND C. HONGLER, *Neural tangent kernel: Convergence and generalization in neural networks*, in Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., 2018.

[62] H. JIANG, U. V. SHANBHAG, AND S. P. MEYN, *Distributed computation of equilibria in misspecified convex stochastic nash games*, IEEE Transactions on Automatic Control, 63 (2018), pp. 360–371.

[63] R. JIN, X. HE, AND H. DAI, *On the security-privacy tradeoff in collaborative security: A quantitative information flow game perspective*, IEEE Transactions on Information Forensics and Security, 14 (2019), pp. 3273–3286.

[64] X. JIN AND Y. ZHANG, *Privacy-preserving crowdsourced spectrum sensing*, IEEE/ACM Transactions on Networking, 26 (2018), pp. 1236–1249.

[65] J. KANG, Z. XIONG, D. NIYATO, S. XIE, AND J. ZHANG, *Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory*, IEEE Internet of Things Journal, 6 (2019), pp. 10700–10714.

[66] S. KANNAN, J. MORGENSTERN, R. ROGERS, AND A. ROTH, *Private pareto optimal exchange*, ACM Trans. Econ. Comput., 6 (2018).

[67] S. P. KASIVISWANATHAN, H. K. LEE, K. NISSIM, S. RASKHODNIKOVA, AND A. SMITH, *What can we learn privately?*, in 2008 49th Annual IEEE Symposium on Foundations of Computer Science, 2008, pp. 531–540.

[68] H. KAYAN, M. NUNES, O. RANA, P. BURNAP, AND C. PERERA, *Cybersecurity of industrial cyber-physical systems: A review*, ACM Comput. Surv., 54 (2022).

[69] M. KEARNS, M. PAI, A. ROTH, AND J. ULLMAN, *Mechanism design in large games: Incentives and privacy*, in Proceedings of the 5th Conference on Innovations in Theoretical Computer Science, ITCS'14, New York, NY, USA, 2014, Association for Computing Machinery, pp. 403–410.

[70] K. KLEIN AND I. A. OF PRIVACY PROFESSIONALS, *Canadian Privacy: Data Protection and Policy for the Practitioner*, International Association of Privacy Professionals, 2020.

[71] P. W. KOH AND P. LIANG, *Understanding black-box predictions via influence functions*, in Proceedings of the 34th International Conference on Machine Learning - Volume 70, JMLR.org, 2017, pp. 1885–1894.

[72] N. KOHLI AND P. LASKOWSKI, *Epsilon voting: Mechanism design for parameter selection in differential privacy*, in 2018 IEEE Symposium on Privacy-Aware Computing (PAC), Sep. 2018, pp. 19–30.

[73] A. KRIZHEVSKY, G. HINTON, ET AL., *Learning multiple layers of features from tiny images*, 2009.

[74] G. LAN, *Nonconvex Optimization*, Springer International Publishing, Cham, 2020, pp. 305–420.

[75] Y. LECUN, *The mnist database of handwritten digits*, http://yann. lecun. com/exdb/mnist/, (1998).

[76] J. LEE, L. XIAO, S. S. SCHOENHOLZ, Y. BAHRI, R. NOVAK, J. SOHL-DICKSTEIN, AND J. PENNINGTON, *Wide neural networks of any depth evolve as linear models under gradient descent*, in Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., 2019.

[77] A. LEMAY, J. CALVET, F. MENET, AND J. M. FERNANDEZ, *Survey of publicly available reports on advanced persistent threat actors*, Computers & Security, 72 (2018), pp. 26–59.

[78] W. Y. B. LIM, Z. XIONG, J. KANG, D. NIYATO, C. LEUNG, C. MIAO, AND X. SHEN, *When information freshness meets service latency in federated learning: A task-aware incentive scheme for smart industries*, IEEE Transactions on Industrial Informatics, 18 (2022), pp. 457–466.

[79] B. LIU, M. DING, S. SHAHAM, W. RAHAYU, F. FAROKHI, AND Z. LIN, *When machine learning meets privacy: A survey and outlook*, ACM Comput. Surv., 54 (2021).

[80] C. LIU, T. ZHU, J. ZHANG, AND W. ZHOU, *Privacy intelligence: A survey on image privacy in online social networks*, ACM Comput. Surv., 55 (2022).

