



Title	A Study on Security and Privacy for AI Supplied by Malicious Model Trainers
Author(s)	岩花, 一輝
Citation	大阪大学, 2026, 博士論文
Version Type	VoR
URL	https://doi.org/10.18910/105504
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

A Study on Security and Privacy for AI Supplied by Malicious Model Trainers

Submitted to
Graduate School of Information Science and Technology
The University of Osaka

January 2026

Kazuki IWAHANA

List of publication

Journals (Peer-reviewed)

- [J1] **Kazuki Iwahana**, Naoto Yanai, Atsuo Inomata, Toru Fujiwara, “Backdoor Attacks by Leveraging Latent Representation in Competitive Learning for Resistance to Removal”, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Volume 108, pp.254-266, 2025.
- [J2] Takayuki Miura, Masanobu Kii, Toshiki Shibahara, **Kazuki Iwahana**, Tetsuya Okuda, Atsunori Ichikawa, and Naoto Yanai. Setsubun, “Revisiting membership inference game for evaluating synthetic data generation”, Journal of Information Processing, Vol. 32, pp. 757-766, 2024.
- [J3] Hiromasa Kitai, Naoto Yanai, **Kazuki Iwahana**, Masataka Tatsumi, Jason Paul Cruz, “A Study on Quantized Parameters for Protection of a Model and Its Inference Input”, Journal of Information Processing, Vol.31, 667-678, 2023.
- [J4] **Kazuki Iwahana**, Naoto Yanai, Jason Paul Cruz, Toru Fujiwara, “SPGC: Integration of Secure Multiparty Computation and Differential Privacy for Gradient Computation on Collaborative Learning”, Journal of Information Processing, Vol.30, No. 3, pp.1-17, 2022.
- [J5] 矢内 直人, 岩花 一輝, 北井 宏昌, 四方 寿樹, 竹村 達也, クルーズ ジェイソン ポール, “機械学習のモデル保護の研究とその周辺動向”, 電子情報通信学会論文誌 (B), Vol. J104-B, No.10, pp.742-760, 2021.

- [J6] **Kazuki Iwahana**, Tatsuya Takemura, Ju Chien Cheng, Nami Ashizawa, Naoki Umeda, Kodai Sato, Ryota Kawakami, Rei Shimizu, Yuichiro Chinen, Naoto Yanai, “MADMAX: Browser-Based Malicious Domain Detection through Extreme Learning Machine”, IEEE Access, Vol.9, pp. 78293-78314, IEEE 2021.

International Conferences (Peer-reviewed)

- [IC1] Osamu Saisho, Takayuki Miura, **Kazuki Iwahana**, Masanobu Kii, Rina Okada, “Efficient Privacy-Preserving Data Annotation via Active PrivBayes Synthetic Data Generation”, PerCom Workshops 2025.
- [IC2] **Kazuki Iwahana**, Osamu Saisho, Takayuki Miura, Akira Ito, “Can Inputs’ Reconstruction Information Be Coded into Machine Learning Model’s Outputs?”, SECAI 2023 on ESORICS Workshops.
- [IC3] **Kazuki Iwahana**, Naoto Yanai, Toru Fujiwara, “Backdoor Attacks Leveraging Latent Representation in Competitive Learning”, SECAI 2023 on ESORICS Workshops.
- [IC4] Osamu Saisho, Keiichiro Kashiwagi, Sakiko Kawai, **Kazuki Iwahana**, Koki Mitani, “Sandbox AI: We Don’t Trust Each Other but Want to Create New Value Efficiently Through Collaboration Using Sensitive Data”, UbiComp/ISWC Adjunct 2023.
- [IC5] Hiromasa Kitai, Naoto Yanai, **Kazuki Iwahana**, Masataka Tatsumi, Jason Paul Cruz, “MOTUS: How Quantized Parameters Improve Protection of Model and Its Inference Input”, SecITC 2022.
- [IC6] **Kazuki Iwahana**, Naoto Yanai, Jason Paul Cruz, Toru Fujiwara, “SPGC: Integration of Secure Computation and Differential Privacy for Gradient Computation on Collaborative Learning”, Data Privacy Management 16th International Workshop (DPM 2021), LNCS, volume 13140, 2021-10.

Domestic Conferences (Not Peer-reviewed)

- [DC1] 岩花 一輝, 伊東 燦, 三浦 堯之, 山崎雄輔, 芝原俊樹, “埋め込み表現の分布シフトに基づくバックドアニューロンの特定および除去手法”, コンピュータセキュリティシンポジウム 2025 (CSS 2025), 3A4-5, 2025-10.
- [DC2] 税所 修, 三浦 堯之, 岩花 一輝, 紀伊 真昇, “モデルの情報効用を最大化するタスク誘導型差分プライバシー保護合成データ生成”, コンピュータセキュリティシンポジウム 2025 (CSS 2025), 1H1-4, 2025-10.
- [DC3] 中井 厚博, 岩花 一輝, 木下 洋輝, 芝原 俊樹, 内田 真人, “トークン選択を通じた人間介入型 LLM ジェイルブレイクの脅威分析”, コンピュータセキュリティシンポジウム 2025 (CSS 2025), 4E1-1, 2025-10.
- [DC4] 芝原 俊樹, 松林 勝, 岩花 一輝, 小山 卓麻, 大湊 健一郎, “進化戦略に基づく LLM 出力の自動修正による有害度低減”, コンピュータセキュリティシンポジウム 2025 (CSS 2025), 4E1-3, 2025-10.
- [DC5] 松林 勝, 岩花 一輝, 小山 卓麻, 芝原 俊樹, 遠田 耕司, 千田 成樹, 大湊 健一郎, “ソースコード著作権侵害の過検知抑制に向けた LLM の記憶強度に基づくありふれた表現の識別”, コンピュータセキュリティシンポジウム 2025 (CSS 2025), 4E1-4, 2025-10.
- [DC6] 岩花 一輝, 菊池 亮, “連結かつ譲渡不可能な選択開示手法”, 暗号と情報セキュリティシンポジウム 2025 (SCIS 2025), 3D3-1, 2025-01.
- [DC7] 武内 裕哉, 岩花 一輝, 三浦 堯之, 芝原 俊樹, 山下 恭佑, “インコンテキスト学習による Jailbreak に対するバックドアを利用した防御手法の一検討”, 暗号と情報セキュリティシンポジウム 2025 (SCIS 2025), 4G2-4, 2025-01.
- [DC8] 木下 洋輝, 芝原 俊樹, 三浦 堯之, 廣見 紗妃, 山田 真徳, 岩花 一輝, “大規模言語モデルに対する無害な応答を用いたバックドアによる Jailbreak 攻撃”, 暗号と情報セキュリティシンポジウム 2025 (SCIS 2025), 4G2-5, 2025-01.

- [DC9] 山下恭佑, 岩花 一輝, “主張可能集約署名の提案”, 暗号と情報セキュリティシンポジウム 2025 (SCIS 2025), 1C1-3, 2025-01.
- [DC10] 岩花一輝, 伊東燦, 山田真徳, 芝原俊樹, 山下智也, 三浦堯之, “モデル合成に対するプライバシーリスク評価”, コンピュータセキュリティシンポジウム 2024 (CSS 2024), 2F4-3, 2024-10.
- [DC11] 松本 知優, 三浦 堯之, 芝原 俊樹, 紀伊 真昇, 岩花 一輝, 税所 修, 岡村 真吾, “構造化状態空間モデルと拡散モデルを用いた差分プライバシーな系列データ合成”, コンピュータセキュリティシンポジウム 2024 (CSS 2024), 4F4-1, 2024-10.
- [DC12] 税所 修, 三浦 堯之, 岩花 一輝, 紀伊 真昇, 岡田 莉奈, “差分プライバシーとラベル付け効率を同時に考慮した能動合成データ生成”, コンピュータセキュリティシンポジウム 2024 (CSS 2024), 4F4-4, 2024-10.
- [DC13] 税所 修, 柏木 啓一郎, 岩花 一輝, 神谷 弘樹, “サンドボックス AI：秘匿すべき情報を出し合いながら人間の試行錯誤も行えるセキュアな AI 構築基盤”, 暗号と情報セキュリティシンポジウム 2024 (SCIS 2024), 3F1-4, 2024-01.
- [DC14] 岩花 一輝, 山下 恭祐, 矢内 直人, 税所 修, “機械学習モデルの更新の保証に向けたゼロ知識証明の適用”, コンピュータセキュリティシンポジウム 2023 (CSS 2023), 4F1-1, 2023-10.
- [DC15] 岩花 一輝, 税所 修, 三浦 堯之, 伊東 燦, “入力データの復元情報を AI の出力結果に埋め込めるか？”, 暗号と情報セキュリティシンポジウム 2023 (SCIS 2023), 1F3-2, 2023-01.
- [DC16] 三浦 堯之, 紀伊 真昇, 市川 敦謙, 岩花 一輝, 芝原 俊樹, 奥田 哲矢, 山本 充子, 矢内 直人, “合成データ生成の出力を評価するメンバーシップ推論攻撃フレームワーク”, コンピュータセキュリティシンポジウム 2021 (CSS 2021), 2C1-3, 2022-10.
- [DC17] 岩花 一輝, 三浦 堯之, 奥田 哲矢, 矢内 直人, “電子指紋は機械学習の二段階攻撃に使えるか？”, 暗号と情報セキュリティシンポジウム 2021 (SCIS 2022), 2D2-1, 2022-01.
- [DC18] 岩花 一輝, 矢内 直人, 藤原 融, “トリガとモデル両方の識別不可能性を持つバックドア攻撃”, コンピュータセキュリティシンポジウム 2021 (CSS 2021), 1C4-1, 2021-10.

- [DC19] 辰巳 将崇, 岩花 一輝, 矢内 直人, 穴戸 克成, 清水 俊也, 樋口 裕二, 森川 郁也, 矢島 純, “TEMPEST: 表形式データに対するモデル抽出攻撃”, コンピュータセキュリティシンポジウム 2021 (CSS 2021), 1C4-2, 2021-10.
- [DC20] 穴戸 克成, 清水 俊也, 樋口 裕二, 森川 郁也, 矢島 純, 辰巳 将崇, 岩花 一輝, 矢内 直人, “表形式データ学習モデルに対する外部公開統計情報を用いたモデル複製攻撃”, コンピュータセキュリティシンポジウム 2021 (CSS 2021), 1C4-3, 2021-10.
- [DC21] 岩花 一輝, 竹村 達也, 鄭 儒謙, 芦澤 奈美, 梅田 直希, 佐藤航大, 川上 遼太, 清水 嶺, 知念 祐一郎, 矢内直人 “MADMAX : Extreme Learning Machine を用いたブラウザベース悪性ドメイン検知アプリケーション” DICO2021, 2G-1, 2021-07.
- [DC22] 岩花 一輝, 矢内 直人, 藤原 融, “差分プライバシーと秘密計算の融合による秘匿性がデータ提供者の数に依存しない秘匿協調学習”, コンピュータセキュリティシンポジウム 2020 (CSS 2020), 1C4-2, 2020-10.

Preface

Recent advances in artificial intelligence (AI) have accelerated its deployment across society, with AI now widely used in domains such as medicine, transportation, and finance. The development and deployment of AI models form an AI supply chain involving multiple entities, including data providers, model trainers, operators, and model users. Within this supply chain, security and privacy risks have become increasingly evident, such as unintended dangerous model behaviors and unintended data leakage to third parties, posing fundamental challenges to the reliability and trustworthiness of AI systems.

This thesis aims to clarify security and privacy threats that arise from malicious model trainers in the AI supply chain, and to contribute to the establishment of a secure and privacy-preserving AI supply chain systems through the analysis of both attack techniques and corresponding defense mechanisms.

First, in Chapter 2, we propose Secure and Private Gradient Computation (SPGC), a defense mechanism against threats to model security and data privacy posed by malicious local model trainers (data providers) in distributed learning settings such as collaborative or federated learning. SPGC is a secure and privacy-preserving framework that integrates Secure Multiparty Computation (MPC) with Differential Privacy (DP). By concealing gradient information via MPC, SPGC reduces the amount of DP noise required, thereby preventing attacks while maintaining high model accuracy.

On the other hand, it remains unclear to what extent a malicious model trainer in a centralized learning setting can mount attacks that induce dangerous model behavior or compromise data privacy. We therefore investigate the feasibility of both types of attacks when the model trainer

behaves maliciously and discuss possible defense strategies.

In Chapter 3, we propose TALPA, a backdoor attack in which a malicious model trainer induces dangerous model behavior while increasing resistance to detection in both the input space and the latent space. Our results demonstrate that, once the model trainer becomes an adversary, it becomes possible to realize backdoor attacks that are significantly more difficult to defend against than conventional approaches. We further suggest that detecting abnormality across classes in the internal representations of the model may provide a promising direction for defenses against such attacks.

In Chapter 4, we propose a data reconstruction attack in which a malicious model trainer trains the model so that its outputs encode information sufficient to reconstruct user inputs. This attack enables the reconstruction of confidential inputs from model outputs even in confidential computing environments that rely on Trusted Execution Environments or Secure Multiparty Computation. We thereby demonstrate that, once the model trainer becomes an adversary, user privacy can be compromised even when input data are kept secret. As a potential mitigation, we show that transforming inputs to increase their diversity can reduce the effectiveness of this attack.

In summary, by analyzing attack and defense mechanisms under malicious model trainers across different learning settings, this thesis establishes a principled foundation for constructing secure and privacy-preserving AI supply chain systems. We believe that the findings of this thesis will support the advancement of trustworthy AI supply chain systems and enhance the understanding of the complex interplay among security, privacy, and the structure of the AI supply chain.

Acknowledgments

This thesis could not have been accomplished without the assistance of many people in various ways, and I would like to acknowledge all of them.

I would like to express my sincere and profound gratitude to Prof. Atsuo Inomata for welcoming me as a member of his laboratory midway through my doctoral studies. I am deeply thankful for his invaluable comments on my dissertation and for his constructive and insightful advice throughout the course of this research. Without his significant contributions and generous support, this thesis could not have been completed. I am truly grateful for his guidance and encouragement. I would like to express my sincere gratitude to Prof. Hirozumi Yamaguchi, Prof. Hideyuki Shimonishi, Associate Prof. Shunsuke Saruwatari, and Associate Prof. Yuki Koizumi for their precise and insightful comments on my research. Their valuable feedback greatly improved the quality of this dissertation and played an essential role in its completion. I would also like to express my heartfelt appreciation to Dr. Naoto Yanai, who has guided me since my undergraduate and master's studies. He taught me not only what research is and how rewarding it can be, but also shared invaluable lessons on how to think and act as a researcher. His guidance has profoundly influenced my approach to research, and I am deeply thankful for his mentorship. I am sincerely grateful to Prof. Toru Fujiwara for his meaningful comments and constructive advice since my undergraduate and master's studies. His perspectives and suggestions consistently highlighted important aspects of research that I had overlooked, often providing moments of realization that significantly shaped my thinking. I truly appreciate his continued support and guidance. I would like to sincerely thank Shingo Okamura for his valuable comments and constructive suggestions on my research, which significantly contributed to improving the quality of this work. I am also grateful to Jason Paul

Cruz for his insightful comments and constructive feedback on my research. In particular, I greatly benefited from his guidance and support in academic English writing, which was extremely helpful throughout the preparation of this thesis. I would like to thank Assistant Prof. Kyosuke Yamashita for his friendly and supportive attitude since the beginning of my doctoral studies. I am grateful for his valuable research advice, particularly from theoretical perspectives, and for the fruitful discussions that led to new research ideas. His guidance greatly enriched my doctoral experience. I would also like to express my sincere gratitude to the secretaries of the former Fujiwara Laboratory (currently the Isobe Laboratory) and the Inomata Laboratory. Their continuous support in administrative procedures, communication, and daily academic life made my undergraduate, master's, and doctoral studies run smoothly. I am deeply appreciative of their kind assistance throughout my academic journey.

I am grateful to my colleagues at NTT Social Informatics Laboratories for their support and encouragement during my doctoral studies. In particular, I would like to thank Mr. Osamu Saisho for teaching me how to approach research as an industrial researcher and for providing valuable insights into conducting research in a corporate environment. I am also grateful for his constructive feedback and meaningful comments on this work. I sincerely thank Dr. Takayuki Miura for engaging in numerous research discussions with me since my student days and for his continuous support and valuable comments on my research. He has also been a reliable senior colleague who laid the foundation of the AI security research community within the company, and I am deeply grateful for his guidance and support. I would like to express my deep appreciation to Dr. Akira Ito for teaching me how to think as a professional researcher and how to write high-quality academic papers. I firmly believe that the foundation of my research skills was established through his guidance. I am also grateful for his constructive feedback and valuable comments on this research. I am deeply indebted to Dr. Toshiki Shibahara for his invaluable guidance on all aspects of my research, ranging from technical details to overall research direction. I am also sincerely grateful for his thoughtful support within the company, which greatly contributed to the completion of this thesis. I would also like to thank Koki Mitani, Tomohiro Inoue, Kazuyuki Takaya, Shigeru Fujimura, and Sadahiro Ishizaki for their advice on pursuing a doctoral degree while working in industry and for their substantial support as my supervisors. I would like to express my sincere appreciation to all co-authors

who collaborated with me on my research. I am grateful for the many fruitful discussions and for their generous support, which significantly contributed to the development of this work.

I would like to express my heartfelt gratitude to my parents, grandparents and friends for their constant support, understanding, and encouragement throughout my doctoral journey. Their patience and trust have been a great source of motivation, especially during challenging times. Finally, I would like to express my deepest and most special gratitude to my wife. Thank you for always being by my side and for providing unwavering emotional support and a stable living environment throughout my doctoral journey. I truly believe that I could not have overcome the challenges of this journey without your constant encouragement, understanding, and patience. I am profoundly grateful for everything you have done for me.

Contents

List of publication	i
Preface	vii
Acknowledgments	ix
1 Introduction	1
1.1 Background	1
1.2 Goal and Contribution of Thesis	2
1.2.1 Secure and Private Collaborative Learning Framework under Malicious Local Model Trainers	3
1.2.2 Advanced Backdoor Attack under a Malicious Model Trainer	5
1.2.3 Advanced Data Reconstruction Attack under a Malicious Model Trainer . .	6
2 Secure and Private Collaborative Learning Framework under Malicious Local Model Trainers	9
2.1 Introduction	9
2.1.1 Backgrounds	9
2.1.2 Contribution	10
2.2 Preliminaries	11
2.2.1 Deep Learning and Collaborative Learning	12
2.2.2 Differential Privacy	13

2.2.3	Secure Multiparty Computation	14
2.3	Problem Description	14
2.3.1	Secure and Privacy-Preserving Collaborative Learning	14
2.3.2	Technical Difficulty	15
2.4	Design of SPGC	16
2.4.1	Overview	16
2.4.2	Lack of Correctness on CGC	18
2.4.3	Construction	20
2.4.4	Participant-Side Process	21
2.4.5	Server-Side Process	21
2.4.6	Correctness	22
2.4.7	Privacy Analysis	24
2.5	Experiments	26
2.5.1	Implementation	27
2.5.2	Experimental Setup	27
2.5.3	Results	31
2.6	Discussion	34
2.6.1	Accuracy	35
2.6.2	Training Time	37
2.6.3	Number of participants	40
2.6.4	Variation of Noise Generation	41
2.6.5	Effect by Clipping Parameter	41
2.6.6	Limitation	42
2.7	Related Works	42
2.7.1	Integration of Secure Multiparty Computation and Differential Privacy	42
2.7.2	Other Privacy-Preserving Machine Learning	43
2.7.3	Differential Privacy	44
2.7.4	Secure Multiparty Computation	45
2.8	Conclusion	45

3	Advanced Backdoor Attack under a Malicious Model Trainer	47
3.1	Introduction	47
3.2	Related Works	49
3.2.1	Backdoor Removal for Countermeasures	49
3.2.2	Backdoor Attacks with Resistance to Backdoor Removal	50
3.3	Problem Setting	50
3.3.1	Backdoor Attacks	51
3.3.2	Threat Model	51
3.4	TALPA	52
3.4.1	Overview	52
3.4.2	Objective Functions	53
3.5	Experiments	55
3.5.1	Setting	55
3.5.2	Results of Accuracy and Attack Success Rate	58
3.5.3	Results of Backdoor Removal	58
3.6	Discussion	61
3.6.1	Impacts of Hyperparameters	61
3.6.2	Impacts of Norms	62
3.6.3	Extension to Other Layers	63
3.6.4	Suspicion of Poison Inputs	64
3.6.5	Resistance against Existing Backdoor Defense	65
3.6.6	Limitation	66
3.6.7	Potential Countermeasures	66
3.7	Conclusion	67
4	Advanced Data Reconstruction Attack under a Malicious Model Trainer	69
4.1	Introduction	69
4.2	Related Work	71
4.3	Problem Setting	73

4.3.1	Confidential Inference in MLaaS	73
4.3.2	Attack Scenarios	73
4.3.3	Adversary’s Capacity and Goal	74
4.4	Attack Method	75
4.4.1	Training Phase	75
4.4.2	Attack Phase	75
4.5	Evaluation	76
4.5.1	Experimental Setup	77
4.5.2	Results	77
4.6	Discussion	79
4.6.1	Effectiveness of Our Attack	79
4.6.2	Similarity with Reconstructed Images	80
4.6.3	Difficulty of data reconstruction	81
4.6.4	Relation to Autoencoders	82
4.6.5	Potential Countermeasures	83
4.7	Conclusion	84
5	Conclusion	89
	Bibliography	93

Chapter 1

Introduction

1.1 Background

Recent advances in artificial intelligence (AI) and machine learning (ML) have been remarkable, and their applications now span a wide range of domains, including image recognition, natural language processing, medical diagnosis, traffic management, and financial trading. These technologies have become integral components of social and industrial infrastructures. At the same time, as AI systems proliferate and become deeply embedded in critical decision-making processes, concerns are rapidly increasing regarding “model security”, which preserves AI models from exhibiting unintended or dangerous behaviors, and “data privacy”, which involves preventing the leakage of sensitive information contained in training or inference data. The more AI systems automate decision-making, the more essential it becomes to ensure both the security of model behavior and the privacy of the data used within them.

The development and deployment of AI technologies do not rely on a single algorithmic step but instead consist of a chain of interconnected processes, including data collection, model training, deployment, and operation. In practice, AI models are often developed and trained by one party and then supplied to others as pretrained models, APIs, or embedded components. In this thesis, we refer to this socio-technical workflow, where multiple entities contribute to producing, delivering, and using AI, as the *AI supply chain systems*. In the AI supply chain systems, there are multiple

1.2 Goal and Contribution of Thesis

entities, such as data providers who supply training data, model trainers who train models, model operators who manage or operate them, and model users who query their own data to them and obtain model outputs.

While the involvement of multiple entities contributes to flexibility and performance improvements in AI systems, each stage of the AI supply chain systems is associated with distinct types of security and privacy risks. Among these, two particularly critical threats have been extensively studied: backdoor attacks, which compromise model security, and data reconstruction attacks, which compromise data privacy. Backdoor attacks [1–3] intentionally embed a trigger pattern into the model during training, so that the model behaves normally on benign inputs but returns adversary-controlled outputs when the trigger is present. A representative approach to backdoor attacks involves an adversary acting as a data provider who inserts trigger patterns into training samples and manipulates class labels [1]. On the other hand, data reconstruction attacks [4–9] attempt to recover sensitive information about the data used to train or query the model. For example, an adversary acting as a model user may issue queries to the model and infer properties of the training data solely from its outputs [4]. As these examples illustrate, the complexity of the AI supply chain systems creates multiple potential attack surfaces, and different entities such as data providers, model trainers, and model users may each become sources of vulnerability.

This thesis focuses on security and privacy risks arising from such malicious model trainers. The model trainer itself is likely to be an external or untrusted party and to act maliciously, so it gains privileged control over the training pipeline including training data, training algorithm, and architectures, which may enable stronger attacks than those assumed in other threat scenarios. Therefore, it is important to identify the threats posed by malicious model trainers within the AI supply chain systems and to develop corresponding defense mechanisms.

1.2 Goal and Contribution of Thesis

The goal of this thesis is to investigate security and privacy threats in the AI supply chain systems, with a particular focus on AI supplied by potentially malicious model trainers, and to develop principled defenses toward secure and privacy-preserving AI supply chain systems.

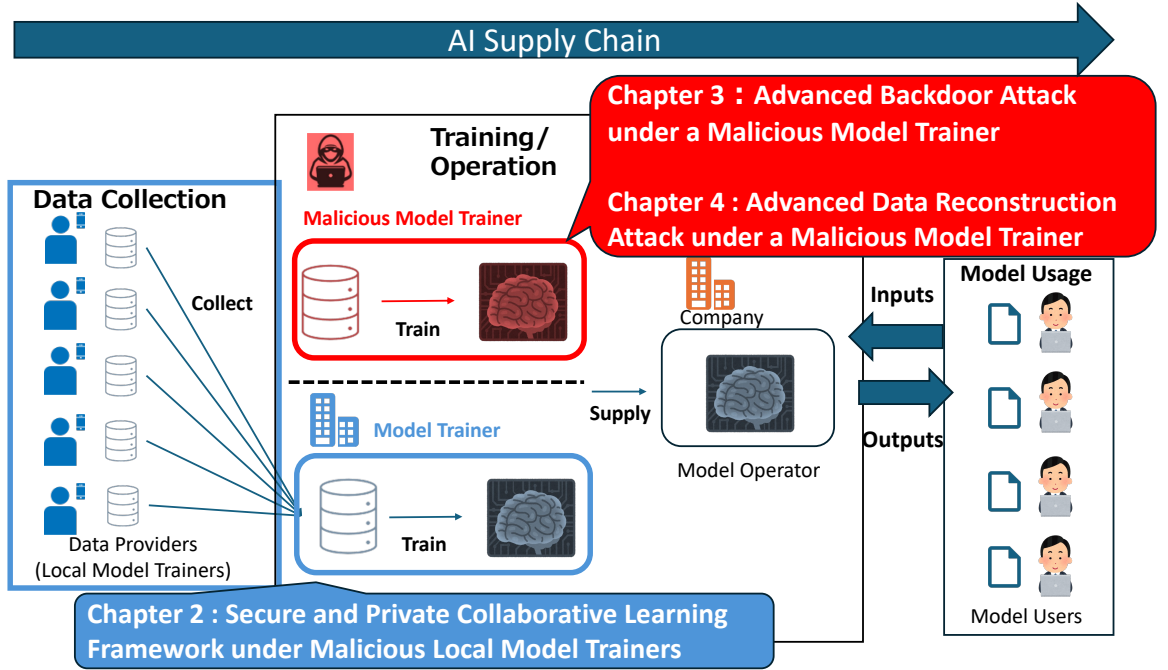


Figure 1.1: Overview of this thesis.

The concrete contributions of this thesis are summarized in Figure 1.1 and as follows:

- We establish defense mechanisms against backdoor attacks and data reconstruction attacks in collaborative learning, where multiple local model trainers jointly construct a single model and may behave maliciously.
- We present a novel backdoor attack that becomes feasible when the model trainer behaves maliciously, and further discuss possible defense strategies.
- We demonstrate a new form of data reconstruction attack that can be executed by a malicious model trainer, and also discuss possible defense strategies.

1.2.1 Secure and Private Collaborative Learning Framework under Malicious Local Model Trainers

Training AI models by collecting all data into a centralized learning setting is often costly or impractical due to data volume, privacy concerns, and organizational constraints. As a result, collaborative

1.2 Goal and Contribution of Thesis

or federated learning [10, 11] has attracted significant attention as a flexible training paradigm, in which data providers locally train models on their own data and collaboratively construct a single global model. Chapter 2 refers to such data providers as local model trainers. In such settings, each local model trainer may behave maliciously, posing realistic threats such as backdoor and data reconstruction attacks. For example, they may inject malicious updates associated with backdoors, thereby compromising the global model [12–14], or reconstruct other participants’ data from collected gradient updates [15–17]. Therefore, it is essential to develop defense mechanisms that allow honest model trainers to intervene in the training process to construct models secure against backdoor attacks, and to ensure that model outputs or gradients remain private against data reconstruction attacks.

A widely studied defense mechanism against backdoor attacks and data reconstruction attacks is differentially private stochastic gradient descent (DP-SGD) [18]. Differential Privacy (DP) [19] limits the influence of any individual data point on the final model by introducing calibrated random noise into gradient updates. This mechanism provides two important forms of defense: it mitigates the effect of malicious gradient updates corresponding to trigger patterns in backdoor attacks [20, 21], and it obscures data-specific signatures in gradients, thereby reducing the feasibility of data reconstruction attacks [22, 23]. In this sense, DP offers a unified theoretical framework for addressing both security and privacy threats.

However, in collaborative learning settings, DP alone suffers from a degradation in its privacy guarantee as the number of data providers increases. To address this limitation, this thesis proposes the Secure and Private Gradient Computation (SPGC) framework, which integrates DP with Secure Multiparty Computation (MPC), which computes a function on inputs given by participants without revealing the inputs themselves. SPGC securely aggregates gradients via MPC while applying DP noise to the aggregated gradients in MPC. This process achieves higher model accuracy than a local differential privacy (LDP) approach in which each participant independently adds DP noise [24] while keeping the same privacy level as the existing LDP approach. Our experiments conducted on three datasets confirm that SPGC maintains significantly higher accuracy while providing strong security and privacy guarantees.

SPGC thus serves as a foundational framework for ensuring security and privacy in distributed

learning settings such as collaborative or federated learning environments where local model trainers may act maliciously.

1.2.2 Advanced Backdoor Attack under a Malicious Model Trainer

In Chapter 2, we presented a defense method with DP and MPC against backdoor attacks in distributed learning environments, where adversarial local model trainers can access and manipulate only a part of the overall training process through their local updates. In contrast, the threats that arise in a fundamentally different and more powerful setting, where a single model trainer has complete access to and control over the entire training process, remain insufficiently understood.

Therefore, in Chapter 3, we investigate backdoor threats under this stronger adversarial scenario, in which the malicious model trainer operates within a centralized learning setting and can arbitrarily manipulate the full training pipeline. This threat scenario is not merely hypothetical. In modern AI systems, pretrained models are often obtained from public model repositories and model hubs, such as Hugging Face ¹, where model operators routinely deploy models trained by third-party model trainers. As a result, the scenario in which AI systems rely on models supplied by potentially malicious model trainers has become both realistic and increasingly common in practice.

Traditional backdoor attacks [1, 25, 26] largely assume that the adversary is a data provider who manipulates training data by inserting trigger patterns. Such attacks are limited in their expressive power and are vulnerable to common defense strategies, including pruning [27] and transfer learning [28]. Their reliance on input-space manipulation also makes it difficult to embed highly persistent malicious behavior within the model.

In contrast, a malicious model trainer has privileged access to the entire training pipeline, including model architecture, optimization procedures, loss functions, and latent representations. This privileged position allows the adversary to realize sophisticated and persistent backdoor behaviors that are infeasible for traditional adversaries. Under such a stronger adversary, revealing potential threats of backdoor attacks enables us to proactively develop corresponding defense mechanisms. In this thesis, we propose a novel backdoor attack that directly manipulates latent representations

¹<https://huggingface.co/>

1.2 Goal and Contribution of Thesis

within the model. The key idea is to enforce the latent representations of trigger-embedded inputs to align closely with those of a target class, thereby embedding backdoor behavior into the deeper structure of the model. By optimizing trigger generation and latent manipulation jointly, the proposed method achieves both high stealthiness and high attack success while preserving clean accuracy.

Moreover, because the backdoor is embedded at the level of latent representations rather than input patterns, the attack demonstrates strong resilience against pruning, transfer learning, and other common transformations. The proposed method demonstrates the potential threats posed by a malicious model trainer, underscoring the importance of including the model trainer in security threat models and discussing corresponding defense strategies.

1.2.3 Advanced Data Reconstruction Attack under a Malicious Model Trainer

In Chapter 2, we presented a defense method with DP and MPC against data reconstruction attacks in settings where adversaries such as local model trainers can observe or influence only limited aspects of the learning process, for example through model outputs or local updates. In contrast, the threats that arise in a more powerful setting, where a single model trainer has complete access to and control over the entire training process, remain insufficiently understood.

Therefore, in Chapter 4, similarly to Chapter 3, we investigate new data reconstruction threats under the malicious model trainer scenario, with a particular focus on whether concealed user inputs can be reconstructed from model outputs. In recent years, protecting input data privacy using confidential computing technologies, such as MPC and trusted execution environments (TEEs), has become increasingly important and is already adopted in real-world systems, including cloud platforms such as Microsoft Azure ². In such environments, although user inputs are protected during computation, model trainers may still obtain access to the model outputs for purposes such as anomaly detection or service quality improvement. This makes it realistic to consider scenarios in which a malicious model trainer attempts to reconstruct concealed input data solely from output observations, even under confidential computing settings.

²<https://azure.microsoft.com/ja-jp/solutions/confidential-compute>

Prior works [5–9, 29] on data reconstruction primarily assume that model users attempt to infer training data from observed outputs, placing strict limitations on the adversary’s power. However, if the model trainer itself is malicious, it can deliberately encode input-dependent information into the model’s output during training, turning the model output into a covert communication channel.

In Chapter 4, we propose a data reconstruction attack in which the model trainer modifies the training process so that the output distribution implicitly contains enough information to reconstruct inputs for model users. Under such a stronger adversary, identifying the potential threats of data reconstruction attacks allows us to proactively develop appropriate defense mechanisms.

Our analysis demonstrates that the difficulty of reconstruction strongly depends on the intrinsic diversity of the dataset: datasets with lower class diversity tend to preserve more input-specific information in the model output, making reconstruction easier. This finding suggests that the structural properties of data directly affect privacy risks within the AI supply chain systems.

Chapter 2

Secure and Private Collaborative Learning Framework under Malicious Local Model Trainers

To begin, this chapter presents secure and privacy-preserving learning methods under the threat scenario in collaborative learning where local model trainers may act as adversaries.

2.1 Introduction

2.1.1 Backgrounds

Collaborative learning also known as federated learning [10, 11] is a kind of machine learning where multiple participants (local model trainers) pool all training data, and a model is trained on this communal pool. In general, each participant trains a local model with its data and then periodically exchanges and updates model parameters with other participants. Therefore, security and privacy risks are an essential issues in collaborative learning because training data are collected and pooled from many sources. In security risks, a participant injects malicious updates associated with backdoors, thereby compromising the global model [12–14]. In privacy risks, inference attacks on training data privacy that reveal information about the training data [30, 31] have been found in

2.1 Introduction

the past years.

A typical approach to defense them in collaborative learning is to achieve *differential privacy* [32], that is, informally, each participant gives data noise in local [24, 33] and then updates a model with the data from all participants. However, such an approach often generates significant amounts of noise that degrade the accuracy of its resultant model. In contrast, the privacy guarantee weakens when the noise generated in local is slight because the training data can be inferred from model parameters [30, 31].

A potential solution to the privacy issues mentioned above is the integration [34] of differential privacy and *secure multiparty computation*, which computes a function on inputs given by participants without revealing the inputs themselves. In the integration approach, model parameters are protected by the use of secure multiparty computation before they are exchanged among participants, and small amounts of noise for differential privacy are generated and added when the model is updated [35, 36]. Intuitively, the integration approach enables participants to adjust the total amount of noise they generate and thus supports both the privacy guarantee and the inference accuracy.

However, the privacy guarantee in existing integration protocols [35, 36] on collaborative learning weakens when the number of participants increases. In collaborative learning, many participants are expected to join the network, given that the primary motivation for collaborative learning is to collect data from many sources. Therefore, a protocol that offers a privacy guarantee independent of the number of participants is desirable. This chapter aims to answer the following fundamental question: *Can a new integrated protocol of differential privacy and secure multiparty computation for collaborative learning which maintains a strong privacy guarantee and accuracy without depending on the number of participants be constructed?*

2.1.2 Contribution

In this chapter, we present *SPGC*, pronounced as “speak” and which is an abbreviation for Secure and Private Gradient Computation. SPGC is an integrated protocol of differential privacy and secure multiparty computation for collaborative learning that provides a privacy guarantee independent of

the number of participants. As another contribution, we conducted extensive experiments on SPGC with academic and medical diagnosis datasets to evaluate its performance.

The construction of SPGC is significantly different from existing works [35,36]. Existing works adopt local differential privacy [24] (LDP) in which each participant generates noise in local and then *removes* the noise via secure multiparty computation. However, an LDP-based construction will weaken the privacy guarantee even by the use of *any* secure multiparty computation. In contrast, the main idea behind SPGC is the *creation of noise within secure multiparty computation*.

The idea mentioned above was inspired by the collaborative gradient computation (CGC) protocol of Chase et al. [37], in which the noise is generated by using secure multiparty computation. However, we note that CGC is incomplete, and that SPGC is quite an improvement over CGC. In particular, we found a negative case in the training method of CGC where the positive and negative gradients become indistinguishable from each other. SPGC can avoid such an issue as well. Moreover, Chase et al. did not discuss experimental evaluation in detail, more specifically they only provided the inference accuracy on the MNIST dataset and they did not provide training time. Detailed experimental evaluation is crucial for an integration protocol for differential privacy and secure multiparty computation machine learning. How noise on the differential privacy affects evaluation on each dataset is essential for accuracy, and communication overhead on the secure multiparty computation is significant for training time. We formally show that SPGC overcomes the negative case described, and then present the experimental evaluation in detail including evaluations on medical diagnosis datasets in addition to the MNIST dataset and those training times. (See Section 2.4 and Section 2.5 for more detail.) We then show that the training time increases with respect to the noise generation through the experimental results. We also demonstrate that the privacy guarantee is independent of the number of participants compared to an LDP-based construction as a naive approach. (See Section 2.6 for more details.)

2.2 Preliminaries

In this section, we provide backgrounds on deep learning and collaborative learning. We then describe differential privacy and secure multiparty computation as building blocks of SPGC.

2.2.1 Deep Learning and Collaborative Learning

Deep Learning. Deep learning is composed of a *training* phase to find optimal weight parameters on a model to be trained and an *inference* phase to solve a task via inference on the trained model. The goal of the training phase is to find weight parameters that yield an acceptably small loss on a loss function between an output of the model and its training data.

Let d_{in} denote the input dimension, $\mathbf{w}$ the model parameters, and N_C the number of classes. Consider a model (neural network) $f : \mathbb{R}^{d_{\text{in}}} \rightarrow [0, 1]^{N_C}$ that outputs class probabilities for an input sample $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ and is parameterized by $\mathbf{w}$. Let $\ell(\cdot, \cdot)$ denote the loss for a single sample. For simplicity, $L(\mathbf{w}_t, \mathbf{x}_i) = \ell(f(\mathbf{x}_i; \mathbf{w}_t), y_i)$, where y_i is the class of $\mathbf{x}_i$. The stochastic gradient descent (SGD) algorithm is often utilized to minimize L . The SGD algorithm computes a average gradient $\mathbf{g} = \nabla_{\mathbf{w}_t} \frac{1}{u} \sum_{i=1}^u L(\mathbf{w}_t, \mathbf{x}_i)$ and updates the current weight parameter $\mathbf{w}_t$ to $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}$, where η is a learning rate. In general, the update process of weight parameters is done for each data group called a *batch*. One of the well-known implementations to instantiate the SGD algorithm is `AdamOptimizer()`, which is also utilized in this work.

Collaborative Learning. A typical deep learning algorithm considers a centralized setting whereby a central server gathers data for training. In contrast, a deep learning algorithm to train a model distributively by multiple participants is called *collaborative learning* as known as federated learning [10, 11].

The motivation for collaborative learning is to train a model so that even if training data is distributed to multiple participants, each participant trains a local model on its data and then exchanges model parameters with other participants. The main advantage of collaborative learning is to collect training data from multiple participants. In other words, in a situation where training data is distributed to each participant, a participant can share the training with the other participants. To do this, for each step in training a model, each participant sends model parameters, such as gradients or weights, to n central servers. Hereafter, we focus on gradients as model parameters.

For a set $P = \{P_1, P_2, \dots, P_k\}$ of participants and a set $X = \{X_{P_1}, X_{P_2}, \dots, X_{P_k}\}$ of data, a participant P_j for any $j \in [1, k]$ computes a gradient $\mathbf{g}^{(X_{P_j})}$ for data X_{P_j} in local and then sends it to the central servers. Roughly speaking, collaborative learning updates a model parameter

$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\frac{1}{k} \sum_{i=1}^k \mathbf{g}^{(X_{P_i})})$ using gradients sent from all the participants on n central servers $H = \{H_1, H_2, \dots, H_n\}$. Afterwards, the servers return $\mathbf{w}_{t+1}$ to all the participant. After receiving $\mathbf{w}_{t+1}$ from the central servers, the participants utilize $\mathbf{w}_{t+1}$ in the next training. At the end of the training, each participant owns the same model parameter as the other participants and the central servers.

After the training phase of the collaborative learning is finished, the final model parameter $\mathbf{w}_T$ often becomes publicly available for inference by a client. In doing so, a client can give the model parameter $\mathbf{w}_T$ a query on inference with any data $\mathbf{x}_{\text{client}}$ and then obtain the inference results $f(\mathbf{w}_T; \mathbf{x}_{\text{client}})$.

2.2.2 Differential Privacy

Differential privacy [32] is a mathematical notion that guarantees privacy theoretically. Recall the definition below.

Definition 1. A randomized mechanism $M : D \rightarrow R$ with domain D and range R satisfies (ϵ, δ) -differential privacy if, for any two adjacent inputs $x, x' \in D$ and for any subset of outputs $V \subseteq R$, the following equation holds:

$$\Pr(M(x) \in V) \leq \exp(\epsilon) \Pr(M(x') \in V) + \delta.$$

Note that ϵ -differential privacy is identical to a special case for $\delta = 0$ in the definition above. When a deterministic real-valued function is defined as $O : D \rightarrow R$, a typical way to satisfy the differential privacy for O is to perturb the output of a function O by adding noise. More precisely, a mechanism M is instantiated via additive noise calibrated to O 's sensitivity S_O , which is defined as the maximum of the absolute distance $|O(x) - O(x')|$ where x and x' are adjacent inputs. The Gaussian mechanism or the Laplace mechanism is often utilized for generating noise. Hereafter, we focus on the Gaussian mechanism which is defined by $M(x) = O(x) + \mathcal{N}(0, (s_O \sigma)^2)$, where $\mathcal{N}(0, (s_O \sigma)^2)$ is the Gaussian distribution with mean 0 and standard deviation $s_O \sigma$. According to Theorem 3.22 in [38], the Gaussian mechanism for function O of the sensitivity s_O satisfies (ϵ, δ) -differential privacy for $\delta \leq \frac{4}{5} \exp(-(\sigma \epsilon)^2/2)$ and $\epsilon < 1$.

2.3 Problem Description

2.2.3 Secure Multiparty Computation

Secure multiparty computation is a cryptographic tool that is commonly used for evaluating a function between multiple participants without leaking any information beyond what is revealed by the output of the computation. We describe garbled circuits [39] and secret sharing [40] as building blocks of the secure multiparty computation below.

Garbled Circuit. A garbled circuit is a secure multiparty computation protocol used for evaluating any function as well as preserving the privacy of inputs. Garbled circuits are often utilized in a two-party setting. In this protocol, a function is presented as Boolean circuits to be encrypted. TinyGarble [41] is a publicly available library of garbled circuits.

Secret Sharing. Secret sharing is a cryptographic tool used for encoding data into multiple shares such that each share reveals nothing about the original data. The original data itself can then be recovered when shares more than a threshold designated in advance are gathered.

We describe how smod operations defined by Chase et al. [37] can be computed with secret sharing below. The smod operation is defined as $x \text{ smod } C = ((x+C) \bmod 2C) - C$, where x and C are any integer. For instance, for any integer $x \in [-C, C)$, shares of x are generated by $\langle x \rangle_1 = (x + \langle x \rangle_2) \text{ smod } C$, where $\langle x \rangle_2$ is uniformly distributed in $[-C, C)$. The resulting share $\langle x \rangle_1, \langle x \rangle_2$ then reveals nothing about x . Then, x is recovered by computing $x = \langle x \rangle_1 - \langle x \rangle_2 \text{ smod } C$.

2.3 Problem Description

In this section, we describe secure and privacy-preserving collaborative learning as the primary problem setting and its technical difficulty to be solved in this chapter.

2.3.1 Secure and Privacy-Preserving Collaborative Learning

Privacy-preserving collaborative learning is collaborative learning where the privacy of training data provided by k participants $P = \{P_1, \dots, P_k\}$ is preserved against n servers in the training phase and a client in the inference phase. For a privacy-preserving mechanism M and a model parameter w_t trained by collaborative learning for the entire training datasets $X = \{X_{P_1}, X_{P_2}, \dots, X_{P_k}\}$, all

participants try to compute $M(\mathbf{w}_t)$ and a client can utilize $M(\mathbf{w}_t)$ for an inference on any chosen input $\mathbf{x}_{\text{client}}$, that is, $f(M(\mathbf{w}_t); \mathbf{x}_{\text{client}})$ is computed.

In this chapter, the requirements for achieving training data privacy against an adversary are described below. These requirements are the same as in HybridAlpha [36]. In the following requirements, we assume that a participant P_1 and a server H_1 are honest for the sake of convenience.

Privacy of computation: An adversary can collude with $n - 1$ servers and $k - 1$ participants in the training phase, and the servers and participants follow a protocol under the honest-but-curious setting. Then, any information except for $M(\mathbf{w}_t)$ concerning training data X_{P_1} provided by an honest participant P_1 is not revealed to the adversary.

Privacy of output: An adversary can collude with a client who follows a protocol under the honest-but-curious setting in the inference phase. Then, no information with respect to training data X_{P_1} provided by P_1 is revealed to the adversary except for $f(M(\mathbf{w}_t); \mathbf{x}_{\text{client}})$ for any chosen $\mathbf{x}_{\text{client}}$.

As discussed in Section 2.1, attacks that infer training data from gradients during training have been proposed [30, 31]. To protect training data from such attacks, the privacy to protect gradients in the training phase, or more specifically the privacy of computation, is necessary. In addition, even when the privacy of computation is achieved, some attacks [4, 42] can reveal the original training data from inference results. This means that privacy in the inference phase, or in other words the privacy of output, should be guaranteed to protect the training data. Therefore, by achieving both the privacy of computation and the privacy of output, a collaborative learning algorithm that achieves the privacy guarantee for training data in the training and inference phases can be realized.

2.3.2 Technical Difficulty

The privacy guarantee of existing integration protocols [35, 36] of secure multiparty computation and differential privacy for collaborative learning weakens as the number of participants increases. These protocols allow each participant to perturb model parameters locally by adding noise to these parameters. These then require a central server to compute the mean of the perturbed parameters via secure multiparty computation. Then, the amount of noise in the existing protocols is often

2.4 Design of SPGC

excessively reduced instead of improving accuracy. Consequently, the privacy guarantee becomes weaker in exchange for improved accuracy. However, the privacy must be guaranteed to prevent inference attacks [4, 42] as described in the previous section.

The aforementioned problem on existing protocols [35, 36] is difficult to solve. Local differential privacy is used to guarantee privacy in collaborative learning [24] in general. Indeed, the existing protocols [35, 36] are extensions of local differential privacy. According to their experimental results [35, 36], accuracy of the local differential privacy decreases significantly in comparison with other settings, for example the non-privacy setting and an integration protocol with secure multiparty computation. More specifically, the existing protocols have utilized secure multiparty computation to remove the noise of local differential privacy for maintaining accuracy. In other words, if local differential privacy is adopted, removing noise by using secure multiparty computation is necessary to maintain accuracy because each user individually generates noise. Besides, when an adversary colludes with participants, the adversary may request the participants to reduce noise in order to infer target participant P_1 's training data X_{P_1} . Consequently, an approach based on removing the noise will blatantly reveal X_{P_1} .