[81] G. LIU, X. MA, Y. YANG, C. WANG, AND J. LIU, *Federaser: Enabling efficient client-level data removal from federated learning models*, in 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), 2021, pp. 1–10.

[82] X. LIU, K. WAN, Y. DING, X. ZHANG, AND Q. ZHU, *Weighted-sampling audio adversarial example attack*, Proceedings of the AAAI Conference on Artificial Intelligence, 34 (2020), pp. 4908–4915.

[83] Y. LIU, L. XU, X. YUAN, C. WANG, AND B. LI, *The right to be forgotten in federated learning: An efficient realization with rapid retraining*, in IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, 2022, pp. 1749–1758.

[84] S. K. LO, Q. LU, C. WANG, H.-Y. PAIK, AND L. ZHU, *A systematic literature review on federated machine learning: From a software engineering perspective*, ACM Comput. Surv., 54 (2021).

[85] A. LUIGI, G. NICOLA, AND S. GIUSEPPE, *Gradient Flows: in Metric Spaces and in the Space of Probability Measures*, Birkhäuser Basel, Basel, Switzerland, 1 ed., 2005.

[86] M. H. MANSHAEI, Q. ZHU, T. ALPCAN, T. BACŞAR, AND J.-P. HUBAUX, *Game theory meets network security and privacy*, ACM Comput. Surv., 45 (2013).

[87] V. MATTA, M. DI MAURO, M. LONGO, AND A. FARINA, *Cyber-threat mitigation exploiting the birth–death–immigration model*, IEEE Trans. on Info. Forensics and Security, 13 (2018), pp. 3137–3152.

[88] J. MAYNARD SMITH, *The theory of games and the evolution of animal conflicts*, J. of Theoretical Biology, 47 (1974), pp. 209–221.

[89] F. MCSHERRY, *Privacy integrated queries: An extensible platform for privacy-preserving data analysis*, Commun. ACM, 53 (2010), pp. 89–97.

[90] F. MCSHERRY AND K. TALWAR, *Mechanism design via differential privacy*, in 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), IEEE, 2007, pp. 94–103.

[91] P. MELL, K. SCARFONE, AND S. ROMANOSKY, *Common vulnerability scoring system*, Security & Privacy, IEEE, 4 (2006), pp. 85–89.

[92] V. MNIH, K. KAVUKCUOGLU, D. SILVER, A. GRAVES, I. ANTONOGLOU, D. WIERSTRA, AND M. RIEDMILLER, *Playing atari with deep reinforcement learning*, arXiv preprint arXiv:1312.5602, (2013).

[93] V. MNIH, K. KAVUKCUOGLU, D. SILVER, A. A. RUSU, J. VENESS, M. G. BELLEMARE, A. GRAVES, M. RIEDMILLER, A. K. FIDJELAND, G. OSTROVSKI, S. PETERSEN, C. BEATTIE, A. SADIK, I. ANTONOGLOU, H. KING, D. WIERSTRA, S. LEGG, AND D. HASSABIS, *Human-level control through deep reinforcement learning*, Nature, 518 (2015), pp. 529–533.

[94] D. MONDERER AND L. S. SHAPLEY, *Potential games*, Games and Economic Behavior, 14 (1996), pp. 124–143.

[95] S. MOOTHEDATH, D. SAHABANDU, J. ALLEN, A. CLARK, L. BUSHNELL, W. LEE, AND R. POOVENDRAN, *A game-theoretic approach for dynamic information flow tracking to detect multistage advanced persistent threats*, IEEE Trans. on Automatic Control, 65 (2020), pp. 5248–5263.

[96] A. MU'ALEM AND N. NISAN, *Truthful approximation mechanisms for restricted combinatorial auctions*, Games and Economic Behavior, 64 (2008), pp. 612 – 631.

[97] D. MYATT, *Instant Exit from the War of Attrition*, Economics Papers 9922, Economics Group, Nuffield College, University of Oxford, 1999.

[98] R. B. MYERSON, *Optimal auction design*, Mathematics of Operations Research, 6 (1981), pp. 58–73.