Thus, besides the existing works [35, 36] described above, any protocol based on local differential privacy will sacrifice either the privacy guarantee or the accuracy *even if it uses secure multiparty computation*.

2.4 Design of SPGC

In this section, we present SPGC, a privacy-preserving collaborative learning protocol based on the integration of differential privacy and secure multiparty computation. We first describe an overview of SPGC, including the main idea for overcoming the problems described in the previous section and then show its construction and privacy analysis.

2.4.1 Overview

The key concept of SPGC is to perturb gradients by *creating noise for gradients within secure multiparty computation on servers* in contrast to the existing works [35, 36] which *remove* the noise

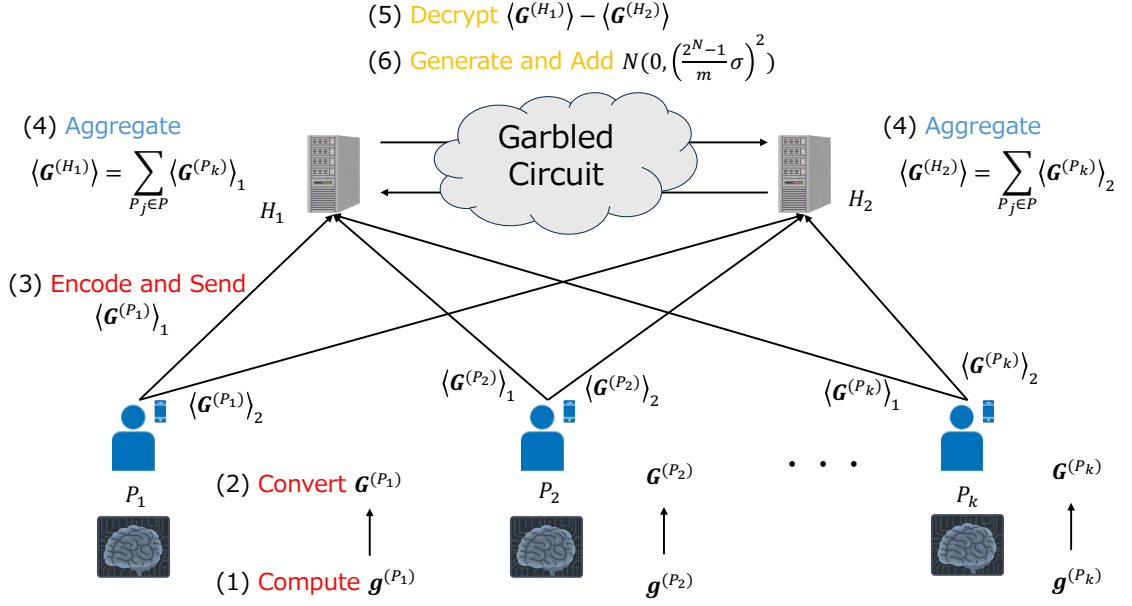


Figure 2.1: Overview of SPGC protocol.

via secure multiparty computation. Intuitively, both privacy guarantee and accuracy of SPGC are independent of the number of participants because it does not include a process for removing noise. Thus, SPGC can maintain a strong privacy guarantee even if many participants join the protocol. We show the overview of SPGC in Fig. 2.1.

SPGC allows each participant to conceal gradients, which are computed local, via secret sharing at the beginning of the protocol. Then, SPGC requires central servers to gather and aggregate the concealed gradients via secret sharing without recovering the original gradients. The aggregation process for gradients can be executed fast because the secret sharing provides servers with fast arithmetic operations for secure multiparty computation.

Next, the servers recover the aggregated gradients and simultaneously add noise to the gradients within a garbled circuit. The process of recovering the aggregated gradients and simultaneously adding noise to the gradients involves complicated computations on the secret sharing. Thus, we focused on using garbled circuits because they can evaluate any function, and the number of servers is two in SPGC because a garbled circuit is a fast protocol between two parties.

2.4 Design of SPGC

Based on the aforementioned construction, SPGC can provide both the privacy guarantee independent of the number of participants and a practical accuracy in comparison with the use of the local differential privacy.

The construction described above was inspired by the collaborative gradient computation (CGC) by Chase et al. [37], but CGC cannot distinguish positive and negative gradients from each other when a gradient is equal to a modulus of the protocol. We show the incorrectness of CGC in Section 2.4.2. In contrast, SPGC is constructed in a manner where the gradients are *always* smaller than the value of a modulus and therefore it can always train a model correctly.

2.4.2 Lack of Correctness on CGC

We show a lack of correctness on CGC by Chase et al. [37] below. Loosely speaking, there are cases in which gradients of CGC are not accurately recovered. In particular, for summation of gradients provided by all participants, the summation may be equivalent to $-mC$ or mC , that is, $-mC \leq \sum_i^m \mathbf{g}^{(x_i)} \leq mC$, where m is the number of gradients i.e., batch size. In doing so, for the use of secret sharing whose modulus is mC , values of $-mC$ and mC are indistinguishable from each other. In other words, the training will become incorrect because gradients that are identical to mC and $-mC$ are dealt with by the same value on the smod operation. The lack of the correctness described above is formally shown through Lemma 1 and Theorem 1 as described below. Here, assume that the L2 norm is utilized as a clipping norm.

At first, in CGC, let $\mathbf{v} = (v_1, v_2, \dots, v_n)$ be a vector, the clip function $\text{Clip}(C, \mathbf{v})$ be computed as $\min(1, \frac{C}{\|\mathbf{v}\|_2})\mathbf{v}$.

Lemma 1. *For any vector $\mathbf{v} = \{v_1, v_2, \dots, v_n\}$ and $C > 0$, an element $v_i \in \boldsymbol{\nu}$ where $\boldsymbol{\nu} = \text{Clip}(C, \mathbf{v})$ is $|v_i| \leq C$ for any $i \in [1, n]$.*

Proof. We prove the lemma by discussing each case with respect to $\|\mathbf{v}\|$ and C .

- For $\|\mathbf{v}\|_2 \leq C$, the following equation holds:

$$\boldsymbol{\nu} = \text{Clip}(C, \mathbf{v}) = \mathbf{v} = (v_1, v_2, \dots, v_n).$$

That is, for any $v_i \in (v_1, v_2, \dots, v_n)$, $|\nu_i| = |v_i|$ holds. Then, the following inequation holds:

$$|\nu_i| = |v_i| \leq \sqrt{|v_1|^2 + |v_2|^2 + \dots + |v_n|^2} = \|\mathbf{v}\|_2 \leq C.$$

Furthermore, when $v_j = 0$ holds for any $j \in [1, n] \setminus \{i\}$, $|\nu_i| = |v_i| = C$ holds. That is, $|\nu_i| = |v_i| \leq C$ holds for any ν_i .

- For any $\|\mathbf{v}\|_2 > C$, the following equation holds:

$$\boldsymbol{\nu} = \text{Clip}(C, \mathbf{v}) = \frac{C}{\|\mathbf{v}\|} \mathbf{v} = \frac{C}{\sqrt{|v_1|^2 + |v_2|^2 + \dots + |v_n|^2}} \mathbf{v}.$$

That is, for any $v_i \in (v_1, v_2, \dots, v_n)$, $|\nu_i| = \frac{C}{\sqrt{|v_1|^2 + |v_2|^2 + \dots + |v_n|^2}} |v_i|$ holds. Then, the following equation holds:

$$|\nu_i| = \frac{C}{\sqrt{|v_1|^2 + |v_2|^2 + \dots + |v_n|^2}} |v_i| \leq \frac{C}{\sqrt{|v_i|^2}} |v_i| = C,$$

because of $|v_i| \geq 0$ holds. Furthermore, when $v_j = 0$ holds for any $j \in [1, n] \setminus \{i\}$, $|\nu_i| = \frac{C}{\sqrt{|v_1|^2 + |v_2|^2 + \dots + |v_n|^2}} |v_i| = C$ holds. That is, $|\nu_i| = \frac{C}{\sqrt{|v_1|^2 + |v_2|^2 + \dots + |v_n|^2}} |v_i| \leq C$ holds for any ν_i .

□

Next, we show that CGC lacks correctness by presenting a counterexample where gradients are not accurately recovered to an original value. Intuitively, since a modulus on CGC is mC , gradients whose values are mC and $-mC$ are indistinguishable from each other when gradients are equivalent to mC . This fact is presented as the following theorem.

Theorem 1. *Let a per-example gradient for an example $\mathbf{x}_i \in \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ be $\tilde{\mathbf{g}}^{(\mathbf{x}_i)} = \text{Clip}(C, \nabla_{\mathbf{w}_t} \ell(f(\mathbf{w}_t; \mathbf{x}_i), y_i))$. There is then a case where a summation of all the gradients is mC and corresponds to $-mC$ in smod operation.*

Proof. First, $|\tilde{\mathbf{g}}^{(\mathbf{x}_i)}| \leq C$ holds for per-example gradients computed by each participant in accordance with Lemma 1. Then, a summation of gradients given from all the participants is computed

2.4 Design of SPGC

as

$$\tilde{\mathbf{g}} = \left| \sum_{i=1}^m \tilde{\mathbf{g}}^{(\mathbf{x}_i)} \right| \leq mC. \quad (2.1)$$

Therefore, if $\tilde{\mathbf{g}}(\mathbf{x}_i) = C$ holds for any $\mathbf{x}_i \in \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, $\tilde{\mathbf{g}} = mC$ holds from Equation (2.1).

Then, from Theorem 1, for gradients mC ,

$$\sum_{i=1}^m \tilde{\mathbf{g}}^{(\mathbf{x}_i)} \text{ smod } mC = (mC + mC) \bmod 2mC - mC = -mC$$

holds from the definition of the smod operation. Gradients with mC then become $-mC$ by the recovering phase of secure multiparty computation and therefore the original value mC cannot be recovered. $\square$

For instance, in case of $\nabla_{\mathbf{w}_t} L(\mathbf{w}_t, \mathbf{x}_i) = \{0, \dots, j, 0, \dots, 0\}$ for any $j \in \mathbb{R}$ such that $j > C$ holds, $\tilde{\mathbf{g}}(\mathbf{x}_i) = C$ holds.

2.4.3 Construction

SPGC consists of two algorithms, more specifically one for the participant-side and another for the server-side. Hereafter, we assume the following preconditions of SPGC:

- each participant $P_j \in P$ utilizes a dataset X_{P_j} as a form of batches $\{B_{1,P_j}, B_{2,P_j}, \dots, B_{\ell,P_j}\}$ of random examples where ℓ is the number of batches;
- let the entire batch size obtained from P be $m = \sum_{j \in [1,k]} |B_{t,P_j}|$ for any $t \in [1, \ell]$;
- each participant P_j for $j \in [1, k]$ owns an initial model, whose weights $\mathbf{w}_0$ and architecture are identical to those of each other.

The update of weights, more specifically the update of the model itself, is conducted on each batch. In other words, we describe only a gradient computation for a single step as the algorithms of SPGC below.

Algorithm 1 Participant-Side Process of SPGC

Input: The current batch B_{t,P_j} in a training step, the current weight w_t in a training step, the entire batch size m , gradient norm bound (sensitivity) $C > 0$, modulus 2^N where $N > \log_2(mC+1)$.

Output: $\langle G \rangle_1$ sent to a server H_1 , $\langle G \rangle_2$ sent to a server H_2 .

- 1: **for** $x_i \in B_{t,P_j}$ **do**
- 2: $\bar{g}^{(x_i)} = \nabla_{w_t} L(w_t, x_i)$
- 3: $\tilde{g}^{(x_i)} = \min(1, \frac{C}{\|\bar{g}^{(x_i)}\|_2}) \bar{g}^{(x_i)}$
- 4: **end for**
- 5: $g = \sum_{x_i \in B_{t,P_j}} \tilde{g}^{(x_i)}$
- 6: $G = \frac{2^N - 1}{mC} g$
- 7: $r \sim \text{Uniform}[-2^N, 2^N]^{|G|}$
- 8: $\langle G \rangle_1 = G + r \text{ smod } 2^N$
- 9: $\langle G \rangle_2 = r \text{ smod } 2^N$

2.4.4 Participant-Side Process

All the participants run Algorithm 1 in parallel. A participant $P_j \in P$ computes their per-example gradient $\tilde{g}^{(x_i)}$ in local. Then, P_j clips the L^2 norm for each gradient $\bar{g}^{(x_i)}$ and computes a summation g of the gradients. This clipping decides the sensitivity, which is utilized for the range of noise, and this is C in SPGC. Then, P_j converts a floating-point gradient g into a fixed-point gradient G in line 8 because of secret sharing for training. Therefore, P_j encodes G into two shares $\langle G^{(P_j)} \rangle_1$ and $\langle G^{(P_j)} \rangle_2$. Finally, outputs $\langle G^{(P_j)} \rangle_1$ and $\langle G^{(P_j)} \rangle_2$ of Algorithm 1 are sent to servers H_1 and H_2 , respectively.

2.4.5 Server-Side Process

After all the participants have run Algorithm 1, the servers H_1 and H_2 run Algorithm 2. H_1 and H_2 receive shares $\langle G^{(P_j)} \rangle_1$ and $\langle G^{(P_j)} \rangle_2$ from each participant P_j , respectively. In doing so, H_1 and H_2 need to wait until they receive the shares from all participants. Moreover, H_1 and H_2 compute $\langle G^{(H_1)} \rangle_1$ and $\langle G^{(H_2)} \rangle_2$, respectively. Then, H_1 and H_2 execute garbled circuits.

Next, in line 5, both the recovery of gradients and the generation of noise are executed within the garbled circuits. In particular, the gradients are recovered by computing $(\langle G^{(H_1)} \rangle_1 - \langle G^{(H_2)} \rangle_2) \text{ smod } 2^N$. Simultaneously, via a xor computation of s_1 and s_2 , a seed for generating noise of the differential privacy is computed. The xor computation within the garbled circuit is performed to conceal the

2.4 Design of SPGC

Algorithm 2 Server-Side Process of SPGC

Input: The participant sets P , $\langle \mathbf{G}^{(P_j)} \rangle_1$ output from participant $P_j \in P$, $\langle \mathbf{G}^{(P_j)} \rangle_2$ output from participant $P_j \in P$, the deviation σ of the Gaussian distribution, the entire batch size m , gradient norm bound $C > 0$, modulus 2^N where $N > \log_2(mC + 1)$.

Output: $\mathbf{g}^{(\text{DP})} = \sum_{P_j \in P} \mathbf{g}^{(P_j)} + \mathcal{N}(0, C^2 \sigma^2)$

- 1: $H_1 : \langle \mathbf{G}^{(H_1)} \rangle_1 = \sum_{P_j \in P} \langle \mathbf{G}^{(P_j)} \rangle_1 \text{ smod } 2^N$
 - 2: $H_1 : \text{generate a random seed } s_1$
 - 3: $H_2 : \langle \mathbf{G}^{(H_2)} \rangle_2 = \sum_{P_j \in P} \langle \mathbf{G}^{(P_j)} \rangle_2 \text{ smod } 2^N$
 - 4: $H_2 : \text{generate a random seed } s_2$
 - 5: $\mathbf{G}^{(\text{DP})} = (\langle \mathbf{G}^{(H_1)} \rangle_1 - \langle \mathbf{G}^{(H_2)} \rangle_2 \text{ smod } 2^N) + \mathcal{N}_{s_1 \oplus s_2}(0, (\frac{2^N - 1}{m} \sigma)^2)$
 - 6: $\mathbf{g}^{(\text{DP})} = \frac{mC}{2^N - 1} \mathbf{G}^{(\text{DP})}$
-

generated noise itself from each server. $\mathbf{G}^{(\text{DP})}$ is then obtained with a noise generated from the seed $s_1 \oplus s_2$ and the recovered gradient $(\langle \mathbf{G}^{(H_1)} \rangle_1 - \langle \mathbf{G}^{(H_2)} \rangle_2) \text{ smod } 2^N$. Finally, $\mathbf{g}^{(\text{DP})}$ is obtained as a floating-point value.

Note: SPGC may look similar to CGC [37], but its construction is strictly different from CGC, which contains a theoretical problem. CGC might compute a gradient imprecisely when the value of a gradient is equal to a modulus α . To overcome this problem, SPGC restricts the range of fixed-point gradients to be smaller than a modulus by setting $2^N > mC + 1$. Therefore, SPGC can always compute gradients because their values are not equal to a modulus. We also prove in Theorem 2 that SPGC can compute gradients precisely. Our idea enables SPGC to achieve higher accuracy than CGC as well. Furthermore, by creating a modulus in the smod operation in the form of 2^N , the smod operation within garbled circuits becomes faster.

2.4.6 Correctness

The correctness of gradient computation in SPGC is proven as shown in Theorem 2. To prove Theorem 2, we first present the following lemma.

Lemma 2. For any modulus $\alpha \in \mathbb{N}$ such that $\alpha > mC > 1$ holds, $|\sum_{i=1}^m \frac{\alpha-1}{mC} \tilde{\mathbf{g}}^{(\mathbf{x}_i)}| < \alpha$ holds.

Proof. From the assumption where $\alpha > mC > 1$ holds, $\frac{\alpha-1}{mC} > 0$ holds. Then, because $|\tilde{\mathbf{g}}(\mathbf{x}_i)| \leq C$

holds from Theorem 1, the following relationship holds for $\tilde{\mathbf{g}}(\mathbf{x}_i)$:

$$\begin{aligned} -C &\leq \tilde{\mathbf{g}}(\mathbf{x}_i) \leq C \\ \Leftrightarrow -\frac{(\alpha-1)}{m} &\leq \frac{\alpha-1}{mC} \tilde{\mathbf{g}}(\mathbf{x}_i) \leq \frac{(\alpha-1)}{m}. \end{aligned}$$

Then, on a summation of gradients, the following relationship holds from the relationship described above:

$$\begin{aligned} -\alpha + 1 &\leq \sum_{i=1}^m \frac{\alpha-1}{mC} \tilde{\mathbf{g}}(\mathbf{x}_i) \leq \alpha - 1 \\ \Leftrightarrow -\alpha &< \sum_{i=1}^m \frac{\alpha-1}{mC} \tilde{\mathbf{g}}(\mathbf{x}_i) < \alpha. \end{aligned}$$

Therefore, for any $\alpha \in \mathbb{N}$ such that $\alpha > mC > 1$ holds, $|\sum_i^m \frac{\alpha-1}{mC} \tilde{\mathbf{g}}(\mathbf{x}_i)| < \alpha$ holds. □

□

The above lemma restricts a summation of gradients in SPGC to the range of values less than $\pm\alpha$. Here, a modulus α is defined as 2^N in SPGC as shown in Section 2.4.3. Then, the following theorem is proven.

Theorem 2. *If all servers and participants follow the SPGC protocol, an output of Algorithm 2 is $\sum_{P_j \in P} \mathbf{g}^{(P_j)} + \mathcal{N}(0, C^2 \sigma^2)$.*

Proof. H_1 and H_2 compute as follows:

$$\begin{aligned} \langle \mathbf{G}^{(H_1)} \rangle_1 &= \sum_{P_j \in P} \langle \mathbf{G}^{(P_j)} \rangle_1 \text{ smod } 2^N \\ &= \sum_{P_j \in P} (\mathbf{G}^{(P_j)} + \mathbf{r}^{(P_j)} \text{ smod } 2^N) \text{ smod } 2^N \\ &= (\sum_{P_j \in P} \mathbf{G}^{(P_j)} \text{ smod } 2^N + \sum_{P_j \in P} \mathbf{r}^{(P_j)} \text{ smod } 2^N) \text{ smod } 2^N, \end{aligned}$$

2.4 Design of SPGC

and

$$\begin{aligned}\langle \mathbf{G}^{(H_2)} \rangle_2 &= \sum_{P_j \in P} \langle \mathbf{G}^{(P_j)} \rangle_2 \text{ smod } 2^N \\ &= \sum_{P_j \in P} \mathbf{r}^{P_j} \text{ smod } 2^N.\end{aligned}$$

Moreover, the following equation is computed within garbled circuits:

$$\begin{aligned}(\langle \mathbf{G}^{(H_1)} \rangle_1 - \langle \mathbf{G}^{(H_2)} \rangle_2) \text{ smod } 2^N &= (\sum_{P_j \in P} \mathbf{G}^{(P_j)} \text{ smod } 2^N) \text{ smod } 2^N \\ &= \sum_{P_j \in P} \mathbf{G}^{(P_j)} \text{ smod } 2^N\end{aligned}\tag{2.2a}$$

$$= \sum_{P_j \in P} \mathbf{G}^{(P_j)}.\tag{2.2b}$$

For each batch, a summation of gradients provided by each participant P_j is equal to a summation of per-example gradients computed from Algorithm 1. That is, $\sum_{i=1}^m \frac{\alpha-1}{mC} \tilde{\mathbf{g}}^{(\mathbf{x}_i)} = \sum_{P_j \in P} \mathbf{G}^{(P_j)}$ holds. Then, the conversion from Equation (2.2a) to Equation (2.2b) is based on Lemma 2 described above and Lemma 4 in [37].

Next, the noise generated from the Gaussian distribution is added to the recovered gradients $(\langle \mathbf{G}^{(H_1)} \rangle_1 - \langle \mathbf{G}^{(H_2)} \rangle_2) \text{ smod } 2^N$. The output of $\mathbf{G}^{(\text{DP})}$ is expanded by $\frac{2^N-1}{mC}$, which means to scale back a floating point by multiplying $\frac{mC}{2^N-1}$. After scaling back $\mathbf{G}^{(\text{DP})}$ to $\mathbf{g}^{(\text{DP})}$, the generated noise is almost equal to the noise generated from the Gaussian distribution $\mathcal{N}(0, C^2\sigma^2)$.

An approximate computation of $\mathbf{g}^{(\text{DP})} = \sum_{P_j \in P} \mathbf{g}^{(P_j)} + \mathcal{N}(0, C^2\sigma^2)$ is then obtained. $\square$

2.4.7 Privacy Analysis

We analyze the privacy of computation and the privacy of output for SPGC below.

Privacy of Computation

First, we assume that both the garbled circuits and secret sharing achieve privacy of computation. Then, the privacy of computation in the training phase of the SPGC protocol can be achieved following the composition theorem by Kushilevitz et al. [43]. In other words, in Algorithm 2, the computations in lines 1 and 3 are executed via secret sharing, while the computation in line 5 is executed within a garbled circuit. Unless an adversary colludes with both H_1 and H_2 , the adversary cannot identify any value except for $\mathbf{g}^{(\text{DP})}$. In addition, since noise with differential privacy is generated within garbled circuits, an adversary cannot know the accurate noise value. Consequently, no information with respect to training data X_{P_1} is revealed even if an adversary colludes with either H_1 or H_2 and any participant $P_{j \neq 1}$. More formally, following the proof of CGC [37], we can prove the following theorem.