[99] ——, *Game Theory: Analysis of Conflict*, Harvard University Press, 1991.

[100] P. NAGHIZADEH AND A. SINHA, *Adversarial contract design for private data commercialization*, in Proceedings of the 2019 ACM Conference on Economics and Computation, EC '19, New York, NY, USA, 2019, Association for Computing Machinery, pp. 681–699.

[101] J. NASH, *Non-cooperative games*, Annals of Mathematics, 54 (1951), pp. 286–295.

[102] J. F. NASH, *Equilibrium points in n-person games*, Proceedings of the National Academy of Sciences, 36 (1950), pp. 48–49.

[103] S. NEEL, A. ROTH, AND S. SHARIFI-MALVAJERDI, *Descent-to-delete: Gradient-based methods for machine unlearning*, in Proceedings of the 32nd International Conference on Algorithmic Learning Theory, vol. 132 of PMLR, PMLR, 16–19 Mar 2021, pp. 931–962.

[104] Y. NETZER, T. WANG, A. COATES, A. BISSACCO, B. WU, AND A. Y. NG, *Reading digits in natural images with unsupervised feature learning*, in NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.

[105] K. C. NGUYEN, T. ALPCAN, AND T. BASAR, *Security games with incomplete information*, in 2009 IEEE International Conference on Communications, 2009, pp. 1–6.

[106] T. T. NGUYEN, T. T. HUYNH, P. L. NGUYEN, A. W.-C. LIEW, H. YIN, AND Q. V. H. NGUYEN, *A survey of machine unlearning*, 2022.

[107] K. NISSIM, R. SMORODINSKY, AND M. TENNENHOLTZ, *Approximately optimal mechanism design via differential privacy*, in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS'12, New York, NY, USA, 2012, Association for Computing Machinery, pp. 203–213.

[108] C. NIU, Z. ZHENG, F. WU, S. TANG, X. GAO, AND G. CHEN, *Unlocking the value of privacy: Trading aggregate statistics over private correlated data*, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'18, New York, NY, USA, 2018, Association for Computing Machinery, pp. 2031–2040.

[109] M. OSBORNE AND A. RUBINSTEIN, *A Course in Game Theory*, The MIT Press, MIT Press, 1994.

[110] J. H. PAIK, *A novel tf-idf weighting scheme for effective ranking*, in Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, New York, NY, USA, 2013, ACM, pp. 343–352.

[111] C. PAULSEN, *Glossary of key information security terms*, tech. rep., National Institute of Standards and Technology, 2018.

[112] C. PAULSEN, E. MCDUFFIE, W. NEWHOUSE, AND P. TOTH, *Nice: Creating a cybersecurity workforce and aware public*, IEEE Security & Privacy, 10 (2012), pp. 76–79.

[113] D. PAVLOVIC, *Gaming security by obscurity*, European Economic Review - EUR ECON REV, (2011).

[114] J. PAWLICK AND Q. ZHU, *A stackelberg game perspective on the conflict between machine learning and data obfuscation*, in 2016 IEEE International Workshop on Information Forensics and Security (WIFS), 2016, pp. 1–6.

[115] N. PHAN, M. N. VU, Y. LIU, R. JIN, D. DOU, X. WU, AND M. T. THAI, *Heterogeneous gaussian mechanism: Preserving differential privacy in deep learning with provable robustness*, in Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 4753–4759.

[116] L. T. PHONG, Y. AONO, T. HAYASHI, L. WANG, AND S. MORIAI, *Privacy-preserving deep learning via additively homomorphic encryption*, IEEE Transactions on Information Forensics and Security, 13 (2018), pp. 1333–1345.

[117] S. QUINTERO-BONILLA AND A. MARTIN DEL REY, *A new proposal on the advanced persistent threat: A survey*, AppSci, 10 (2020).

[118] S. RASS, S. KÖNIG, AND E. PANAOUSIS, *Cut-the-rope: A game of stealthy intrusion*, in Decision and Game Theory for Security, T. Alpcan, Y. Vorobeychik, J. S. Baras, and G. Dán, eds., 2019, pp. 404–416.