Theorem 3. *If all servers and participants follow the protocol, no information except for an output $\mathbf{g}^{(\text{DP})} = \sum_{P_j \in P} \mathbf{g}^{(P_j)} + \mathcal{N}(0, C^2 \sigma^2)$ of Algorithm 2 is revealed to an adversary.*

Proof. In this proof, we discuss from two standpoints, more specifically the participant-side and server-side. On the participant-side, any participant will obtain the end result $\mathbf{g}^{(\text{DP})}$ from H_1, H_2 and nothing else. Hence, without colluding with H_1 and H_2 , an adversary cannot reveal any information about $\mathbf{g}^{(P_1)}$. On the server-side, $\langle \mathbf{G}^{(P_j)} \rangle_1 = \mathbf{G}^{(P_j)} + \mathbf{r}^{(P_j)} \bmod 2^N$ and $\langle \mathbf{G}^{(P_j)} \rangle_2 = \mathbf{r}^{(P_j)} \bmod 2^N$ are uniformly distributed values by virtue of secret sharing as shown in Section 2.2. If there exists a mechanism M' controlled by an adversary, which is given $\mathbf{g}^{(\text{DP})}$ and $\mathbf{G}^{(P_j)} + \mathbf{r}^{(P_j)}$ and then predicts any property p about $\mathbf{G}^{(P_1)}$, then the following distribution is obtained from the view of the adversary:

$$\begin{aligned}
 & \Pr[M'(\mathbf{g}^{(\text{DP})}, \langle \mathbf{G}^{(P_j)} \rangle_1) = p(\mathbf{G}^{(P_1)})] \\
 &= \Pr \left[M'(\mathbf{g}^{(\text{DP})}, \langle \mathbf{G}^{(P_j)} \rangle_1) = p(\mathbf{G}^{(P_1)}) \mid \langle \mathbf{G}^{(P_j)} \rangle_1 \right] \\
 &= \Pr \left[M'(\mathbf{g}^{(\text{DP})}, \langle \mathbf{G}^{(P_j)} \rangle_1) = p(\mathbf{G}^{(P_1)}) \mid \langle \mathbf{G}^{(P_j)} \rangle_1 \sim \text{Uniform}[-2^N, 2^N] \right].
 \end{aligned} \tag{2.3}$$

Therefore, no mechanism M' can reveal any more information other than that viewed by any participant and server, who observe only $\mathbf{g}^{(\text{DP})}$. $\square$

2.5 Experiments

Privacy of Output

To achieve privacy of output, SPGC needs to satisfy the differential privacy [44, 45]. When adding noise with (ϵ, δ) -differential privacy in one step of training, SPGC constantly satisfies $(\tilde{\epsilon}, \tilde{\delta})$ -differential privacy after T epochs for the number of participants k based on the theorem 4. In contrast, the existing works [35, 36] satisfy $(\sqrt{k}\tilde{\epsilon}, \tilde{\delta})$ -differential privacy. In other words, unlike existing works, SPGC can guarantee privacy independent of the number of participants.

Theorem 4. *If all servers and participants follow the protocol and (ϵ, δ) -differential privacy at each batch of the training, SPGC satisfies $(\tilde{\epsilon}, \tilde{\delta})$ -differential privacy in the entire process of the training after T epochs, where*

$$\begin{aligned}\tilde{\delta} &= 1 - (1 - \delta)^T(1 - \delta), \\ \tilde{\epsilon} &= \min \left\{ T\epsilon, \frac{(e^\epsilon - 1)T\epsilon}{e^\epsilon + 1} + \epsilon\sqrt{2T \log(e + \frac{\sqrt{T}\epsilon^2}{\tilde{\delta}})}, \right. \\ &\quad \left. \frac{(e^\epsilon - 1)T\epsilon}{e^\epsilon + 1} + \epsilon\sqrt{2T \log(\frac{1}{\tilde{\delta}})} \right\}.\end{aligned}\tag{2.4}$$

Proof. For computation on each batch, $\sum_{P_j \in P} \mathbf{g}^{(P_j)} + \mathcal{N}(0, C^2\sigma^2)$ is obtained from an output of Algorithm 2. In doing so, σ is chosen so that (ϵ, δ) -differential privacy is satisfied with each batch. Since a training dataset for each participant is disjointedly divided into ℓ batches, each sample appears at once. Then, a perturbed gradient based on (ϵ, δ) -differential privacy is computed for each batch, and then a model is updated with the gradient. In doing so, SPGC satisfies (ϵ, δ) -differential privacy for each epoch following the parallel composition theorem [46]. Therefore, from Theorem 3.4 by Kairouz et al. [47], an output of the algorithms of SPGC after T epochs satisfies the $(\tilde{\epsilon}, \tilde{\delta})$ -differential privacy described above. $\square$

2.5 Experiments

In this section, we present experiments for measuring the accuracy and training time of SPGC.

2.5.1 Implementation

We conducted experiments with several academic and medical diagnosis datasets to measure the performance of SPGC. All the experiments were conducted on a computer having Ubuntu 18.04, 83GB RAM, Intel Xeon(R) CPU E5-2630 v3 2.40GHz, and no GPU.

Training. SPGC is implemented mainly with the TensorFlow library¹ in python, which is an open-source platform developed by Google. Keras in particular is utilized with a TensorFlow backend for training a neural network model. Meanwhile, to restrict the sensitivity during the training, gradients are computed with the `per_example_gradient` operator [48] and then clipped in TensorFlow. When this is easily implemented, the computational complexity becomes $\mathcal{O}(m)$ relative to batch size m . Surprisingly, by utilizing the gradient computation proposed by Goodfellow [48], the process is vectorized, more specifically, computations become faster because parallel computations are available. The TensorFlow library contains the `vectorized_map` function for computing the gradient mentioned above, and it is utilized in our implementation.

Server Communication. A secure multiparty computation protocol often needs a heavy communication overhead during protocol execution in addition to the computation itself. The communication process of SPGC is implemented with the gRPC library² in python. The gRPC library is an open-source framework that can design a communication environment with a remote computer and measure training rigorously. Thus, the actual performance of SPGC can be measured via environments with LAN and WAN settings. Meanwhile, garbled circuits used between servers are implemented with the TinyGarble library [41] in C++. The noise generation for differential privacy is directly embedded in the garbled circuits so that the circuits themselves contain the Gaussian distribution, which is identical to a random seed. Consequently, noise identical to a random seed is generated in garbled circuits.

2.5.2 Experimental Setup

Purpose of Experiments. The purpose of our experiments is to evaluate the accuracy and the training time of SPGC for academic and medical diagnosis datasets. To do this, we first evaluate

¹TensorFlow: <https://www.tensorflow.org/>

²gRPC: <https://grpc.io/docs/tutorials/basic/python/>

2.5 Experiments

the accuracy and training time of SPGC relative to the amount of noise for differential privacy. Then, we compare the accuracy and training time of SPGC with the baselines, namely, non-privacy setting, the use of differential privacy, and secure multiparty computation.

Baseline. The following settings are utilized as baselines to compare with the performance of SPGC.

- **Non-privacy.** We train a model without the privacy guarantee. Effects on accuracy and computational overhead for training are evaluated compared to the performance of the Non-privacy setting. The Non-privacy setting significantly differs from the setting of SPGC in that it assumes the use of a single server, which aggregates gradients sent from participants.
- **Local differential privacy (LDP).** We train a model with a mechanism satisfying local differential privacy (LDP) [24]. LDP is a trivial way for the privacy guarantee in collaborative learning and is also faster than SGPC because it does not use secure multiparty computation. The performance improvement of accuracy and its related overhead on the training phase for SPGC can be evaluated by comparing them with the accuracy and training time for LDP.
- **Only secure multiparty computation (Only-MPC).** In SPGC protocol, we train a model with only garbled circuits and secret sharing instead of noise generation. Original data is recovered from shares within the garbled circuits. The Only-MPC setting is identical to a classic construction [49] based on secure multiparty computation. Accordingly, effects on accuracy and training time with respect to noise generation for the differential privacy in SPGC can be evaluated.

Although one might think to compare SPGC with the existing works [35, 36], their implementations have not been released. An implementation of CGC [37] was not released, but CGC is the motivation of SPGC. Thus, based on experimental results described below, we compare SPGC with CGC by referring to the performance described in [37] in Section 2.6. As described by more details later on, we compare only the accuracy of SPGC with CGC since CGC evaluated only the accuracy on the MNIST dataset.

Choice of Parameters in SPGC. On Algorithm 1 and Algorithm 2, let the number of participants

be 3, the gradient norm bound C be 1, and N be 16, more specifically a 16-bit integer is utilized. The standard deviation of a noise for differential privacy is computed as $\sigma = \sqrt{2 \log(\frac{1.25}{\delta})} / \epsilon$ relative to parameters ϵ and δ of the differential privacy [38]. Meanwhile, noise parameters are $\epsilon = 0.5, 2.0$, and 8.0 , which are in common with CGC [37]. These parameters are used also in existing privacy-preserving machine learning [18, 50], for instance the MNIST evaluation in [18] utilized $\epsilon = 0.5, 2.0$, and 8.0 whereas the Cancer evaluation in [50] utilized $\epsilon = 8.0$. Likewise, let δ be 10^{-3} . Parameters ϵ and δ in SPGC indicate the privacy level at one step in training. Meanwhile, ϵ and δ in LDP indicate the privacy level when a participant generates noise locally. The privacy levels in both settings may seem to have different values, but an adversary can discover a perturbed gradient by an honest participant in SPGC if the adversary colludes with all participants except for the honest participant. In particular, SPGC provides the same privacy level as when only an honest participant adds noise with a gradient with LDP. We can, therefore, fairly compare the privacy level of SPGC with that of LDP at one step in training.

Datasets and Their Architectures. The datasets used in the experiments and their architectures are shown below.

MNIST: The MNIST dataset contains 70,000 images of handwritten digits from 0 to 9. In particular, the dataset has 60,000 training samples and 10,000 test samples, each with 784 features representing 28×28 pixels in the image. In this experiment, the training samples are equally divided among three participants or more specifically 20,000 samples per participant for training. Furthermore, the batch size for each participant is set to 1,000, that is, the entire batch size $m = 3,000$. A 3-layer network consisting of the following structure is utilized as shown in Fig. 2.2:

1. a 5×5 convolution layer with 16 outputs per window, a $2 - 2$ stride, ReLU activation function, and padding;
2. a $1 \times 5 \times 5$ convolution layer with 32 outputs per window, a $1 - 2 - 2$ stride, no weight sharing on the first axis, and ReLU activation function;
3. a dense layer with 10 outputs.

Cancer: The Cancer dataset³ contains 569 data points with each point containing 30 real-valued

³Cancer: <https://www.kaggle.com/uciml/breast-cancer-wisconsin-data>

2.5 Experiments

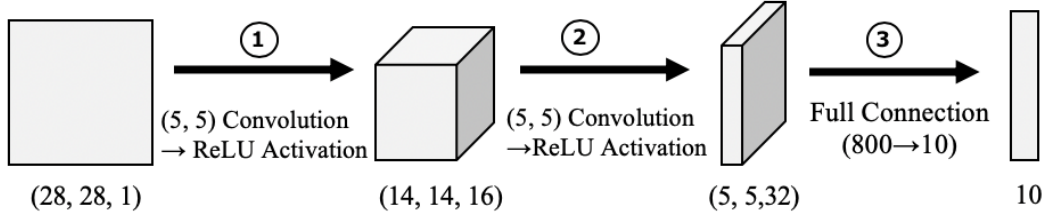


Figure 2.2: The architecture used in our experiment for MNIST.

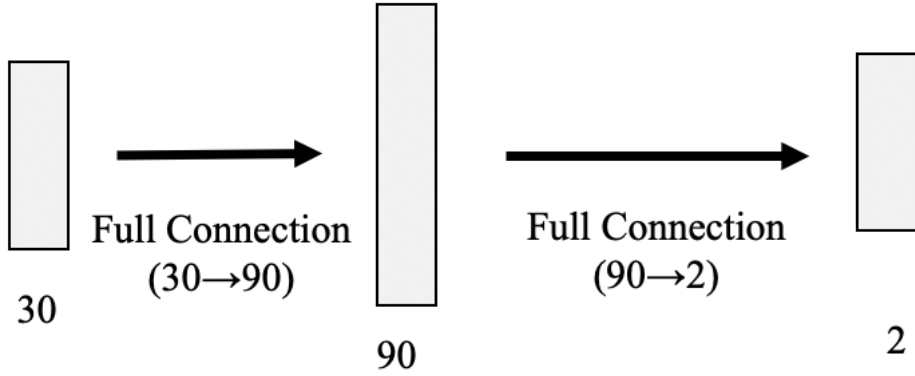


Figure 2.3: The architecture used in our experiment for Cancer.

features. The inference task is to infer whether a tumor is cancerous or benign. In this experiment, the dataset is divided into 390 training samples and 179 test samples, and then each participant owns 130 training samples for training. The batch size for each participant is set to 10, that is, $m = 30$. We utilized a single full connection layer from 90 to 2 neurons as shown in Fig 2.3.

Diabetes: The Diabetes dataset⁴ contains 768 data points with each point containing 8 real-valued features. The inference task is to infer whether a tumor is diabetic or benign. In this experiment, the dataset is divided into 600 training samples and 168 test samples, and then each participant owns 200 training samples for training. The batch size for each participant is set to 10,

⁴Diabetes: <https://www.kaggle.com/uciml/pima-indians-diabetes-database>

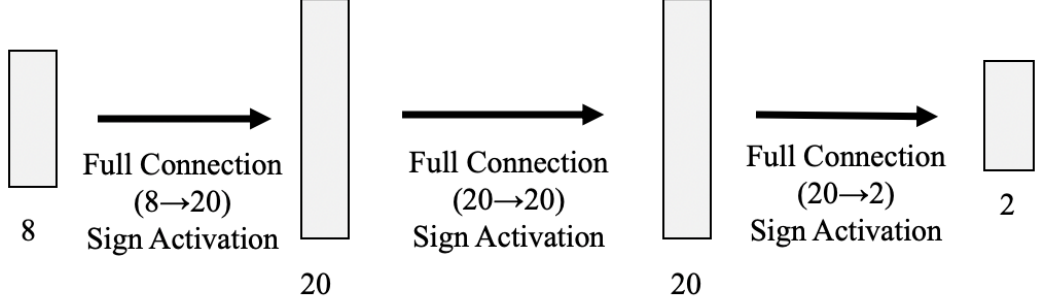


Figure 2.4: The architecture used in our experiment for Diabetes.

that is, $m = 30$. An architecture for Diabetes as shown in Fig. 2.4 is the following structure:

1. a full connection layer with 20 neurons and a Sign activation function;
2. a full connection layer with 20 neurons and a Sign activation function;
3. a dense layer with 2 outputs.

2.5.3 Results

The inference accuracy and training time of SPGC with each dataset are described below.

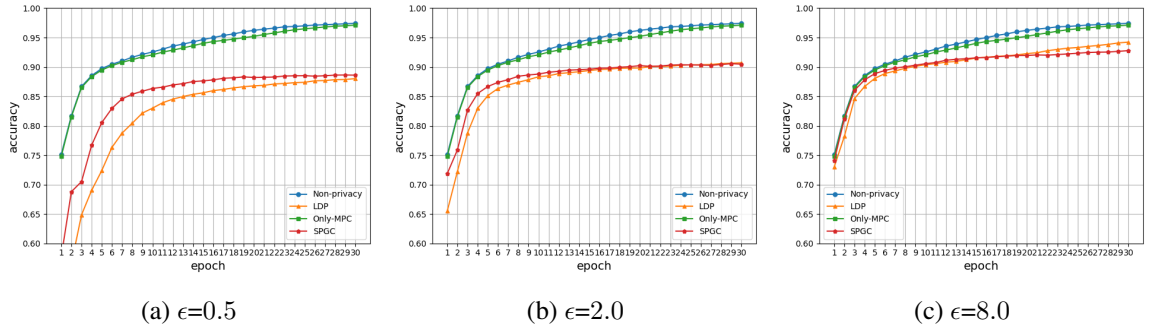


Figure 2.5: Inference accuracy of SPGC with MNIST.

2.5 Experiments

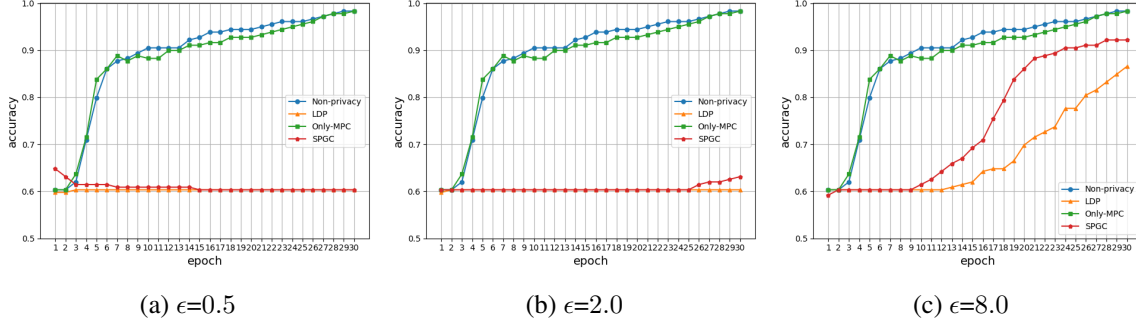


Figure 2.6: Inference accuracy of SPGC with Cancer.

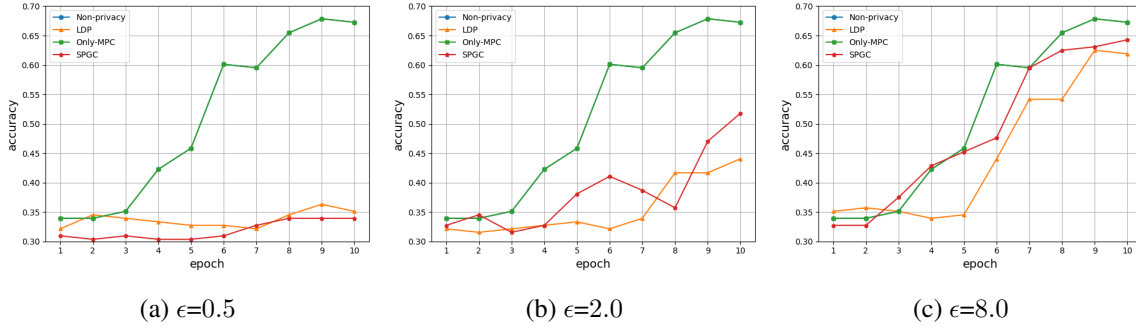


Figure 2.7: Inference accuracy of SPGC with Diabetes.

Accuracy. The accuracy of inference of SPGC with the MNIST, Cancer, and Diabetes datasets is shown in Fig. 2.5, Fig. 2.6, and Fig. 2.7, respectively.

MNIST: The accuracy of inference is 97.4% under the non-privacy setting and 97.1% under the Only-MPC setting. In contrast, the accuracy of SPGC is 88.6%, 90.5%, and 92.8% for each noise as presented in Fig. 2.5a, Fig. 2.5b, and Fig. 2.5c, respectively. These results show that SPGC improved accuracy compared to the accuracy of the LDP setting, especially with an intense noise such as $\epsilon = 0.5$. Meanwhile, the SPGC and the LDP setting have almost the same accuracy at epoch 18 for $\epsilon = 8.0$ and $\epsilon = 2.0$. This means that, in proportion to the amount of noise, e.g., in Fig. 2.5a and Fig. 2.5b, the difference in the accuracy between LDP and SPGC becomes smaller.

Cancer: Both the non-privacy setting and the Only-MPC setting have 98.3% accuracy. Meanwhile, as shown in Fig. 2.6a and Fig. 2.6b, the accuracy of SPGC is 60.3 % and 63.1% for $\epsilon = 0.5$

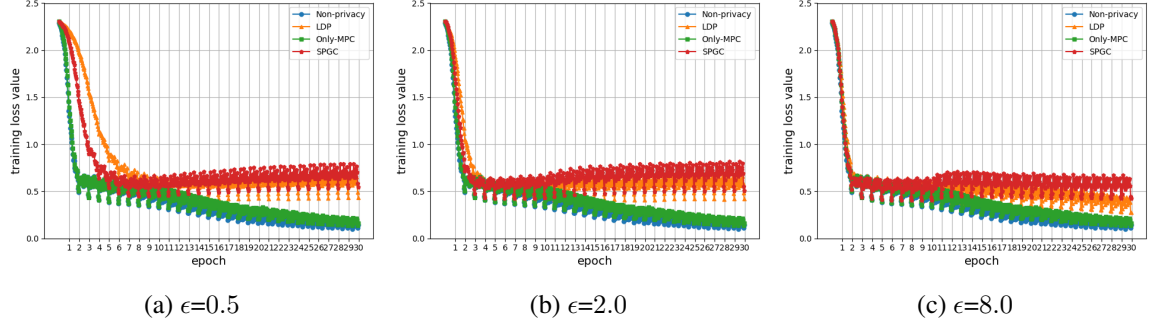


Figure 2.8: Training loss value of SPGC with MNIST.

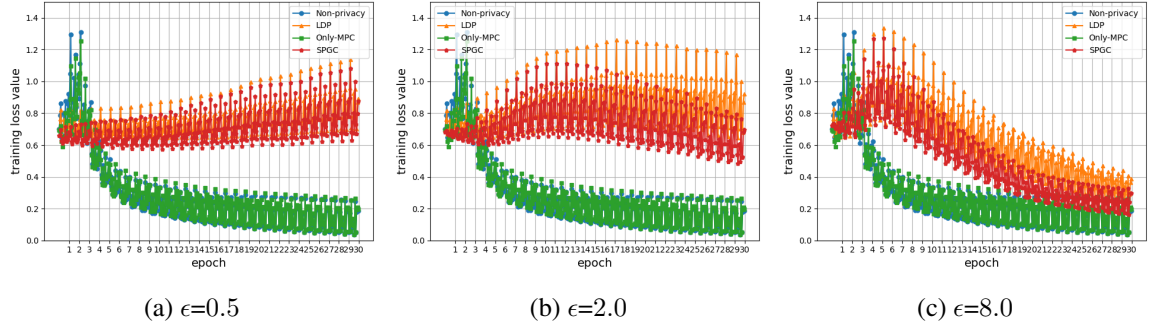


Figure 2.9: Training loss value of SPGC with Cancer.

and $\epsilon = 2.0$, respectively. In contrast, the accuracy for $\epsilon = 8.0$ is 92.1%, as shown in Fig. 2.6c. Besides, in a comparison between SPGC and the LDP setting, as shown in Fig. 2.6a, there is no difference between those accuracies at epoch 30, which is the convergence of the training on the Non-privacy setting, when the amount of noise is large at $\epsilon = 0.5$. Nevertheless, the accuracy of SPGC is 92.1%, and the LDP setting is 86.6% when the amount of noise is small at $\epsilon = 8.0$. In other words, SPGC has higher accuracy of inference than the LDP setting.

Diabetes: Both the non-privacy setting and the Only-MPC setting have 67.2% accuracy. On the other hand, as shown in Fig. 2.7a, the accuracy by SPGC is 33.9%, which is lower than the LDP setting. In contrast, in Fig. 2.7b, the accuracy of SPGC at the convergence of training is 51.8% while that of the LDP setting is 44.0%. Furthermore, in Fig. 2.7c, the accuracy of SPGC is 64.5% while that of the LDP setting is 61.9%.

2.6 Discussion

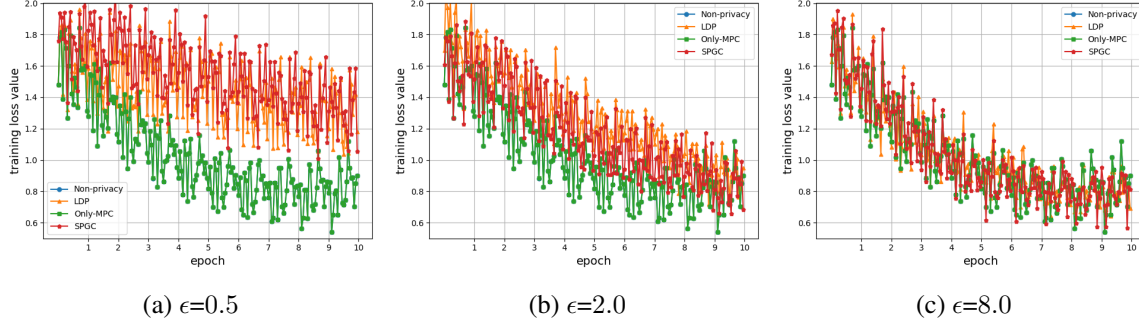


Figure 2.10: Training loss value of SPGC with Diabetes.

Training Time. Training time of SPGC with MNIST, Cancer and Diabetes datasets are shown in Fig. 2.11a, Fig. 2.11b and Fig. 2.11c, respectively.

MNIST: SPGC required 115.7 hours while the Only-MPC setting required only 70.1 hours, a difference of 45.6 hours. Meanwhile, there was a 3-hour difference in the training time between $\epsilon = 8.0$ and $\epsilon = 0.01$. The total training time for SPGC is 72 times longer than that of LDP.

Cancer: SPGC required 10.8 hours for the training while the Only-MPC setting required 6.5 hours, a difference of 4.3 hours. Meanwhile, there was a 0.4 hour difference in the training time between $\epsilon = 8.0$ and $\epsilon = 0.01$. The total training time for SPGC is 98 times longer than that of LDP.

Diabetes: SPGC required 1.23 hours for the training while the Only-MPC setting required 0.80 hours, a difference of 0.44 hours. Meanwhile, there was a 0.05 hours difference in the training time between $\epsilon = 8.0$ and $\epsilon = 0.01$. The total training time for SPGC is 21 times longer than that of LDP.

2.6 Discussion

In this section, we discuss considerations on the accuracy and training time of SPGC and the effect on the number of participants.

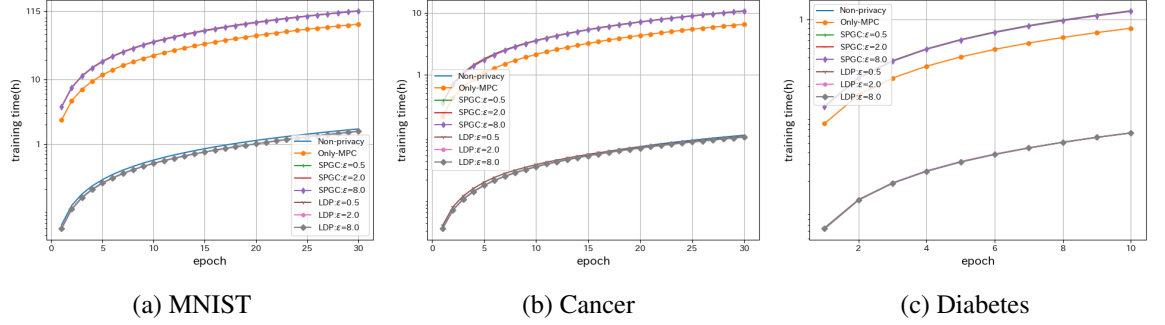


Figure 2.11: Training time of SPGC with each dataset

Table 2.1: Details on training time: for each dataset, let Training(h) be the training time until convergence, Offline(h) be gradient computation time as shown in lines 1-5 of Algorithm 1, Online(h) be the remaining time of Algorithm 1 and Algorithm 2. The values enclosed in () are under the LDP setting.

	MNIST			Cancer			Diabetes		
	Offline	Online	Training	Offline	Online	Training	Offline	Online	Training
$\epsilon = 0.01$	1.69(1.53)	115.77(0.04)	117.45(1.57)	0.11(0.11)	10.93(0.01)	11.03(0.12)	0.07(0.07)	1.19(0.00)	1.26(0.07)
$\epsilon = 0.1$	1.66(1.53)	114.47(0.04)	116.13(1.57)	0.11(0.10)	10.50(0.01)	10.61(0.11)	0.07(0.06)	1.20(0.00)	1.27(0.06)
$\epsilon = 0.5$	1.67(1.55)	114.05(0.04)	115.7(1.59)	0.11(0.10)	10.65(0.01)	10.76(0.11)	0.07(0.06)	1.16(0.00)	1.23(0.06)
$\epsilon = 2.0$	1.67(1.55)	113.45(0.04)	115.1(1.59)	0.11(0.09)	10.52(0.00)	10.63(0.10)	0.07(0.06)	1.15(0.00)	1.22(0.06)
$\epsilon = 8.0$	1.67(1.55)	112.78(0.04)	114.5(1.59)	0.11(0.09)	10.53(0.00)	10.63(0.10)	0.07(0.06)	1.14(0.00)	1.21(0.06)
Only-MPC	1.66	70.16	71.82	0.11	6.40	6.51	0.07	0.73	0.80
Non-privacy	1.68	0.05	1.73	0.10	0.01	0.11	0.06	0.00	0.06

Table 2.2: Communication overhead (GB): for each dataset, these values are measured between H_1 and H_2 .

	MNIST	Cancer	Diabetes
$\epsilon = 0.01$	7758.0	736.9	81.6
$\epsilon = 0.1$	7746.9	707.8	78.4
$\epsilon = 0.5$	7641.9	711.7	78.9
$\epsilon = 2.0$	7641.9	699.8	77.5
$\epsilon = 8.0$	7528.2	698.8	77.5
Only-MPC	433.8	39.5	4.4

2.6.1 Accuracy

Based on the results shown in Section 2.5.3, we discuss the performance of SPGC in terms of two standpoints, more specifically the difference in accuracy between SPGC and LDP for each dataset and those between the Non-privacy setting and the Only-MPC setting. To do this, we also measure

2.6 Discussion

training loss values as shown in Fig. 2.8 to Fig. 2.10. Training loss values represent the convergence of the training process. The figures show that the training on each dataset is converged. We note that a periodic change in the training loss values is often caused by typical training even in the Non-privacy setting.

Comparing SPGC with the LDP setting, the accuracy of SPGC on the MNIST dataset is almost the same as the LDP setting for $\epsilon = 8.0$. Specifically, a sufficient number of participants is needed to improve the accuracy of the LDP setting. Since noise is generated from the Gaussian distribution with a mean of zero, the amount of noise becomes close to zero when many participants join the protocols. The number of participants is three in the experimental setting, and the number of participants is insufficient to make the amount of noise zero in the LDP setting. Therefore, LDP achieved the same accuracy for $\epsilon = 8.0$ as SPGC. For $\epsilon = 0.5$, SPGC can provide higher accuracy than LDP because much noise is provided in the LDP setting. In contrast, for $\epsilon = 8.0$, the accuracy of LDP became higher than SPGC after epoch 23. The result is thought to stem from the bit truncation caused by the use of secure multiparty computation, more specifically the use of modulo operations. In this setting, the bit truncation by the secure multiparty computation affects accuracy significantly more than the noise itself.

Likewise, on evaluation of Cancer, SPGC can provide higher accuracy for $\epsilon = 8.0$. Meanwhile, for $\epsilon = 0.5$ and $\epsilon = 2.0$, the accuracy of SPGC is decreased in proportion to epochs. The decrease was caused because noise for $\epsilon = 0.5$ and $\epsilon = 2.0$ are too strong for the Cancer dataset, as shown in Fig. 6. Samples near the decision boundary are often misclassified due to these strong noises. Namely, only the noise for $\epsilon = 8.0$ is suitable for the Cancer dataset, and then the relation between SPGC and the LDP setting is familiar as seen in Fig. 2.5a. Consequently, the figures show that SPGC can control noise better than LDP.

Nevertheless, for $\epsilon = 0.5$ in Diabetes, the accuracy worsens in proportion to the number of epochs on both SPGC and the LDP setting. Because this noise is too large relative to the original data features, inference became entirely random. Meanwhile, for $\epsilon = 2.0$ in the evaluation of Diabetes, SPGC improved the accuracy after epoch 26 compared to the LDP setting. This improvement is thought to stem from greater control of noise by SPGC than in the LDP setting.

Next, the difference in accuracy between the Only-MPC setting and Non-privacy setting is

caused by the bit truncation from the conversion to the fixed points for modulo operations. More specifically, in the evaluation of MNIST, the accuracy of the Only-MPC setting decreased by a few percent because of the bit truncation since MNIST contains more features than the other two datasets. Likewise, in the evaluation of Cancer, the bit truncation affected the accuracy for lower epochs than 27. However, the accuracy of the Only-MPC setting is identical to Non-privacy at the convergence of the training, which is epoch 30. In contrast, on the evaluation of Diabetes, the accuracy of the Only-MPC setting is identical to that of Non-privacy because the number of data samples is small, and there is no sample whose evaluation result is changed by the bit truncation. This means that the Only-MPC setting does not affect accuracy for medical diagnosis datasets.

Finally, we compare the accuracy of SPGC with CGC [37]. An implementation of CGC has not yet been released so we refer here to the accuracy of the MNIST evaluation described in the original CGC paper [37]. For instance, the accuracies of CGC are 82.3% for $\epsilon = 0.5$, 87.8% for $\epsilon = 2.0$, and 92.1% for $\epsilon = 8.0$, respectively. On the other hand, those of SPGC are 88.6% for $\epsilon = 0.5$, 90.5% for $\epsilon = 2.0$, and 92.8% for $\epsilon = 8.0$, respectively. We believe that the above advantage of SPGC over CGC is by virtue of our conversion from a floating-point gradient into a fixed-point gradient, more specifically revising the incorrectness of CGC. The above advantage becomes more obvious in proportion to the noise parameter ϵ .

2.6.2 Training Time

We discuss the effect on the training time for SPGC. According to Figs. 2.8- 2.10, the convergence of the training process in SPGC and the LDP setting is slower than the Non-privacy setting. This is considered an adverse effect caused by the noise for differential privacy. Namely, gradients may draw away the convergence due to the noise generated from the Gaussian distribution. Hence, the periodic change of the training loss values becomes larger, and consequently, the convergence becomes slower.

Also, training time increases when an amount of noise increases as shown in Section 2.5.3 in Fig. 2.11a. This increase is caused by two computations, the gradient computations by participant in lines 1-4 on Algorithm 1 and the communication overhead between two servers in line 5 on

2.6 Discussion

Algorithm 2. To provide further evidence from the standpoints described above, we conduct more experiments with additional parameters $\epsilon = 0.01$, $\epsilon = 0.1$ for each dataset, and then discuss the results (including those in the previous section) in detail. More specifically, we measure the computational time for gradients in lines 1-6 on Algorithm 1 and the communication overhead in Algorithm 2. We show the results in Table 2.1.

According to Table 2.1, for all datasets, the computational time for Algorithm 2 increases in proportion to ϵ , more specifically in the amount of noise. In contrast, the computational time for gradients is almost stable. This means that the training time depends on the operations of the server-side. Indeed, by evaluating the computational time for the algorithms in detail, we discovered that the server-side operations consumed about 98% of the total computational time. Although SPGC needs significantly heavier online results than the LDP setting, we recall that the accuracy of the LDP setting deteriorates, as described in Section 2.6.1. For instance, for $\epsilon = 0.1$ on the MNIST evaluation, the accuracy of SPGC is 83.6% while the LDP setting is 78%. Likewise, for $\epsilon = 8.0$ on the Cancer evaluation, the accuracy of SPGC is five points higher than the LDP setting. Thus, it is considered that SPGC still has an advantage because its accuracy is higher in spite of worse online results.

More specifically, a large majority of the computational time is caused by communication for garbled circuits between servers. Table 2.2 shows results on the communication overhead between two servers for each dataset and parameter. According to the table, the computational cost of SPGC significantly differs from the Only-MPC setting for all datasets. The difference was caused by the noise generated within the garbled circuits. Notably, the communication overhead of SPGC becomes more prominent in proportion to the amount of noise, e.g., 433.8GB at the Only-MPC setting, 7528.2GB at $\epsilon = 8.0$ and 7758.0GB at $\epsilon = 0.01$.

The reason is that the size of garbled circuits becomes larger to generate noise for the differential privacy in proportion to the noise itself. Let σ' be $\frac{2^N-1}{m}\sigma$. To generate noise with $\mathcal{N}(0, \sigma'^2)$ within garbled circuits, the scaled standard deviation σ' is multiplied by the noise generated from the Gaussian distribution $\mathcal{N}(0, 1)$. The size of the circuits increases with respect to a value of σ' .

Based on the observation mentioned above, we conclude that the training time increases significantly due to increasing the communication overhead relative to noise generation. Dwork et al. [51]

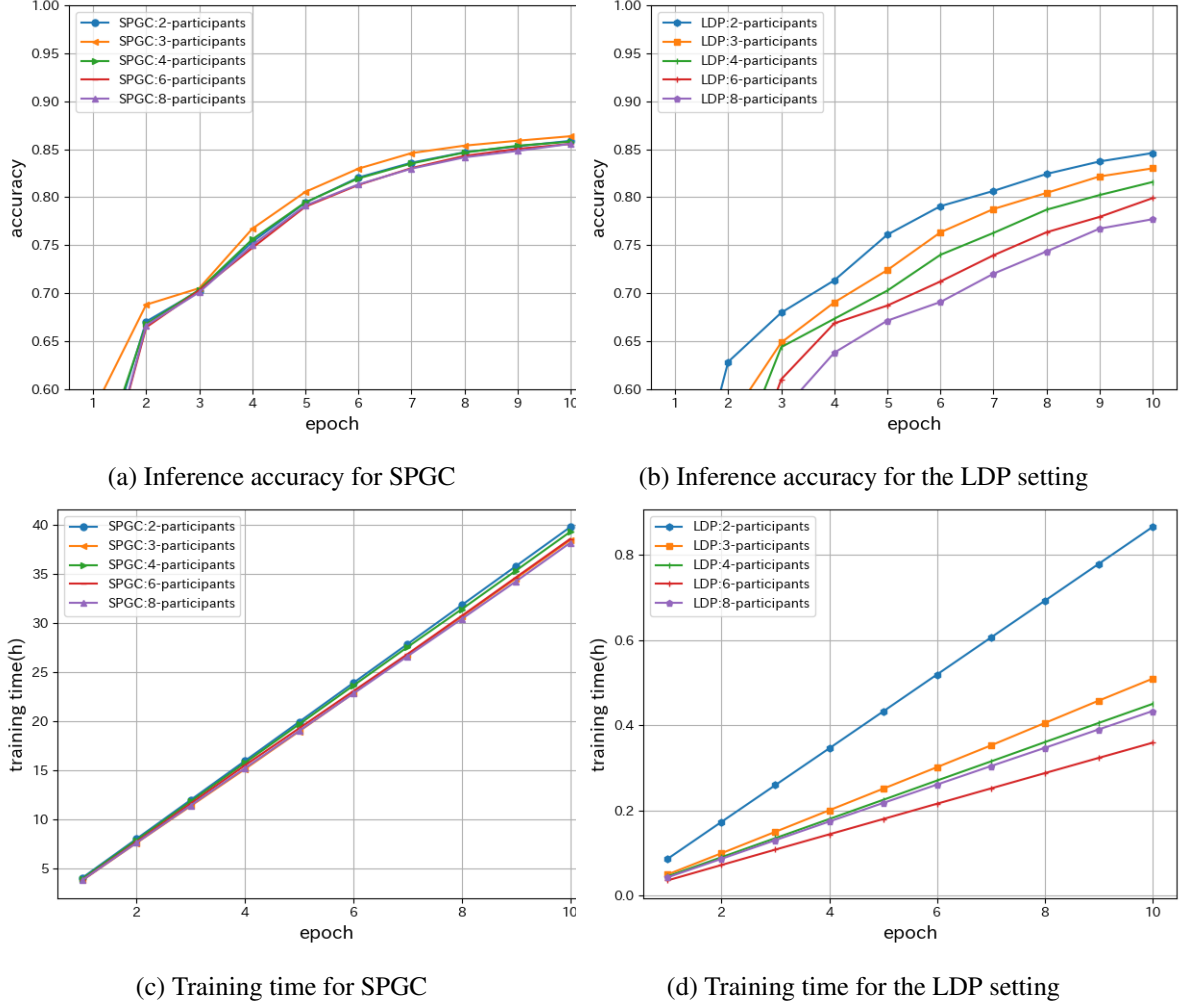


Figure 2.12: Effect of the number of participants on accuracy and training time. We measured the accuracy and the training time for a noise parameter $\epsilon = 0.5$ on the evaluation of the MNIST dataset by changing the number of participants.

showed that the communication overhead of secret sharing increases due to the amount of noise for differential privacy. Our observation about the relationship between garbled circuits and differential privacy is identical to the result by Dwork et al., except that we use garbled circuits in collaborative learning.

2.6.3 Number of participants

To understand the effect from the number of participants on accuracy and training time, we evaluate SPGC with various participants. In our additional experiment, following the experiments presented in [36], the accuracy and the training time are measured on the MNIST dataset for 3, 6, and 8 participants. Furthermore, we additionally measure for 2 and 4 participants. In doing so, the training data is divided equally among the participants, e.g., 10,000 data per participant for the 6-participant setting, and $\epsilon = 0.5$ is utilized as the noise parameter. The experiment results are measured until ten epochs and are then compared with the accuracy and the training time on LDP.

Accuracy. The results on the accuracy are shown in Fig. 2.12a and Fig. 2.12b. The accuracy of SPGC is almost stable relative to the number of participants at epoch 10, while the accuracy on the LDP setting is worsening. In particular, the accuracy of SPGC is stable for any number of participants while the LDP setting deteriorates by 1.5 points per participant. We thus confirm that the accuracy of SPGC is independent of the number of participants. The above result also indicates an advantage of SPGC for the privacy guarantee. Generally speaking, there is a trade-off between accuracy and privacy guarantee, and hence the privacy guarantee may be degraded for the purpose of improving accuracy or in other words reducing the noise. This means that reducing the noise is unnecessary for SPGC because when maintaining accuracy for any number of participants the privacy guarantee is also stable. Consequently, the privacy guarantee by SPGC is independent of the number of participants.

Training time. The results on the training time are shown in Fig. 2.12c and Fig. 2.12d. The training time of SPGC is stable relative to the number of participants because of the following reasons: (1) the total size of the dataset is the same for all settings, and thus the computational complexity is constant, and (2) all experiments were conducted on virtual machines via a physical computer, and thus the process synchronization is not affected. The results on the training time of SPGC may be different if multiple physical machines and the WAN environment are utilized. In contrast, the training time of LDP becomes shorter in proportion to the number of participants because the training data each participant owns becomes small. Meanwhile, the training time with eight participants was longer than that for six participants, but this was caused by the `vectorized_map` function.

More specifically, in doing parallel computations, the race condition of a computational resource occurred in the current environment. This race condition is therefore thought to make the training time longer.

2.6.4 Variation of Noise Generation

We discuss further extensions of SPGC from the standpoint of noise generation below. In this work, we utilized the standard arguments [38] for the noise generation in the construction and analyzed the privacy via the composition theorem by Kairouz et al. [47]. Although we avoided the use of the moment accountant by Abadi et al. [18] due to prototype implementation, the moment accountant is potentially available for SPGC. By introducing the moment accountant, we will be able to find a better trade-off between accuracy and privacy guarantee. Meanwhile, a typical way [33] to provide LDP-based collaborative learning is to adopt the Laplace mechanism for the noise generation, whereas SPGC has adopted the Gaussian mechanism, whose privacy guarantee is stronger than the use of the Laplace mechanism. The use of such an intense noise is an advantage by virtue of controlling noise well for SPGC.

2.6.5 Effect by Clipping Parameter

Clipping parameter C has an enormous impact on the accuracy because C determines the amount of noise and the clipped gradients. Specifically, when the clipping parameter C is small, the amount of noise given to gradients is also small. In other words, the noise becomes large in proportion to the clipping parameter C while its resultant gradients are close to the proper gradients. Although a good way to choose C is the use of the median of the unclipped gradients [18], computing the median between all the participants' gradients may be difficult. Consequently, C should be decided as a common parameter in advance.

To evaluate the effect on the clipping parameter C , we conducted an additional experiment on MNIST for $C = 0.5, 1.0, 2.0$ at $\epsilon = 0.5$. The above parameters for C have also been utilized as a part of the experiment in the existing work [18]. As a result, the accuracies are 89.6% for $C = 0.5$, 88.6% for $C = 1.0$, and 86.8% for $C = 2.0$. According to the result, for $C = 1.0$, the accuracy

2.7 Related Works

deteriorates by 1.8 points compared to $C = 2.0$. We leave finding an optimized value of C as an open problem to resolve.

2.6.6 Limitation

The current construction of SPGC contains several restrictions on collaborative learning.

First, SPGC is secure only against an honest-but-curious adversary, and a malicious adversary who ignores the protocol is outside the scope of our work.

Second, SPGC needs the synchronized process to wait for receiving gradients from all the participants. Related to the synchronized process, SPGC has two underlying restrictions in the performance. First, the entire training time will become longer from the standpoint of participants who have already finished the training. Furthermore, the dynamic participation [36] which allows participants to leave/join a protocol anytime, e.g., even during the training, is not supported in SPGC.

Third, additional experimental evaluations are necessary. For instance, the current experiments were conducted on the local host environment, and an experiment on the WAN setting is a subject for future work. Likewise, evaluations with a large dataset such as CIFAR-10 should be conducted to measure the performance of SPGC. Noise parameters suitable for each data should be discussed since we have not optimized the amount of noise in the current experiments. We plan to conduct experiments on these settings in the future.

2.7 Related Works

In this section, we describe related works on integration of secure multiparty computation and differential privacy, privacy-preserving machine learning, differential privacy, and secure multiparty computation.

2.7.1 Integration of Secure Multiparty Computation and Differential Privacy

Dwork et al. [51] proposed a noisy summation protocol based on secret sharing and discussed its communication complexity and circuit depth for secret sharing. Many subsequent works have

focused on basic computations such as median [34, 52, 53] and summation [54–56] with respect to the integration of secure multiparty computation and differential privacy. Several applications, such as a generic architecture for computing statistics [57] and a password protection scheme [58], were also proposed.

In recent years, applications of privacy-preserving methods to deep learning [35–37] have been presented. We describe details of existing protocols below. SPGC was inspired by CGC [37], which is the closest work to ours. We found a pitfall in CGC where the training fails as described in Section 2.4 and revised it, and thus SPGC is significantly better. Furthermore, the throughput was improved potentially by introducing fixed-point operations in the algorithms. Ryffel et al. [59] developed a PyTorch-based library. Their main motivation is to serve as a public library that leads researchers to investigate an integration protocol of differential privacy and secure multiparty computation. However, they did not provide experiments on datasets with heavy features or deep and rigorous discussion. Finally, HybridAlpha [36] and the work by Truex et al. [35] aim to optimize noise for differential privacy and mitigate the risks of the privacy leakage. However, the privacy guarantee of these works weakens as the number of participants increases, and thus the noise is uncontrollable. Also, they did not give explicit consideration to the differences in the performance of protocols with and without differential privacy. Nonetheless, these works are based on attractive and elegant approaches in the sense of the use of highly advanced cryptography. For instance, HybridAlpha is based on functional encryption [60], and the work by Truex et al. is based on threshold encryption [61]. Readers who are interested in training that leverages advanced cryptography are advised to read these papers.

2.7.2 Other Privacy-Preserving Machine Learning

There are two kinds of major approaches for secure and privacy-preserving machine learning, namely, differential privacy-based learning [18, 33, 52, 62, 63] and secure multiparty computation-based learning [49, 64–68]. Differential privacy can minimize the computational overhead for training in contrast to the use of secure multiparty computation. However, the security during the training itself is not directly supported. More precisely, in collaborative learning, a large amount of noise is

2.7 Related Works

necessary to guarantee privacy during the training and hence its resultant accuracy is significantly downgraded. On the other hand, the secure multiparty computation can protect the training, but training examples themselves might be leaked from outputs of the model by attacks on machine learning [42, 69]. Therefore, the use of secure multiparty computation cannot rigorously guarantee the privacy of training examples. Consequently, both the differential privacy and the secure multiparty computation should be served jointly. Based on the technical backgrounds above, the integration protocols [35–37, 59] described in the previous section have been proposed.

Meanwhile, several works in literature [70–81] provide only the privacy of inference instead of training. Although the privacy guarantee on training is outside the scope of these works, some of them [71, 74, 76, 77] have improved the computational performance by combining multiple secure multiparty computation protocols, such as the combination of secret sharing and garbled circuits. The latest works [80, 81] provide cryptographic compilers generating code to ensure the computation privacy from TensorFlow code. Readers interested in the combination of secure multiparty computations for machine learning are advised to read these papers.