[119] R. M. ROGERS AND A. ROTH, *Asymptotically truthful equilibrium selection in large congestion games*, in Proceedings of the Fifteenth ACM Conference on Economics and Computation, EC '14, New York, NY, USA, 2014, ACM, pp. 771–782.

[120] A. SANJAB, W. SAAD, AND T. BASAR, *A game of drones: Cyber-physical security of time-critical uav applications with cumulative prospect theory perceptions and valuations*, IEEE Transactions on Communications, 68 (2020), pp. 6990–7006.

[121] K. SCAMAN AND A. VIRMAUX, *Lipschitz regularity of deep neural networks: Analysis and efficient estimation*, in Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18, 2018, pp. 3839–3848.

[122] A. SEKHARI, J. ACHARYA, G. KAMATH, AND A. T. SURESH, *Remember what you want to forget: Algorithms for machine unlearning*, in Advances in Neural Information Processing Systems, vol. 34, Curran Associates, Inc., 2021, pp. 18075–18086.

[123] S. SENGUPTA, A. CHOWDHARY, D. HUANG, AND S. KAMBHAMPATI, *General sum markov games for strategic detection of advanced persistent threats using moving target defense in cloud networks*, in GameSec, 2019.

[124] R. SHOKRI, M. STRONATI, C. SONG, AND V. SHMATIKOV, *Membership inference attacks against machine learning models*, in 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 3–18.

[125] N. SI, H. ZHANG, H. CHANG, W. ZHANG, D. QU, AND W. ZHANG, *Knowledge unlearning for llms: Tasks, methods, and challenges*, 2023.

[126] N. SUN, J. ZHANG, P. RIMBA, S. GAO, L. Y. ZHANG, AND Y. XIANG, *Data-driven cybersecurity incident prediction: A survey*, IEEE Communications Surveys & Tutorials, 21 (2019), pp. 1744–1772.

[127] P. SUN, H. CHE, Z. WANG, Y. WANG, T. WANG, L. WU, AND H. SHAO, *Pain-fl: Personalized privacy-preserving incentive for federated learning*, IEEE Journal on Selected Areas in Communications, 39 (2021), pp. 3805–3820.

[128] M. TANG AND V. W. WONG, *An incentive mechanism for cross-silo federated learning: A public goods perspective*, in IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021, pp. 1–10.

[129] C. TANKARD, *Advanced persistent threats and how to monitor and deter them*, Network Security, 2011 (2011), pp. 16–19.

[130] A. K. TARUN, V. S. CHUNDAWAT, M. MANDAL, AND M. KANKANHALLI, *Fast yet effective machine unlearning*, 2021.

[131] T. H. THI LE, N. H. TRAN, Y. K. TUN, M. N. H. NGUYEN, S. R. PANDEY, Z. HAN, AND C. S. HONG, *An incentive mechanism for federated learning in wireless cellular networks: An auction approach*, IEEE Transactions on Wireless Communications, 20 (2021), pp. 4874–4887.

[132] D. TIAN, J. ZHOU, Y. WANG, Z. SHENG, X. DUAN, AND V. C. M. LEUNG, *Channel access optimization with adaptive congestion pricing for cognitive vehicular networks: An evolutionary game approach*, IEEE Transactions on Mobile Computing, 19 (2020), pp. 803–820.

[133] W. TIAN, M. DU, X. JI, G. LIU, Y. DAI, AND Z. HAN, *Honeypot detection strategy against advanced persistent threats in industrial internet of things: A prospect theoretic game*, IEEE IoT Journal, (2021), pp. 1–1.

[134] Y. TIAN, Y. WAN, L. LYU, D. YAO, H. JIN, AND L. SUN, *Fedbert: When federated learning meets pre-training*, ACM Trans. Intell. Syst. Technol., 13 (2022).

[135] D. TOSH, S. SENGUPTA, C. A. KAMHOUA, AND K. A. KWIAT, *Establishing evolutionary game models for cyber security information exchange (cybex)*, J. of Computer and System Sciences, 98 (2018), pp. 27–52.

[136] M. USSATH, D. JAEGER, F. CHENG, AND C. MEINEL, *Advanced persistent threats: Behind the scenes*, in 2016 Annual Conference on Information Science and Systems (CISS), 2016, pp. 181–186.