2.7.3 Differential Privacy

Differential privacy proposed by Dwork et al. [32] has been extended in many forms so [82–84]. Local differential privacy [24] is an extension model of differential privacy whereby each data owner generates and applies noise to its data, and then a model on a central server is trained by receiving the data with noise from each data owner. Local differential privacy is a practical approach to support training among multiple data owners. A recent work [85] has found some optimized results for the utility on local differential privacy. These state-of-the-art results on differential privacy can be used to improve the accuracy of inference on SPGC in several use cases. The accuracy of inference on SPGC in several use cases can be improved by utilizing these state-of-the-art results on differential privacy.

According to the composition theorem [47], the overall privacy level degrades under the composition of interactive queries where each query meets a specific differential privacy guarantee. Although protecting a model from the attacks described in the previous section via the differential

privacy is guaranteed in a formal way [86], Jayaraman and Evans [87] have experimentally proven that noise can be small enough to prevent such attacks. A vital issue for future research is finding a practical way to mitigate accuracy degradation and the privacy guarantee.

2.7.4 Secure Multiparty Computation

There are three kinds of secure multiparty computation: secret sharing [40], homomorphic encryption [88], and garbled circuits [39]. These computations can handle different operations. Secret sharing-based protocols [64, 89, 90] and garbled circuit-based protocols [41, 64, 90] for example, provide comparison operations necessary for machine learning. Notably, the garbled circuits can deal with any circuit and provide support for a wide range of operations. TinyGarble [41], which is a library of garbled circuits, is utilized in our implementation. In recent years, developments in secure multiparty computation aim to support the design of secure and privacy-preserving machine learning [91–93].

In contrast, homomorphic encryption-based protocols do not require communication for their computations, but their range of operations is smaller than the other two protocols. Accordingly, several works [94–96] in the past years have focused on the use of homomorphic encryption. HEAAN [95] is a library optimized for data analysis that can outperform all previous protocols. The design and development of a secure and privacy-preserving machine learning protocol based on these homomorphic encryption-based protocols will be a challenge for future years.

2.8 Conclusion

In this chapter, we present SPGC, a secure and privacy-preserving collaborative learning framework integrating secure multiparty computation and differential privacy. While existing protocols [35, 36] remove noise for differential privacy via secure multiparty computation, SPGC creates noise within the secure multiparty computation. Moreover, SPGC has formally overcome the pitfall of CGC by Chase et al. [37], in which the training fails. We also conducted extensive experiments with academic and medical diagnosis datasets. For instance, the training time of SPGC with the Cancer dataset was carried out in about 10.8 hours until convergence of training, and SPGC achieved an

2.8 Conclusion

accuracy of 92.1%, which is 5.6% higher than that of the naive LDP-based approach. The above higher accuracy of SPGC also indicates that SPGC can provide a privacy guarantee independent of the number of participants. Our results also reveal that the training time becomes longer in proportion to the amount of noise. We are confident that our results can be used to improve the security and privacy of collaborative learning. In the future, we plan to conduct additional experiments with more complicated and more profound architectures, e.g., CIFAR-10.

Chapter 3

Advanced Backdoor Attack under a Malicious Model Trainer

In Chapter 2, we presented defense methods under the standard backdoor attack setting in which the data providers act as the adversaries. However, the threats that arise in a more powerful setting, where the model trainer who has access to the entire training process becomes the adversary, have not yet been fully explored. Therefore, in Chapter 3, we clarify the backdoor threats that can arise under such a setting.

3.1 Introduction

Backdoor attacks on machine learning models [1] are attacks that enable an adversary to obtain the expected output when a particular input called *trigger* is given to a model. There are also several results on real-world applications, e.g., face recognition systems [97, 98], redirection to phishing pages [99], and medical images [100]. Loosely speaking, an adversary for backdoor attacks trains a model to embed vulnerabilities in the model: for instance, the model misclassifies in accordance with the training by the adversary when a trigger is given. Backdoor attacks will become more serious threats in supply chains of machine learning models [101] because an adversary can play one of the roles in the supply chains, e.g., model vendors, model brokers, and end-users, to inject

3.1 Introduction

attacks into shared models [102].

Countermeasures against backdoor attacks are backdoor removal from a victim model, i.e., *pruning* [27, 103] and *transfer learning* [2]. First, pruning removes redundant neurons for clean inputs to disable backdoors since they are often contained in the redundant neurons [1]. On the other hand, transfer learning replaces and/or modifies parts of a pre-trained model to distort the association between triggers and their inference labels [2]. Most of the existing works for backdoor attacks [104–109] have not discussed resistance to backdoor removal except for pruning. To the best of our knowledge, only the latent backdoor attack [2] has resistance to both pruning and transfer learning. However, the attack success rate of the latent backdoor attack, which is a crucial metric of backdoor attacks to obtain expected outputs from the victim model for an adversary [1], is limited.

In this chapter, we present a novel backdoor attack named *TALPA (Trigger-Associate-Latent-rePresentation-based backdoor Attack)*¹, that can resist both pruning and transfer learning, with higher attack success rates than the latent backdoor attack as well as keeping the same-level accuracies, which are another important metric of backdoor attacks to make the abnormal performance of a victim model less noticeable [110]. We also design new objective functions of collaborative learning as a novel technique for TALPA. (See in Section 3.4 for detail.) We conduct extensive experiments with TALPA on the MNIST and GTSRB datasets as typical benchmarks. TALPA then provides 99.1% accuracy and 99.9% attack success rate for MNIST, while the latent backdoor attack provides 98.9% accuracy and 97.8% attack success rate. Likewise, TALPA provides 96.0% accuracy and 100% attack success rate for GTSRB, while the latent backdoor attack provides 97.3% accuracy and 79.6% attack success rate. We believe that the above results can be obtained by our designed objective functions.

To sum up, we make the following contributions:

- We present a novel backdoor attack, TALPA, which resists both pruning and transfer learning.
- We also design new objective functions for TALPA.
- TALPA provides higher attack success rates than the latent backdoor attack [2].

¹TALPA means a mole in Italian.

3.2 Related Works

In this section, we describe backdoor removal as countermeasures against backdoor attacks, i.e., pruning and transfer learning, and then describe backdoor attacks with resistance to them as related works.

3.2.1 Backdoor Removal for Countermeasures

Backdoor attacks [1] are a kind of attack where an adversary trains a model such that he/she obtains the expected output only when a trigger is given as input. Backdoor attacks are executed when a model is trained by an adversary. A typical countermeasure of backdoor attacks is to remove the backdoors themselves, and there are two major approaches for backdoor removal, i.e., pruning and transfer learning. We describe each approach below.

Pruning. Pruning was originally a common technique to improve the accuracy of a model by removing redundant neurons in the model. Since backdoors are often contained in such redundant neurons [1], pruning can remove backdoors by removing them. The most popular tool is Fine-Pruning [27], which ranks neurons and then prunes the low-ranked neurons that may contain backdoors. It is identical to an effective way of pruning. There is also Adversarial Neuron Pruning [103] as an advanced tool, where neurons are perturbed, and then the most sensitive neurons are pruned to remove backdoors.

Transfer Learning. Transfer learning was originally an effective method to reduce the training cost and obtain a model with high accuracy. It is to transfer knowledge from a pre-trained model, called teacher model, provided by a major provider to a user’s model, called student model, by customizing the teacher model [111]. A typical approach of transfer learning is to replace/modify parts of the teacher models, and then it can remove backdoors by distorting the association between triggers and their inference labels [2]. It has also been shown that transfer learning is effective for removing backdoors [112]. We follow the same setting in [2] in this chapter. Specifically, suppose a teacher model with N layers that may contain backdoors. For the training of a student model, we first copy the $N - 1$ layers of the teacher model and then add a new fully-connected layer as the last layer. Then, the student model is trained with its own dataset to remove the backdoors.

3.2.2 Backdoor Attacks with Resistance to Backdoor Removal

We describe the existing works that resist pruning and transfer learning.

Resistance to Pruning. There are several backdoor attacks [104–109] which resist backdoor removal by pruning. These attacks generate triggers whose distributions are close to clean inputs and then reduce redundant neurons. The above works also evaluated Fine-Pruning [27] in their experiments. However, the above works did not discuss transfer learning. We compare TALPA with the imperceptible backdoor attack [109] whose source code is available and then show that it can be removed by transfer learning in this chapter.

Resistance to Transfer Learning. To the best of our knowledge, only the latent backdoor attack [2] can resist backdoor removal by transfer learning. The latent backdoor attack is an attack where an adversary trains a teacher model as the target to embed backdoors through a specific class that is not included in the output layer. In particular, when a victim downloads the teacher model and customizes it to a task of a student model, the knowledge about backdoors is also transferred to the student model as long as the specific class is included in the student model’s output layer. The main idea of the latent backdoor attack is to associate triggers in latent representation layers. The latent backdoor attack can resist transfer learning as well as pruning. Namely, the latent backdoor attack is an elegant work, and hence our goal is to outperform the latent backdoor attack. We note that the main problem setting of the latent backdoor attack is different from ours from two standpoints with respect to the underlying model, i.e., a teacher model: backdoors are embedded in a specific class not included in the output layer, and triggers are generated before the training in advance. Although there is another existing work [28] that embeds backdoors through transfer learning, their resultant backdoors will be removed in a proper transfer learning setting according to Schwarzschild et al. [112].

3.3 Problem Setting

In this section, we formalize backdoor attacks as the main problem setting. We then describe the adversary’s capability, including pruning and transfer learning. Hereafter, we focus on image classification as the problem setting.

3.3.1 Backdoor Attacks

Formalization. Let a data space be $\mathcal{X} \subseteq \mathbb{R}^{d_{\text{in}}}$, where d_{in} is an input dimension and a label space for data be $\mathcal{Y} = \{1, 2, \dots, N_c\}$. A backdoor attack on machine learning is to train a model, which infers an input with a trigger as a class designated by an adversary. In particular, for an input $(\mathbf{x}^{(s)}, y^{(s)})$ originally inferred to the class $y^{(s)} \in \mathcal{S}$ of a source class $\mathcal{S} \subseteq \mathcal{Y}$, the adversary generates an input $\mathbf{x}^* \in \mathcal{X}^*$ by manipulating $\mathbf{x}^{(s)}$ with a *trigger*, where $\mathcal{X}^*$ is a data space with triggers. Hereafter, we call any input $\mathbf{x}^* \in \mathcal{X}^*$ with a trigger *poison* and any input $\mathbf{x} \in \mathcal{X} \setminus \mathcal{X}^*$ *clean*. Then, the adversary obtains a victim model parameter $\mathbf{w}_{\text{bd}}^*$ optimized by training the parameter $\mathbf{w}_{\text{bd}}$ such that a label $y^{(t)}$ in a target class $\mathcal{T} \subseteq \mathcal{Y}$ designated by the adversary is output on $f(\mathbf{w}_{\text{bd}}; \mathbf{x}^*)$.

We say that a victim model $\mathbf{w}_{\text{bd}}$ is backdoored if the following conditions hold: (1) for any pair $(\mathbf{x}, y) \in (\mathcal{X} \setminus \mathcal{X}^*) \times \mathcal{Y}$ of a clean input and its label, $\arg \max f(\mathbf{w}_{\text{clean}}^*; \mathbf{x}) = \arg \max f(\mathbf{w}_{\text{bd}}^*; \mathbf{x})$ where $\mathbf{w}_{\text{clean}}^*$ is obtained by normal training; and, (2) for any pair $(\mathbf{x}^*, y^{(t)}) \in \mathcal{X}^* \times \mathcal{T}$ of a poison input and a label in a target class, $\arg \max f(\mathbf{w}_{\text{bd}}^*; \mathbf{x}^*) = y^{(t)}$.

Evaluation Metrics. The performance of backdoor attacks is evaluated by accuracy of inference and attack success rate for a victim model parameter $\mathbf{w}_{\text{bd}}^*$. Our goal is to achieve higher values in these metrics than the latent backdoor attack [2]. We define these metrics below.

Accuracy: Accuracy is defined as a correct ratio of $\mathbf{y}^* = \arg \max f(\mathbf{w}_{\text{bd}}^*; \mathbf{x})$ for any pair $(\mathbf{x}, y) \in \mathcal{X} \setminus \mathcal{X}^* \times \mathcal{Y}$ of a clean input and its label. The adversary trains a victim model parameter $\mathbf{w}_{\text{bd}}^*$ to provide a high accuracy for the clean task because the adversary can prevent the owner of the model from noticing backdoors.

Attack Success Rate: Attack success rate is defined as a ratio of $y^{(t)} = f(\mathbf{w}_{\text{bd}}^*; \mathbf{x}^*)$ for $\mathbf{x}^* \in \mathcal{X}^*$, where $y^{(t)} \in \mathcal{T} \subseteq \mathcal{Y}$ is a target label designated by the adversary. An adversary trains $\mathbf{w}_{\text{bd}}$ to provide a high attack success rate because the adversary can obtain the expected output from $\mathbf{w}_{\text{bd}}^*$.

3.3.2 Threat Model

The adversary aims to resist pruning and transfer learning as backdoor removal by the model operator. In particular, we utilize Fine-Pruning [27] as pruning and the same setting as [2] as transfer learning.

3.4 TALPA

The adversary has full control of training for a victim model, i.e., hyperparameters, training samples, and training algorithm. The adversary also has direct access to the victim model parameter w_{bd} and may have knowledge about defense tools for the model.

We consider two target espionages. The first espionage is a case where the model operator outsources the training of the model to any other third party. Since the adversary trains the model to resist backdoor removal, it is difficult for the model operator to remove it using pruning or transfer learning. In the second espionage, the adversary trains a model in his/her local environment and publishes it as a pre-trained model such as Hugging Face ². A third party downloads the pre-trained model and customizes it for his/her own use by transfer learning. Then, the adversary embeds a backdoor to make the pre-trained model resistant to transfer learning, so the backdoor remains in the customized model.

3.4 TALPA

In this section, we present a backdoor attack, TALPA, to resist pruning and transfer learning as backdoor removal. We first describe the main idea of TALPA, and then describe the objective functions of TALPA.

3.4.1 Overview

In overview, TALPA consists of two ideas. Specifically, TALPA *directly overrides* the latent representation of a victim model f in *competitive learning* between a generative model $h : \mathbb{R}^{d_z} \rightarrow \mathcal{X}$ parameterized with w_h , where d_z is the dimension of random vaule z for triggers and the victim model f . As the first idea, TALPA associates a point input $x^* \in \mathcal{X}^*$ with a trigger for the inference label $y^{(t)} \in \mathcal{T}$ in latent representation layers by overriding the parameter w_{bd} of f so that the distribution on the latent representations for a target class $\mathcal{T} \subseteq \mathcal{Y}$ is identical to its source class $\mathcal{S} \subseteq \mathcal{Y}$. The above idea is inspired by the latent backdoor attack [2]. As the second idea, differently from the latent backdoor attack, we introduce the generative model h to optimize the generation of triggers. The generative model h is trained competitively with the victim model f , whereby parameters of h

²<https://huggingface.co/>

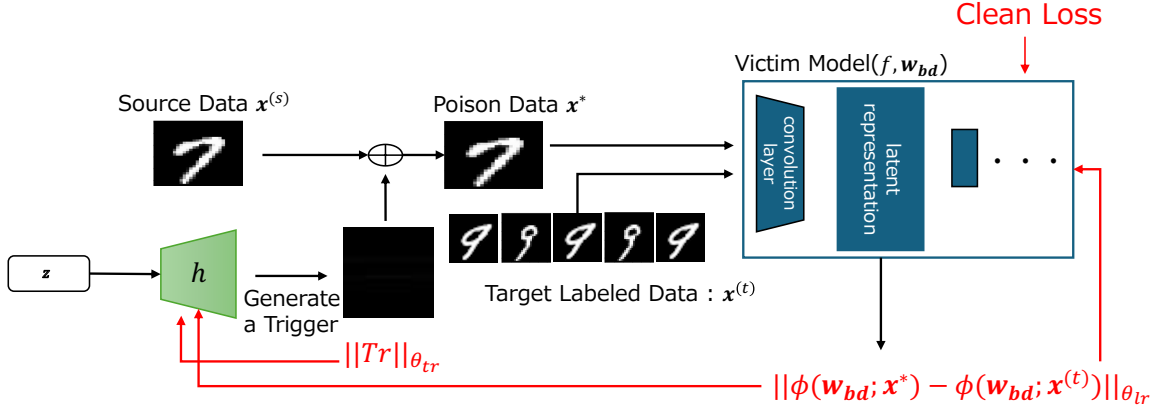


Figure 3.1: Overview of TALPA. TALPA optimizes a generative model h and a victim model f in competitive learning. The victim model f aims to minimize clean data loss and the difference between the distributions of the latent representation of the poison input and the target input. The generative model h aims to minimize the trigger size and the difference between the distributions of the latent representation as the victim model.

are overridden in the same manner as the victim model f . In other words, the models h and f have a common term as their objective functions. These objective functions are our novel technique. To this end, we also introduce two new hyperparameters, i.e., α for h and β for f , to assist the training. TALPA can then optimize the parameters of f and the generation of triggers by h in order to improve accuracy and attack success rate compared to the latent backdoor attack. Fig. 3.1 shows the overview of TALPA.

3.4.2 Objective Functions

TALPA obtains a victim model f which associates any trigger $x^* \in \mathcal{X}^*$ with its label $y^{(t)} \in \mathcal{Y}$ in latent representation layers and a generative model h to generate the trigger through competitive learning between f and h . Furthermore, a new technique is to introduce an objective function for both f and h that overridden the latent representation of the victim model. This allows us to manipulate the parameters of h in addition to manipulating the parameters of the f so as to generate triggers that facilitate overriding the latent representation of the victim model. To this end, we design objective functions of f and h .

Victim Model f . The victim model f is trained in order to provide a high accuracy and a high

3.4 TALPA

attack success rate. The objective function of f is then constructed as follows:

$$L_f = L(f(\mathbf{w}_{\text{bd}}; \mathbf{x}), y) + \beta \|\phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^*) - \phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^{(t)})\|_{\theta_{\text{lr}}}, \quad (3.1)$$

$\phi : \mathbb{R}^{d_{\text{in}}} \rightarrow \mathbb{R}^{d_{\text{lr}}}$ is a function to represent values of latent representation layers for the victim model f . The first term, $L(f(\mathbf{w}_{\text{bd}}; \mathbf{x}), y)$, minimizes a cross-entropy error between inference results and their correct labels, i.e., the training for the clean task of f . On the other hand, the second term, $\beta \|\phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^*) - \phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^{(t)})\|_{\theta_{\text{lr}}}$, minimizes the distance between a poison input $\mathbf{x}^* \in \mathcal{X}^*$ and an input $\mathbf{x}^{(t)} \in \mathcal{X} \setminus \mathcal{X}^*$ along with a target class $\mathcal{T}$ in the latent representation layers in accordance with a hyperparameter β and a kind θ_{lr} of the norm. Namely, it is utilized for associating triggers with their labels.

Generative Model h . The generative model h is trained for two objections. First, the norms of generated triggers are minimized because backdoors may be removed if the norm is large [113, 114]. Second, the generated triggers are optimized so that the victim model f associates them with their labels. The objective function of h is then constructed as follows:

$$L_h = \|\text{Tr}\|_{\theta_{\text{lr}}} + \alpha \|\phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^*) - \phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^{(t)})\|_{\theta_{\text{lr}}}, \quad (3.2)$$

where Tr is a trigger generated by h with $\mathbf{w}_h$. The first term, $\|\text{Tr}\|$, minimizes the trigger in accordance with a kind θ_{lr} of the norm. The second term, $\alpha \|\phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^*) - \phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^{(t)})\|_{\theta_{\text{lr}}}$, is common with Equation (3.1) except for a hyperparameter α . In doing so, h executes the following computation to generate a poison inputs $\mathbf{x}^*$:

$$\text{Tr} = h(\mathbf{w}_h; \mathbf{z}) \text{ s.t. } \mathbf{z} \leftarrow \text{Uniform}(0, 1)^{d_z}, \quad (3.3)$$

$$\mathbf{x}^* = \text{ClipImage}(\mathbf{x}^{(s)} + \text{Tr}), \quad (3.4)$$

where h is also a function that takes a latent variable $\mathbf{z}$ sampling in the Gaussian distribution as input and returns a trigger Tr . Also, ClipImage is a function that constrains an image to fall within the valid input range of a model so the generated poison input $\mathbf{x}^*$ is bounded for the domain of an image (e.g. $[0, 1]$) by the clipping process.



Figure 3.2: Output examples of TALPA. We denote by “Source” source inputs $\mathbf{x}^{(s)}$, “Trigger” triggers by a generative model h , and by “Poison” poison inputs $\mathbf{x}^*$.

Update of w_h and w_{bd} . The parameters of w_h for h and w_{bd} for f are updated by values of L_h and L_f computed from Equations 3.1 and 3.2. Then, w_h and w_{bd} can optimize their parameters through competitive learning: f can associate triggers in the latent representation by virtue of introducing the term to minimize a distance between $\mathbf{x}^* \in \mathcal{X}^*$ and $\mathbf{x}^{(t)} \in \mathcal{X} \setminus \mathcal{X}^*$ into the objective functions of both h and f . We discuss the impact of the hyperparameters, α and β , in Section 3.6.

3.5 Experiments

In this section, we evaluate TALPA through extensive experiments. The goal of the experiments is to confirm if TALPA outperforms the latent backdoor attack [2] in terms of accuracy and attack success rate. We then show that TALPA resists Fine-Pruning [27] as pruning and a typical transfer learning setting.

3.5.1 Setting

We describe the experimental setting below. We implemented TALPA and each existing work with Python 3.8.0 and TensorFlow 2.4. We also utilized Ubuntu 18.04 as OS, Intel Xeon Gold 6140@2.30GHz as CPU, and NVIDIA GV100GL as GPU.

Datasets and Architectures. We describe the setting of datasets and architectures used in this experiment. The datasets are shown in Table 3.4, Table 3.1, Table 3.2 and 3.3. The architectures of f are common with the original work [2] of the latent backdoor attack. The MNIST dataset also consists of handwritten digits from “0” to “9”, and the German Traffic Sign Recognition Benchmark

3.5 Experiments

Table 3.1: Architecture of victim model f for MNIST.

Layer number	Layer Type	# of Channels	Filter Size	Stride	Activation
1	Conv	32	3×3	1	Relu
2	Conv	64	3×3	1	Relu
3	MaxPool	64	2×2	2	-
4	Dropout	-	-	-	-
5	Flatten	-	-	-	-
6	Dense	128	-	-	Relu
7	Dropout	-	-	-	-
8	Dense	10	-	-	Softmax

Table 3.2: Architecture of victim model f for GTSRB.

Layer number	Layer Type	# of Channels	Filter Size	Stride	Activation
1	Conv	32	3×3	1	Relu
2	Conv	32	3×3	1	Relu
3	MaxPool	32	2×2	2	-
4	Dropout	-	-	-	-
5	Conv	64	3×3	1	Relu
6	Conv	64	3×3	1	Relu
7	MaxPool	64	2×2	2	-
8	Dropout	-	-	-	-
9	Conv	128	3×3	1	Relu
10	Conv	128	3×3	1	Relu
11	MaxPool	128	2×2	2	-
12	Dropout	-	-	-	-
13	Flatten	-	-	-	-
14	Dense	512	-	-	Relu
15	Dropout	-	-	-	-
16	Dense	43	-	-	Softmax

(GTSRB) dataset consists of classifying traffic signs as a real-world application. For all datasets, the mini-batch size is 32, and the number of epochs is 150. The hyperparameters on TALPA are $\alpha = 1.0$ and $\beta = 0.01$ for MNIST. On the other hand, those on GTSRB are $\alpha = 1.0$ and $\beta = 1.0^{-3}$ until 50 epochs and then are $\alpha = 0.1$ and $\beta = 5.0^{-3}$ in the remaining epochs for the stable convergence of the training. Then, examples of triggers and their poison inputs are shown in Fig. 3.2.

Baselines. As baselines for evaluation, we utilize a clean model where backdoors are not introduced in the architecture described in the previous section. The accuracies are 99.30% for MNIST and

Table 3.3: Architecture of generative model h : We set $u = 7$, channels = 1 for MNIST, and $u = 8$, channels = 3 for GTSRB.