[137] I. VAKILINIA AND S. SENGUPTA, *Fair and private rewarding in a coalitional game of cybersecurity information sharing*, IET Information Security, 13 (2019), pp. 530–540.

[138] V. V. VAZIRANI, *Approximation Algorithms*, Springer, Berlin, Heidelberg, 2003.

[139] E. F. VILLARONGA, P. KIESEBERG, AND T. LI, *Humans forget, machines remember: Artificial intelligence and the right to be forgotten*, Computer Law & Security Review, 34 (2018), pp. 304–313.

[140] P. VOIGT AND A. VON DEM BUSSCHE, *The eu general data protection regulation (gdpr): A practical guide*, in GDPR, 2017.

[141] M. VOORNEVELD, *Best-response potential games*, Economics Letters, 66 (2000), pp. 289–295.

[142] Z. WAN, J.-H. CHO, M. ZHU, A. H. ANWAR, C. A. KAMHOUA, AND M. P. SINGH, *Foureye: Defensive deception against advanced persistent threats via hypergame theory*, IEEE Transactions on Network and Service Management, 19 (2022), pp. 112–129.

[143] J. WANG, S. GUO, X. XIE, AND H. QI, *Federated unlearning via class-discriminative pruning*, in Proceedings of the ACM Web Conference 2022, WWW '22, ACM, 2022, pp. 622–632.

[144] T. WANG, N. LI, AND S. JHA, *Locally differentially private frequent itemset mining*, in 2018 IEEE Symposium on Security and Privacy (SP), 2018, pp. 127–143.

[145] W. WANG, L. YING, AND J. ZHANG, *The value of privacy: Strategic data subjects, incentive mechanisms, and fundamental limits*, ACM Trans. Econ. Comput., 6 (2018).

[146] Y. WANG, J. CHEN, M. ZHAO, B. LI, H. SHI, AND H. YOU, *Maximum energy penetration rate of pv in distribution network under security constraints based on game theory*, IEEE Transactions on Power Systems, 38 (2023), pp. 3427–3439.

[147] K. WEI, J. LI, M. DING, C. MA, H. H. YANG, F. FAROKHI, S. JIN, T. Q. S. QUEK, AND H. VINCENT POOR, *Federated learning with differential privacy: Algorithms and performance analysis*, IEEE Transactions on Information Forensics and Security, 15 (2020), pp. 3454–3469.

[148] K. WEI, J. LI, C. MA, M. DING, C. CHEN, S. JIN, Z. HAN, AND H. V. POOR, *Low-latency federated learning over wireless channels with differential privacy*, IEEE Journal on Selected Areas in Communications, 40 (2022), pp. 290–307.

[149] C. WU, S. ZHU, AND P. MITRA, *Federated unlearning with knowledge distillation*, arXiv preprint arXiv:2201.09441, (2022).

[150] N. WU, F. FAROKHI, D. SMITH, AND M. A. KAAFAR, *The value of collaboration in convex machine learning with differential privacy*, in 2020 IEEE Symposium on Security and Privacy (SP), 2020, pp. 304–317.

[151] Y. WU, E. DOBRIBAN, AND S. DAVIDSON, *DeltaGrad: Rapid retraining of machine learning models*, in Proceedings of the 37th ICML, vol. 119 of PMLR, PMLR, 13–18 Jul 2020, pp. 10355–10366.

[152] D. XIAO, *Is privacy compatible with truthfulness?*, in Proceedings of the 4th Conference on Innovations in Theoretical Computer Science, ITCS'13, New York, NY, USA, 2013, Association for Computing Machinery, pp. 67–86.

[153] L. XIAO, D. XU, N. B. MANDAYAM, AND H. V. POOR, *Attacker-centric view of a detection game against advanced persistent threats*, IEEE Transactions on Mobile Computing, 17 (2018), pp. 2512–2523.

[154] Z. XU, S. RAY, P. SUBRAMANYAN, AND S. MALIK, *Malware detection using machine learning based analysis of virtual memory access patterns*, in Design, Auto. Test in EU Conf. Exhi.), 2017, 2017, pp. 169–174.

[155] L.-X. YANG, P. LI, X. YANG, AND Y. Y. TANG, *A risk management approach to defending against the advanced persistent threat*, IEEE Trans. on Dep. and Secu. Comput., 17 (2020), pp. 1163–1172.

[156] L.-X. YANG, P. LI, Y. ZHANG, X. YANG, Y. XIANG, AND W. ZHOU, *Effective repair strategy against advanced persistent threat: A differential game approach*, IEEE TIFS, 14 (2019), pp. 1713–1728.

[157] Q. YANG, Y. LIU, T. CHEN, AND Y. TONG, *Federated machine learning: Concept and applications*, ACM Trans. Intell. Syst. Technol., 10 (2019).

[158] D. YE, T. ZHU, S. SHEN, AND W. ZHOU, *A differentially private game theoretic approach for deceiving cyber adversaries*, IEEE Transactions on Information Forensics and Security, 16 (2021), pp. 569–584.

[159] D. YE, T. ZHU, W. ZHOU, AND P. S. YU, *Differentially private malicious agent avoidance in multiagent advising learning*, IEEE Transactions on Cybernetics, 50 (2020), pp. 4214–4227.

[160] H. YIN AND S. PAN, *Knowledge transfer for deep reinforcement learning with hierarchical experience replay*, AAAI, 31 (2017).

[161] X. YIN, Y. ZHU, AND J. HU, *A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions*, ACM Comput. Surv., 54 (2021).

[162] J. YU, B. ZHANG, Z. KUANG, D. LIN, AND J. FAN, *iprivacy: Image privacy protection by identifying sensitive objects via deep multi-task learning*, IEEE Transactions on Information Forensics and Security, 12 (2017), pp. 1005–1016.

[163] Y. ZHAN, P. LI, Z. QU, D. ZENG, AND S. GUO, *A learning-based incentive mechanism for federated learning*, IEEE Internet of Things Journal, 7 (2020), pp. 6360–6368.

[164] Y. ZHAN, J. ZHANG, Z. HONG, L. WU, P. LI, AND S. GUO, *A survey of incentive mechanism design for federated learning*, IEEE Transactions on Emerging Topics in Computing, 10 (2022), pp. 1035–1044.

[165] H. ZHANG, J. TAN, X. LIU, S. HUANG, H. HU, AND Y. ZHANG, *Cybersecurity threat assessment integrating qualitative differential and evolutionary games*, IEEE Transactions on Network and Service Management, 19 (2022), pp. 3425–3437.

[166] L. ZHANG, T. ZHU, P. XIONG, W. ZHOU, AND P. S. YU, *More than privacy: Adopting differential privacy in game-theoretic mechanism design*, ACM Comput. Surv., 54 (2021).

[167] R. ZHANG AND Q. ZHU, *A game-theoretic approach to design secure and resilient distributed support vector machines*, IEEE Transactions on Neural Networks and Learning Systems, 29 (2018), pp. 5512–5527.

[168] ——, *Flipin: A game-theoretic cyber insurance framework for incentive-compatible cyber risk management of internet of things*, IEEE Transactions on Information Forensics and Security, 15 (2020), pp. 2026–2041.

[169] L. ZHAO, Q. WANG, Q. ZOU, Y. ZHANG, AND Y. CHEN, *Privacy-preserving collaborative deep learning with unreliable participants*, IEEE Transactions on Information Forensics and Security, 15 (2020), pp. 1486–1500.

[170] F. ZHOU AND C. CAO, *Overcoming catastrophic forgetting in graph neural networks with experience replay*, Proceedings of the AAAI Conference on Artificial Intelligence, 35 (2021), pp. 4714–4722.

[171] L. ZHU, Z. LIU, AND S. HAN, *Deep leakage from gradients*, in NIPS, vol. 32, Curran Associates, Inc., 2019.

[172] T. ZHU, D. YE, Z. CHENG, W. ZHOU, AND P. S. YU, *Learning games for defending advanced persistent threats in cyber systems*, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 53 (2023), pp. 2410–2422.

[173] T. ZHU, D. YE, W. WANG, W. ZHOU, AND P. S. YU, *More than privacy: Applying differential privacy in key areas of artificial intelligence*, IEEE Transactions on Knowledge and Data Engineering, 34 (2022), pp. 2824–2843.