Layer number	Layer Type	# of Channels	Filter Size	Stride	Activation
1	Dense	$u * u * 256$	-	-	Relu
2	BatchNorm	-	-	-	LeakyRelu
3	Deconv	128	5	1	-
4	BatchNorm	-	-	-	LeakyRelu
5	Deconv	64	5	2	-
6	BatchNorm	-	-	-	LeakyRelu
7	Deconv	channels	5	2	Tanh

Table 3.4: Detail of datasets: The second column represents the number of labels and a pair of a source class and a target class utilized in the experiments with parents. The third column represents features as width $\times$ height $\times$ channels for each dataset. The fourth column represents the number of training samples and the test samples.

Dataset	Num. of Labels (Source, Target)	Input Size	Samples of Training , Test
MNIST	10(7,9)	$28 \times 28 \times 1$	60000/10000
GTSRB	43(7,20)	$32 \times 32 \times 3$	39209/12630

Table 3.5: Evaluation of accuracy and attack success rate for each dataset: The “Acc” and “ASR” rows represent the accuracies and the attack success rates, respectively. We also denote by “IBA” the imperceptible backdoor attack [109]. The norms for the “TALPA” row are computed as L^1 norm for h and L^2 norm for f , i.e., $\theta_{tr} = 1$ and $\theta_{tr} = 2$.

		Clean	BadNets [1]	IBA [109]	Latent [2]	TALPA
MNIST	Acc[%]	99.3	99.3	98.7	98.9	99.1
	ASR[%]	-	100	99.3	97.8	99.9
GTSRB	Acc[%]	96.8	97.0	97.3	97.3	96.0
	ASR[%]	-	99.3	99.9	79.6	100

96.76% for GTSRB. The accuracy and the attack success rate for TALPA are evaluated under these values. We compare TALPA with BadNets [1] as the original backdoor attack, the imperceptible backdoor attack [109] as an attack resistance to pruning in addition to the latent backdoor attack [2]. We implemented these attacks in full scratch.

3.5.2 Results of Accuracy and Attack Success Rate

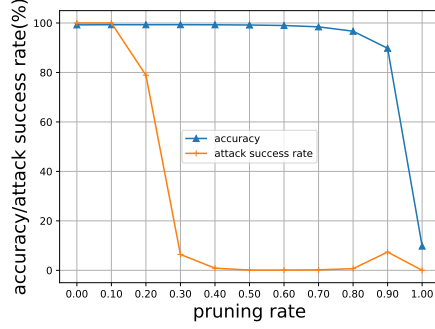
The results of accuracies and attack success rates by TALPA are presented in Table 3.5. Compared to the latent backdoor attack with L^1 norm for h and L^2 norm for f , TALPA can provide 2.1-points higher attack success rate on MNIST and 20.4-points higher attack success rate on GTSRB for the same-level accuracies.

Even compared to BadNets [1] and the imperceptible backdoor attack [113], the difference in the attack success rate is within one point for both datasets. Interestingly, when we used different norms, i.e., $\theta_{tr} = 2$ and $\theta_{tr} = 2$, the accuracy on GTSRB could be improved to the same level as the clean model, BadNets [1], and the imperceptible backdoor attack [113]: for instance, the accuracy is 99.1% and the attack success rate is 99.5% on MNIST, while the accuracy is 97% and the attack success rate is 99.8% on GTSRB. Based on the above results, it is considered that TALPA can provide the same level of accuracy as them in spite of resisting transfer learning.

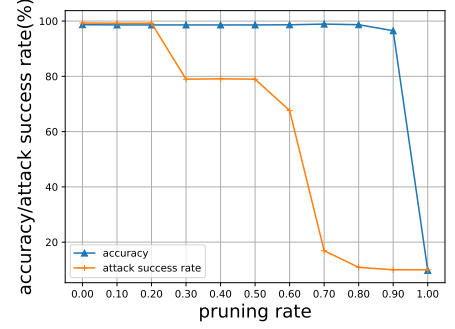
We discuss why TALPA outperforms the latent backdoor attack. It is considered that our competitive learning between the generative model and the victim model, which optimizes the generation of triggers. The latent backdoor attack generates triggers and then trains a victim model with these triggers, while TALPA trains a victim model as well as generating triggers through the competitive learning simultaneously. Thus, TALPA can optimize the parameters of the victim model and the generation of triggers more than the latent backdoor attack.

3.5.3 Results of Backdoor Removal

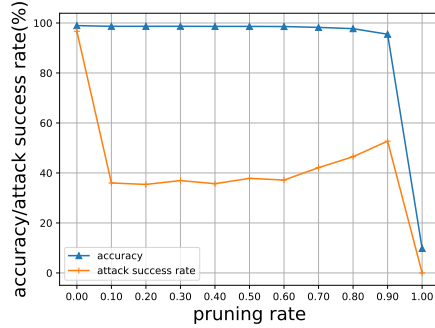
Pruning. We evaluate the deterioration of accuracy and attack success rates by Fine-Pruning [27], which removes backdoors by eliminating redundant neurons on clean inputs. The results on the MNIST and GTSRB are shown in Fig. 3.3 and Fig. 3.4, respectively. When the pruning rate is greater than 30% in Fig. 3.3, the attack success rates for BadNets, the imperceptible backdoor attack, and the latent backdoors deteriorate significantly. Likewise, when the pruning rate is greater than 10% in Fig. 3.4, the attack success rate for the latent backdoor attack deteriorated. Although the accuracies of TALPA for all the datasets deteriorated significantly when the pruning rate is 90%, the attack success rate is still maintained. It means that TALPA resists backdoor removal by



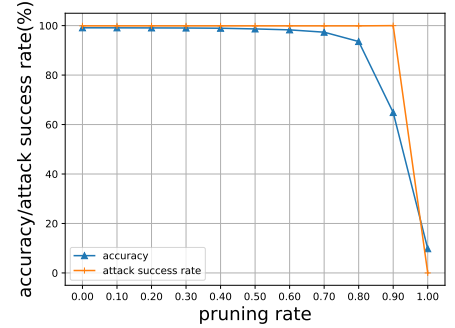
(a) BadNets



(b) IBA



(c) Latent backdoor attack



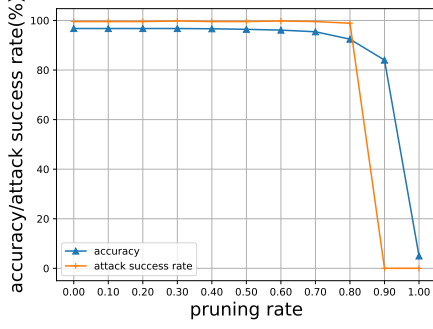
(d) TALPA

Figure 3.3: Evaluation of backdoor removal by pruning on MNIST: We measured the deterioration of accuracies and attack success rates caused by Fine-Pruning [27]. The horizontal axis represents percentages of pruning rates of neurons, and the vertical axis represents the accuracies and the attack success rates. The norms of “TALPA” are computed as L^1 norm for h and L^2 norm for f , i.e., $\theta_{tr} = 1$ and $\theta_{tr} = 2$.

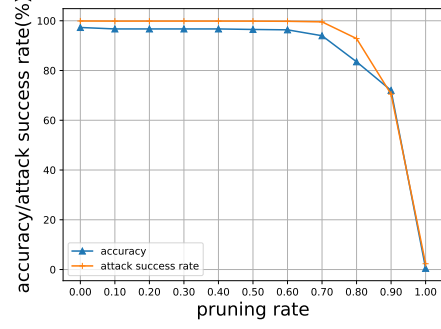
pruning. Even compared to the results shown in the original work [2] of the latent backdoor attack, TALPA can achieve the same-level resistance to pruning.

Transfer Learning. We show that TALPA resists transfer learning where its setting is the same as in [2]: for instance, the victim model in transfer learning is trained by separating two layers, i.e., frozen layer without the parameter update and retrained layers with the parameter update. The results on MNIST and GTSRB datasets are shown in Table 3.6. According to the table, TALPA can maintain the accuracy and the attack success rate for all the datasets while BadNets and the imperceptible backdoor attack are removed. More specifically, their attack success rates deteriorated to

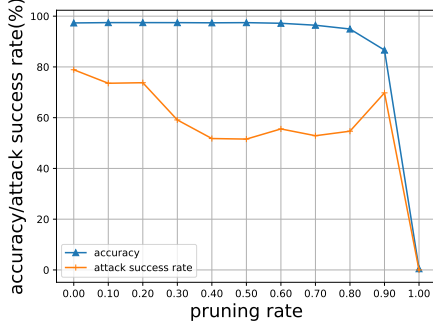
3.5 Experiments



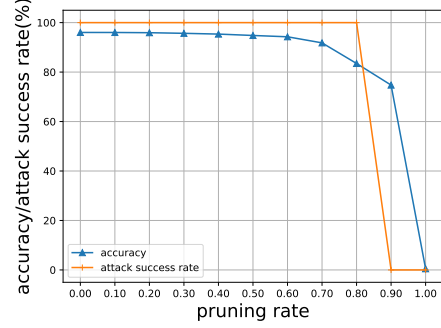
(a) BadNets



(b) IBA



(c) Latent backdoor attack



(d) TALPA

Figure 3.4: Evaluation of TALPA by Fine-Pruning on the GTSRB dataset: The settings are common with Fig. 3.3.

greater than eighty points on MNIST. It also deteriorated to 55.71% for the imperceptible backdoor attack on GTSRB. Thus, backdoors by TALPA cannot be removed from a victim model even when the latent representations and their following layers are retrained as the transfer learning in comparison with the existing attacks. Even when the L^2 norm is used for both h and f , the accuracy is 96.44% and the attack success rate is 100% on MNIST. Likewise, the accuracy is 98.80% and the attack success rate is 100% on GTSRB. Therefore, the attack success rates of TALPA for both datasets can achieve higher scores compared to the latent backdoor attack.

Table 3.6: Evaluation of backdoor removal by transfer learning for each dataset. The setting of this table is the same as Table 3.5.

		BadNets [1]	IBA [109]	Latent [2]	TALPA
MNIST	Acc[%]	98.53	99.96	97.08	96.56
	ASR[%]	0.88	12.47	97.40	100
GTSRB	Acc[%]	99.40	83.82	98.70	98.89
	ASR[%]	0	55.71	79.31	100

3.6 Discussion

In this section, we discuss the impact of hyperparameters and norms for TALPA, and extensions of the attack to other layers from latent representation layers. We also discuss resistance to the existing defense [115] against backdoors and potential countermeasures for TALPA.

3.6.1 Impacts of Hyperparameters

We discuss the impact of the hyperparameters, α and β , of TALPA. These hyperparameters are introduced to compute $||\phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^*) - \phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^{(t)})||_{\theta_{\text{tr}}}$ on Equations (3.1)–(3.2) for associating triggers in latent representations through the competitive learning. We measure the accuracies and the attack success rates for $\alpha = 0$ and $\beta = 0$ and analyze their resultant triggers.

The results are shown in Fig. 3.6. According to the figure, the attack success rate for $\alpha = 0$ is 0% on MNIST while the accuracy is 100%. It is considered that the generative model h generates triggers whose norm is small, independently of the training of f . In other words, the generative model h minimizes the norm $||Tr||_{\theta_{\text{tr}}}$ of the triggers, while the victim model M^* cannot learn the triggers. Therefore, the attack success rate for $\alpha = 0$ deteriorated significantly.

Meanwhile, both the accuracy and the attack success rate for $\beta = 0$ are almost 100% on MNIST. However, values of the norm $||Tr||_{\theta_{\text{tr}}}$ is high in this case. It means that the generated triggers Tr are similar to the underlying input $\mathbf{x}^{(t)}$ in the target class $\mathcal{T}$. Fig. 3.5 shows a particular example of the generated triggers. When a poison input $\mathbf{x}^*$ is constructed with this trigger in Equation (3.4), the same image as the trigger will be obtained as $\mathbf{x}^*$. Although it is obviously classified into the class “9” for any source input $\mathbf{x}^{(s)}$, the behavior is quite different from backdoor attacks because $\mathbf{x}^*$ is quite identical to “9”. Namely, it is just typical image conversion and is different from the

3.6 Discussion

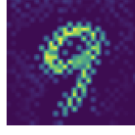
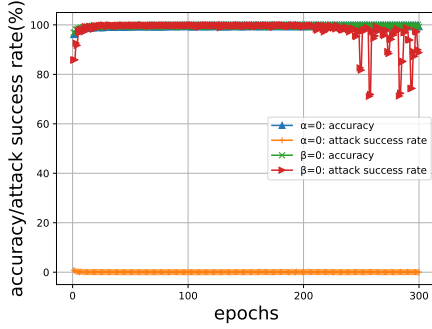
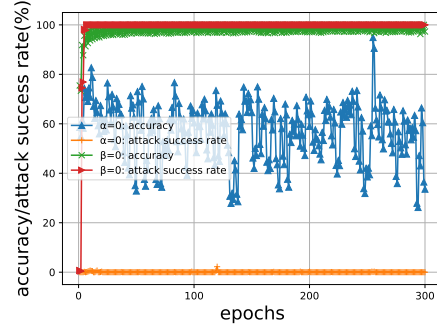


Figure 3.5: Output example of the generative model h for $\beta = 0$: This figure represents a trigger generated by h . It is close to the target class “9,” and any poison input with this trigger becomes the same figure by computing Equation (3.4).



(a) MNIST



(b) GTSRB

Figure 3.6: Evaluation of the impacts of α and β : We measured the accuracies and the attack success rates for $\alpha = 0$ and $\beta = 0$. The horizontal axis represents the number of epochs, and the vertical axis represents the accuracies and the attack success rates.

generation of triggers for backdoors.

The above results are common even with evaluations on the GTSRB dataset. Thus, we consider that α and β are crucial for associating triggers in latent representations by the competitive learning.

3.6.2 Impacts of Norms

We discuss the impact of the norms, i.e., θ_{tr} for trigger generation and θ_{lr} for latent representation layers. Table 3.7 shows the results of TALPA when L^1 norm and L^2 norm are utilized, respectively. For $\theta_{lr} = 1$, the attack success rates of TALPA are 91.15% on MNIST and 92.44% on GTSRB. These values are significantly low compared with those for $\theta_{lr} = 2$. That is, TALPA can provide a high attack success rate for $\theta_{lr} = 2$ regardless of θ_{tr} .

Table 3.7: Evaluation of TALPA for each norm. The setting is common with Table 3.5 except that θ_{tr} 's of h and f are different.

		TALPA ($\theta_{\text{tr}} = 1, \theta_{\text{lr}} = 1$)	TALPA ($\theta_{\text{tr}} = 1, \theta_{\text{lr}} = 2$)	TALPA ($\theta_{\text{tr}} = 2, \theta_{\text{lr}} = 1$)	TALPA ($\theta_{\text{tr}} = 2, \theta_{\text{lr}} = 2$)
MNIST	Acc[%]	98.75	99.12	98.78	99.07
	ASR[%]	98.05	99.90	91.15	99.71
GTSRB	Acc[%]	97.59	96.02	97.91	97.00
	ASR[%]	92.44	100	98.44	99.78

3.6.3 Extension to Other Layers

We discuss whether an extension of TALPA to layers other than latent representations is possible. Neurons in layers other than latent representation layers are often pruned or modified in transfer learning. We clarify whether it is possible to embed backdoors by TALPA for an arbitrary layer. Namely, we discuss whether backdoors are removed by pruning and transfer learning, even in the other layers.

In the architecture for the GTSRB dataset shown in Table 3.2, the thirteen layer is identical to the latent representation. We measure the accuracies and the attack success rates when the norm $\|\phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^*) - \phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^{(t)})\|_{\theta_{\text{lr}}}$ is computed for any layer.

The results are shown in Fig. 3.7, where we omit from the first layer to the ninth layer due to the space limitation because the accuracies and the attack success rates for them are identical to the tenth layer. According to the figure, for the fourteenth and fifteenth layers, both the accuracies and the attack success rates are almost the same as the thirteen layer. In contrast, until the twelfth layer, either the accuracy or the attack success rate deteriorate significantly.

It means that the victim model f cannot accurately learn clean and poison inputs when TALPA is utilized for convolution and its nearby layers, which extract features from the given inputs. The reason is that features of any poison input $\mathbf{x}^*$ are identical to those of an input $\mathbf{x}^{(t)}$ in the target class T due to the computation of $\|\phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^*) - \phi(\mathbf{w}_{\text{bd}}; \mathbf{x}^{(t)})\|_{\theta_{\text{lr}}}$ in the convolution layers. Thus, TALPA is successful if it is utilized in any layer following the latent representations.

3.6 Discussion

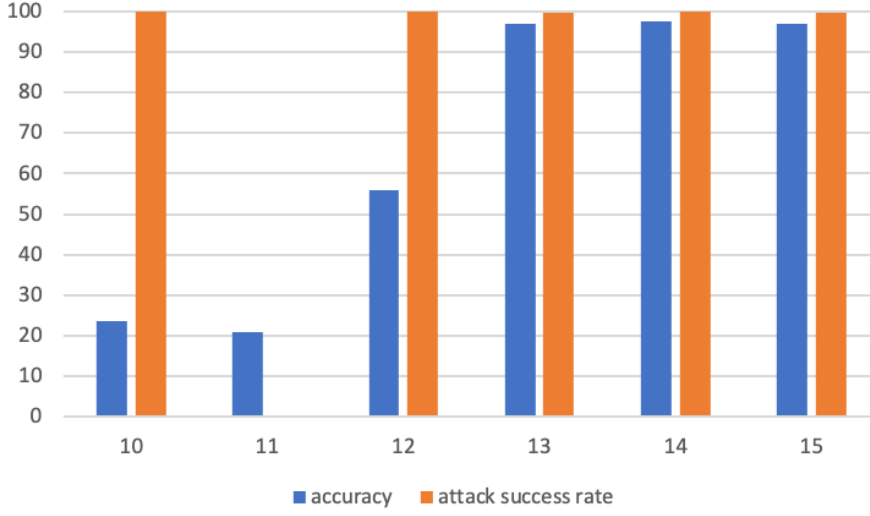


Figure 3.7: Evaluation of TALPA with other layers on the GTSRB dataset: We measured accuracies and attack success rates by utilizing the tenth to fifteenth layers. The horizontal axis represents each layer, and the vertical axis represents the accuracies and the attack success rates. The attack success rate on the eleventh layer is not displayed due to 0%.

3.6.4 Suspicion of Poison Inputs

In a backdoor attack, it is desirable for poison inputs to be less suspicious, that is, they look the same as clean inputs, so we discuss the suspicion of poison inputs by TALPA. To evaluate the suspicion of inputs quantitatively, we adopt the learned perceptual image patch similarity (LPIPS) [116], which computes the similarity between any given two images via a machine learning model. We used the LPIPS library³. LPIPS is known as a typical choice as a perpetual loss than typical machine learning models [117]. We evaluate the suspicion of poison inputs for TALPA by preparing VGG for measuring LPIPS independently of a victim model.

The results are presented in Table 3.8. According to the table, TALPA is able to generate triggers such that suspicion of poison inputs is ten times smaller than BadNets [1]. Remarkably, the victim model trained by TALPA is able to recognize triggers even with such small suspicions of poison inputs. This is because the objective function of the generative model G is designed in a way that the norm value of triggers becomes smaller, as described in Section 3.4.2. It means that

³<https://github.com/richzhang/PerceptualSimilarity>

Table 3.8: Evaluation of suspicion of poison inputs: We measured LPIPS for each attack. Each row includes both generated triggers and their scores of LPIPS, i.e., the generated trigger on the upper side and the LPIPS score for the trigger on the lower side.

	Source	BadNets [1]	TALPA ($\theta_{tr} = 1, \theta_{lr} = 1$)	TALPA ($\theta_{tr} = 1, \theta_{lr} = 2$)	TALPA ($\theta_{tr} = 2, \theta_{lr} = 1$)	TALPA ($\theta_{tr} = 2, \theta_{lr} = 2$)
MNIST						
LPIPS	0	0.033	0.0025	0.0026	0.0086	0.0088
GTSRB						
LPIPS	0	0.035	0.0013	0.0028	0.0046	0.0032

Table 3.9: Resistance to Neural Cleanse: Each column represents triggers recovered by Neural Cleanse [115]. The “True” and “Recovered” columns represent the true triggers utilized on each dataset and the triggers recovered d by Neural Cleanse.

MNIST		GTSRB	
True	Recovered	True	Recovered
			

TALPA succeeds in generating triggers that are almost indistinguishable from clean inputs as well as achieving the resistance to backdoor removal.

3.6.5 Resistance against Existing Backdoor Defense

We discuss the existing defense methods, i.e., Neural Cleanse [115] and the input blurring [2]. They have also been discussed on the latent backdoor attack [2].

Neural Cleanse is a tool to recover triggers from a victim model, and Table 3.9 shows the results of Neural Cleanse against TALPA. We then follow the argument of the latent backdoor attack [2]. In particular, the recovered triggers are different from the true triggers. Especially, the recovered trigger for GTSRB is quite different from the true one rather than the result on MNIST. It is considered that Neural Cleanse fails to recover triggers against TALPA.

Next, the input blurring is to blur any input image before giving it to a victim model. It potentially suppresses the effect of backdoors as outliers on neurons. We did not discuss the input

3.6 Discussion

blurring in this chapter because it intuitively deteriorates the accuracy. It is different from the improvement of the accuracy and the attack success rate, which is our goal. We plan to investigate the resistance of TALPA to the input blurring. We believe that TALPA resists the input blurring since the latent backdoor attack [2] can resist it.

3.6.6 Limitation

TALPA might be erased by retraining the victim model while the conventional backdoor attack [1] cannot be erased by the retraining. In particular, TALPA makes the latent representation of target inputs and poison inputs close to each other, so when the weights of the previous layer are updated as well, the latent representation of target inputs is corrected to the original distribution. It indicates that the latent representation of target inputs and poison inputs are separated, and hence the trigger cannot be recognized, and the attack success rate may decrease. It is an open problem whether backdoors can be embedded in the latent representation so that it remains even after retraining.

TALPA might be unable to embed backdoors into architectures with batch normalization [118]. Batch normalization normalizes the values of neurons by the mean and variance with respect to each batch. In the algorithm of TALPA, source inputs and target inputs are sampled and normalized, respectively, and then the mean and variance are significantly different from the clean data. Consequently, it might be impossible to embed backdoors into architectures with batch normalization. The application of TALPA to architectures with batch normalization will need to be undertaken.

3.6.7 Potential Countermeasures

A potential countermeasure against TALPA is to detect the backdoors themselves [115, 119–122]. The distribution of latent representations for source inputs of TALPA is identical to those for target inputs by the objective functions described in Section 3.4. It means that a victim model of TALPA may have unnatural latent representations compared with a clean model. Based on the above observation, TALPA may be detected by computing similarity between the classes, such as the median absolute deviation [123] utilized in GangSweep [124] and SCA [120] as detection tools. Further studies, which take the detection described above into account, will need to be undertaken.

3.7 Conclusion

In this chapter, we presented a novel backdoor attack, TALPA, which outperforms the latent backdoor attack in terms of the accuracy and the attack success rate without the loss of resistance to pruning and transfer learning. The key idea of TALPA is to directly override parameters in latent representations through the competitive learning between a generative model for triggers and a victim model. We conducted extensive experiments with TALPA for typical benchmark datasets and then demonstrated that TALPA provides higher accuracies and attack success rates than the latent backdoor attack [2]. We also confirmed that TALPA has resistance to pruning and transfer learning. In future work, we plan to discuss the detection of TALPA and its input blurring.

Chapter 4

Advanced Data Reconstruction Attack under a Malicious Model Trainer

In Chapter 2, we presented defense methods under the data reconstruction attacks setting in which the model users or data providers act as the adversaries. However, the threats in a more powerful scenario where the model trainer with full access to the training process acts as the adversary remain insufficiently understood. Similar to Chapter 3, Chapter 4 clarifies the threats of data reconstruction attacks that arise under such a setting.

4.1 Introduction

While Machine Learning as a Service (MLaaS) is becoming more widely used, confidential inference, in which the input data to machine learning models is kept secret due to privacy issues, is also attracting attention [125–128]. User data in MLaaS contains a variety of sensitive information such as patient records and the user’s faces for personal authentication. From the viewpoint of privacy protection [129], cryptography or hardware technology needs to be used to ensure that machine learning services do not reveal the user data.

However, a previous study [130] pointed out that the privacy of input data will be violated by using the outputs of machine learning models, even in situations where the input data is kept

4.1 Introduction

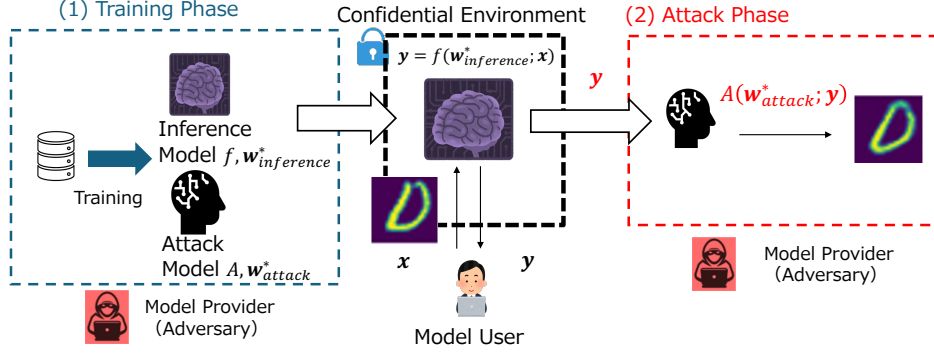


Figure 4.1: Overview of our attack.

secret using cryptography or hardware technology. That is, even if the input data is completely confidential, the model output has the potential to contain sensitive information about the input data. The previous study [130] demonstrated that an adversary is assumed to be a model provider, in which the roles of model trainer and model operator are held by the same individual (or entity), and can infer a sensitive attribute about the input data from the model output without any impact on the main task performances (e.g., face recognition and/or medical diagnosis). Specifically, the adversary trains the model so that the sensitive attribute information of the input data he/she wishes to retrieve is stealthily embedded in the model outputs. When the adversary obtains the model output, he/she can estimate the pre-specified sensitive attribute for the corresponding input data.

It is not trivial whether the input data can be fully reconstructed from the model output, although the previous study demonstrated the risk of extracting a specific sensitive attribute [130]. This is because the amount of the reconstruction information on the input data is more than that of the information on a particular attribute. For example, if the adversary wants to estimate the attribute of male or female from an output result, he/she only needs to embed one bit of information indicating 0 or 1 in the output result. Still, the reconstruction information of input data is expected to be richer because input data has many other sensitive attributes. Therefore it is unclear whether the reconstruction information can be coded into the output result.

In this chapter, we propose a simple yet effective attack method that reconstructs the input data even when it is kept secret. The overview of our attack is shown in Fig. 4.1. Specifically, our attack consists of training and attack phases. In the training phase, the adversary trains two models

simultaneously, called the inference and attack models. The inference model is trained to embed information about the input data in the model output while maintaining the high accuracy of the original task. The attack model is trained to reconstruct the input data from the inference model output. After that, the adversary deploys the inference model obtained through the training phase in the confidential environment, and a service user queries private data to the inference model. The adversary cannot access the queried data because of computing in a confidential environment. In the attack phase, the adversary receives only the output result of the inference model with regard to the queried data and can reconstruct it by using the attack model.

Experimental evaluations of our attack method using six image datasets of different complexity show that the simple dataset produced reconstructed images with a high degree of similarity to the input images. Furthermore, we show that the accuracy of the inference model is maintained at the same level as that of the normal training. We also propose a method to estimate the complexity of the dataset on the basis of LPIPS (Learned Perceptual Image Patch Similarity) [131], which is an image similarity measure and find a strong correlation between the complexity of the dataset and the similarity of the reconstructed images obtained by our attack method to the input images. In other words, by calculating the complexity of the dataset, it is possible to predict in advance how similar images reconstructed by our attack method will be to the input images, which may serve as a potential countermeasure.

4.2 Related Work

Machine learning models leak much information about their training dataset in both the white-box and black-box settings. Machine learning models learn as many features as they can, and also memorize sensitive information about some training data regardless of generalizing unknown data. The information obtained in the white box setting is the output results of the machine learning model and the intermediate states (i.e., the gradients or the values of intermediate layers). On the other hand, the only information available in the black box setting is the output results of the machine learning models. This is a more realistic setting because many machine learning services provide only the output results as an API. Various information obtained in each setting can provoke many

4.2 Related Work

attacks in machine learning models.

A model extraction attack [132, 133] extracts parameter information from a model’s output results and makes a surrogate model that performs similarly to the original model. In the model extraction attack, a model has the richness of the output results, and the richness can be utilized to extract parameter information. Still, we present how this richness can be used for encoding reconstruction information at inference time.

Privacy attacks such as a membership inference attack, an attribute inference attack, and a property inference attack infer partial information about the training data. Membership inference attacks [134, 135] try to determine whether a given individual’s data is included in the training dataset. Attribute inference attacks [136, 137] aim to infer unknown attributes of a partially known record used in the training dataset. Property inference attacks [138, 139] attempt to infer the property e.g., the ratio of males and females in the training dataset.

A data reconstruction attack [5–9, 29] reconstructs the features corresponding to particular target labels. This attack attempts to find data that achieve the maximum likelihood from the output results of the target model. In general, the adversary reconstructs training data using a generative model under the assumption that he/she has a public dataset that is similar to but does not overlap with the training data [5–7, 9]. In federated learning, where multiple parties submit their own gradients to train a global model, the adversary can reconstruct the target party’s local data by using the gradients computed by the target party [140–143]. The gradients can directly reflect the information about the training data used in the training step so that they can make data reconstruction easier. The data reconstruction attack is similar to our attack in that they reconstruct the data itself, although the adversary’s positions and objectives are different. Our attack cannot use rich information such as a public dataset or gradients, so reconstructing unknown input data from only limited dimensional output results is more complicated.

In the context of other privacy attacks, a malicious model provider infers sensitive information about the input data of the target model when the input data is concealed. The attack closest to our attack was proposed by Malekzadeh et al. [130]. They point out the risk of sensitive attribute of input data leaking out from model output. The adversary should directly attempt to reconstruct the data in order to violate more privacy of input data. However, the reconstruction of input data is not

considered in this attack, so the threat of privacy attacks that reconstruct input data is unknown. Song and Shamatikov [144] suggested an attack that reveals sensitive attributes about the input data from the model’s internal representation in overlearning. This attack has to access several internal representations and makes stronger attack assumptions than our assumptions.

4.3 Problem Setting

In this section, we first describe the problem setting of the confidential inference in MLaaS, and then describe the capacity and goal of the adversary who is assumed to be a model provider, in which the roles of model trainer and model operator are held by the same individual (or entity).

4.3.1 Confidential Inference in MLaaS

In MLaaS, the model trainer P trains an inference model $f : \mathbb{R}^{d_{\text{in}}} \rightarrow [0, 1]^{N_c}$ to be used as a service and deploys it in the execution environment (i.e., the cloud or the edge device). The output of the inference model is represented by the logit that is the raw output of the final layer without the softmax function and the confidence that is the normalized logit between 0 and 1 using the softmax function. The logit and confidence are often used as typical output types of a classification model.

We consider a confidential inference using MLaaS for protecting the privacy of the input data, e.g. cloud platforms such as confidential computing by Microsoft Azure ¹. Input data $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ is concealed by some cryptographic tools or hardware techniques and a user queries $\mathbf{x}$ to the inference model f and receives the output result without revealing it to the third parties (i.e., the cloud administrator and the model provider). That is, they cannot obtain the input data $\mathbf{x}$, the output result $\mathbf{y}$, and the internal state of the inference model f when computing $\mathbf{y} = f(\mathbf{w}_{\text{inference}}^*; \mathbf{x})$.

4.3.2 Attack Scenarios

We assume that a model provider is an adversary who violates the privacy of the user. Here, the model provider requires only an output result $\mathbf{y}$ to perform our attack. There are, for example, two situations in which the model provider can receive the model output.

¹<https://azure.microsoft.com/ja-jp/solutions/confidential-compute>

4.3 Problem Setting

The first is the case when the model provider receives the output to detect privacy attacks. The model provider cannot access input data, so its output result from the inference model is used to detect whether the input data is illegal or not. This is because the model provider should ideally check for privacy attacks [132, 134, 136, 138] on his/her model and prevent such attacks. For example, in a study [145], as a countermeasure against model extraction attack [132], the model provider detected abnormal input data used to attack by computing the distribution of their output results.

The second is the case when the model provider receives the output to improve the quality of the provided services [130]. In classification tasks, the final class for input data needs to be decided on the basis of the outputs of machine learning models. To do this, the model provider can provide a high-quality machine learning service by collecting and aggregating the outputs of multiple machine learning models.

4.3.3 Adversary's Capacity and Goal

Capacity. The adversary has two capabilities. First, the adversary has access to the training process. That is, the adversary can freely manipulate the training algorithm and the training data. This assumption is practical since the adversary acts as the model provider. Second, the adversary can only obtain an output result for input data, while keeping the input data secret, but not the input data or its computation process. This is due to the situation described in Section 4.3.1.

Goal. The adversary aims to reconstruct input data from the output result of the inference model. That is, the adversary aims to obtain an attack function $A : [0, 1]^{N_c} \rightarrow \mathbb{R}^{d_{in}}$ parameterized with $\mathbf{w}_{\text{attack}}$ such that $A(\mathbf{w}_{\text{attack}}; \mathbf{y}) = \hat{\mathbf{x}} \approx \mathbf{x}$ upon receiving the output result $\mathbf{y} = f(\mathbf{w}_{\text{inference}}; \mathbf{x})$. Furthermore, the adversary can manipulate $\mathbf{w}_{\text{inference}}$ of f and must provide $\mathbf{w}_{\text{attack}}^*$ with high accuracy, i.e., the same accuracy as the model trained normally. This is because if the test accuracy becomes low due to embedding the reconstruction information, no one can use the inference model.

In summary, the adversary's objective can be formulated as follows;

$$\begin{aligned} \mathbf{w}_{\text{inference}}^*, \mathbf{w}_{\text{attack}}^* = & \underset{\mathbf{w}_{\text{inference}}, \mathbf{w}_{\text{attack}}}{\operatorname{argmin}} \quad \mathbb{E}_{\mathbf{x}}[\text{distance}(A(\mathbf{w}_{\text{attack}}; f(\mathbf{w}_{\text{inference}}; \mathbf{x})), \mathbf{x})] \\ \text{s.t.} \quad & \text{Acc}_{\mathbf{w}_{\text{inference}}} \approx \text{Acc}_{\mathbf{w}_{\text{clean}}^*}, \end{aligned}$$

where distance function can be any distance between two data, w_{clean}^* is obtained by normal training, and Acc_w means the test accuracy of the inference model f with the parameter w .

4.4 Attack Method

Our attack consists of a training phase before deploying the inference model and an attack phase as a service after deployment.

4.4.1 Training Phase

Our training method is shown in Algorithm 3. As shown in line 6, the objective of the attack model A is to reduce the distance $\text{distance}(\hat{x}, x)$ between the data $\hat{x}$ output by A and the input data x of f . The distance function distance can be any distance, including Manhattan distance (L^1 norm) and Euclidean distance (L^2 norm).

On the other hand, as shown in line 7, the inference model f is trained to infer the correct label for the training data as in the usual training. CE in line 7 means the cross entropy function. Furthermore, we also aim to reduce the distance $\text{distance}(\hat{x}, x)$ between two data as the objective of the attack model A . In other words, by training the inference model to produce outputs that reduce the distance between the output data by A , in addition to the usual training, the output results can include reconstruction information of the input data.

Also, α, β is a hyperparameter that adjusts the balance between training normally and embedding the reconstruction information to the output result.

4.4.2 Attack Phase

After completing the training phase, the adversary deploys the updated F to the execution environment for the service. Users for the service can query their private data to F without revealing them to a third party. After that, the adversary receives the output results in some way. The adversary can reconstruct the input data by using the attack model A updated by Algorithm 3.

Algorithm 3 Training Algorithm of Our Attack

Input: $D^{(\text{train})}$: the training dataset, f : the inference model, A : the attack model, α, β : the hyper-parameters for trade-off, BS: batch size, E : the number of epochs.

Output: $w_{\text{attack}}^*, w_{\text{inference}}^*$.

```

1: for range(0,  $E$ ) do
2:   for range(0,  $\lceil \frac{|D^{(\text{train})}|}{\text{BS}} \rceil$ ) do
3:      $(x, y) \leftarrow D^{(\text{train})}$ ; choose BS examples from  $D^{(\text{train})}$ 
4:      $\hat{y} = f(w_{\text{inference}}; x)$ 
5:      $\hat{x} = A(w_{\text{attack}}; \hat{y})$ 
6:      $L_A = \text{distance}(\hat{x}, x)$ 
7:      $L_f = \alpha \text{CE}(y, \text{Softmax}(\hat{y})) + \beta \text{distance}(\hat{x}, x)$ 
8:     Update  $w_{\text{attack}}, w_{\text{inference}}$  from  $L_A$  and  $L_f$ .
9:   end for
10: end for

```

4.5 Evaluation

In this section, we present an experimental evaluation for our attack method. We conduct some experiments to answer the following research questions (RQs);

- *RQ1*: How similar to the input data can be reconstructed by Algorithm 3? This is the main objective of our attack.
- *RQ2*: How much does the accuracy of the inference model trained with Algorithm 3 degrade compared with that of the inference model trained normally? The output results are embedded with the reconstruction information on the input data, so this information may negatively affect the main task performance of the inference model. Therefore, the inference accuracy will be degraded compared with the normal inference model.
- *RQ3*: How does the inference accuracy and the similarity with the input data change depending on the type of output results obtained by the adversary? The reason is that the amount of reconstruction information that an adversary can embed changes depending on the type of output results. In this experiment, we evaluate the logit before applying the softmax function and the confidence after applying the softmax function.

Table 4.1: The information on the datasets and the architectures in our setting.

Dataset	Num of classes C	Input size m	Num of training data	Num of test data	Type of f
MNIST	10	$784(28 \times 28 \times 1)$	60000	10000	LeNet
FashionMNIST	10	$784(28 \times 28 \times 1)$	60000	10000	LeNet
SVHN	10	$3072(32 \times 32 \times 3)$	73257	26032	ResNet18
GTSRB	43	$3072(32 \times 32 \times 3)$	39209	12630	ResNet18
CIFAR10	10	$3072(32 \times 32 \times 3)$	50000	10000	ResNet18
CIFAR100	100	$3072(32 \times 32 \times 3)$	50000	10000	ResNet18

4.5.1 Experimental Setup

In this experiment, we used six datasets for image classification (i.e., MNIST, FashionMNIST, SVHN, GTSRB, CIFAR10, and CIFAR100). Each dataset has a different task complexity (i.e., data variation). The details of each dataset and the types of the inference model architecture are summarized in Table 4.1. The architecture of the attack model A is the same for all datasets and consists of three layers of repeated up-sampling and convolution. In Algorithm 3, the parameters are given by $E = 200$, $B = 32$, $\alpha = 1.0$, $\beta = 1.0$. The α and β were set so that the balance between the inference accuracy and the reconstruction similarity is just right. The distance function d is the Euclidean distance (L^2 norm) in our experimental setting.

4.5.2 Results

Reconstruction of the input data.

Image samples reconstructed by our attack method are shown in Fig. 4.2. As can be seen from the samples, MNIST, FashionMNIST, and GTSRB seem to succeed in reconstructing images that capture the features of the original images. For instance, the slope of numbers in MNIST and the brightness of road signs in GTSRB can be reproduced in the reconstructed images. On the other hand, CIFAR10 and CIFAR100 are generally blurred and do not seem to be able to reconstruct images that are similar to the original images. The degree to which these reconstructed images are similar to the original images is discussed in detail in Section 4.6.

4.5 Evaluation



Figure 4.2: Reconstructed image samples. In each dataset, the left side is the original image and the right side is the image reconstructed by our attack.

The convergence of the attack model A with respect to the test data is shown in Fig. 4.3. The results of running this attack algorithm are shown in the graphs of `attack_logits` and `attack_confidence`. Although some instability in training is observed in Figs. 4.3c and 4.3d, the loss decreases for all datasets as the epoch increases. On the other hand, not all datasets show the same value of convergence, and the values are higher for CIFAR10 and CIFAR100 than for the other datasets.

Inference accuracy.

The change in the inference accuracy of f as the epoch E increases is shown in Fig. 4.4. Although some instability in training is observed when logit is used in the GTSRB dataset in Fig. 4.4d, the accuracy was found to be eventually maintained at the same level as that of normal learning at 200 epochs. In addition, the accuracy is kept as good as that of the normal training on all datasets regardless of whether logit or confidence is used as outputs of the inference model f .

The type of the output results.

Fig. 4.5 shows the reconstructed image samples in accordance with the type of output results. Visually, it appears that the same level of reconstruction is achieved in both cases of the logit and the confidence. However, we can see from Fig. 4.3 that the logit is moving toward convergence with a value smaller than the confidence. The confidence is once normalized from 0 to 1, with the restriction that the sum of the confidence of each class is equal to 1. Logit, on the other hand, is not normalized and the values of the final output layer are used as they are, so there are no restrictions on the values. Therefore, the loss of A is considered to be lower than that of the logit because there is less reconstruction information in the confidence than in the logit.

4.6 Discussion

In this section, we first describe the effectiveness of Algorithm 3. Next, we evaluate it using image-specific similarity measures and discuss how similar the reconstructed images are from a different perspective. We also discuss the difference in the convergence of the loss of the attack model A among datasets, i.e., the difficulty of data reconstruction, as shown in Fig. 4.3.

4.6.1 Effectiveness of Our Attack

We discuss how much of the input data reconstruction information can be embedded in the output results by Algorithm 3. First, we conduct experiments about how much information on the input data can be extracted from the output results of the normal inference model f . That is, we set $\beta = 0$ in Algorithm 3. f does not include the recovered information of the input data in the output results, and A tries to see how similar the output images are in the normal training process. The results are shown in the label of normal_logits and normal_confidence of Fig. 4.3.

These results show that the loss values of normal_logits and normal_confidence are higher than those of attack_logits and attack_confidence for all datasets. That is, the adversary cannot reconstruct the input data from a normally trained inference model, regardless of the availability of logit

4.6 Discussion

or confidence information. Besides, these results indicate that the difference between these loss values denotes the amount of reconstruction information on the input data embedded by the adversary.

4.6.2 Similarity with Reconstructed Images

In Section 4.5.2, we evaluate the distance between the reconstructed images and the input images by using the L^2 norm that can measure the distance among any type of data. In this section, we use SSIM (Structural Similarity) [146] and LPIPS (Learned Perceptual Image Patch Similarity) [131] as image-specific similarity measures.

SSIM is a metric to evaluate the similarity between images by combining the comparisons of brightness, contrast, and structural patterns. For example, when comparing two RGB images, if one image has stronger luminance than the other, the difference in each pixel is calculated and added together. The difference may be larger, but the structure and contrast may be identical. In such a case, the similarity can be evaluated as an indicator of high similarity. The purpose of using SSIM is to evaluate the similarity between the structural states of the recovered image and the original image.

LPIPS is a metric that focuses on the distance between feature representations in neural networks. According to the literature [131], the latent representation of VGG models trained with ImageNet is close to human perceptual recognition. This means that if the distance between the feature representations of trained VGG models in two images is close, they appear to be the same image from the human point of view. The previous works [147, 148] in the context of image reconstruction used LPIPS to evaluate the similarity between the reconstructed images and the original images, so we use LPIPS as in these works.

The results evaluated by SSIM and LPIPS are shown in Figs. 4.6 and 4.7, respectively. Here, SSIM means that the closer to 1, the higher the similarity, and LPIPS means that the closer to 0, the higher the similarity. For clarity in comparing the two indices, the value of $1 - (\text{SSIM})$ is shown in Fig. 4.6.

As shown in Figs. 4.6 and 4.7, the values of SSIM and LPIPS indicate similar trends. In other

words, the reconstructed images show similar trends between structural patterns and human perception. However, the variance in SSIM focusing on structural patterns tends to be larger than that in LPIPS. This indicates that although there is some blurring when focusing on the structural patterns between images, the images are close to each other without much effect on human perception.

4.6.3 Difficulty of data reconstruction

We discuss the difficulty of the input data reconstruction in our attack. One reason for the difficulty is naively considered to be the number of dimensions of the output results. If the number of dimensions of the output results is large, the input data is expected to become easier to reconstruct since more reconstruction information on the input data can be embedded in the output results. However, Figs. 4.6 and 4.7 show the averages of SSIM and LPIPS for CIFAR100 are almost the same as those of CIFAR10, even though the number of output dimensions of CIFAR100 is larger than that of CIFAR10. This fact indicates that larger dimensionality of the output results does not necessarily mean easier reconstruction.

Therefore we consider the complexity of the dataset itself, independent of the number of dimension of the output results, as it relates to the difficulty of reconstructing the data. In fact, as shown in Figs. 4.6 and 4.7, the averages of SSIM and LPIPS are smaller for SVHN than for CIFAR10 for the same number of classes and dimensions, respectively. This can be attributed to the complexity of the dataset. We define the complexity of the dataset as the degree of scatter of image data in each class, i.e., the similarity of data in each class. The degree of scatter of data in class j is expressed by the following equation;

$$\text{Class Complexity} = \frac{1}{q} \sum_{i=1}^q \text{LPIPS}(\mathbf{x}_{i,j}, \mathbf{x}_{i,j}^*). \quad (4.1)$$

Equation 4.1 means LPIPS is computed for two images $\mathbf{x}_{i,j}, \mathbf{x}_{i,j}^*$ sampled randomly from class j and the average of q repetitions is the complexity of class j . That is, if the images in a class are similar overall, the value of LPIPS is small regardless of which two images are sampled, and the value of Class Complexity is also small. On the other hand, if the images in the class are diverse, the value of LPIPS is larger so the value of Class Complexity is also larger.

4.6 Discussion

Also, the following equation shows how accurately the image of class j can be reconstructed;

$$\text{Reconstruction Similarity} = \frac{1}{q} \sum_{i=1}^q \text{LPIPS}(A(\mathbf{w}_{\text{attack}}; f(\mathbf{w}_{\text{inference}}; \mathbf{x}_{i,j})), \mathbf{x}_{i,j}), \quad (4.2)$$

where A is an algorithm such that any image is output from the output results of the inference model f .

We show the relationship between Class Complexity and Reconstruction Similarity for each class in each dataset by using Equations 4.1 and 4.2 in Fig. 4.8. From the results in Fig. 4.8, there is a strong correlation between Class Complexity and Reconstruction Similarity. This indicates that when the value of Class Complexity is small (i.e., when there are many similar data in a class), images highly similar to the input images are reconstructed. As a result, by calculating Class Complexity of the dataset, it is possible to approximately understand how closely our attack will reconstruct the input images. Another aspect implies that reconstruction by our attack method is easy for datasets where each instance is of a similar type for one class. On the other hand, it indicates that reconstruction by our attack is difficult for datasets where each instance is very different for one class.

There are various possible use cases for Class Complexity. For example, when a company creates a dataset and sells it to another company that wants to use the dataset for training MLaaS, Class Complexity of the dataset can be calculated to know the risk of data reconstruction in advance. In addition, if some attack methods are discovered in subsequent research, we can calculate how good the method is by examining the relationship between Class Complexity and Reconstruction Similarity for a dataset.

4.6.4 Relation to Autoencoders

Autoencoders [149–151] are primarily designed to encode the input data into a compressed representation and decode it such that the reconstructed data is as similar to the original data as possible. Autoencoders are close to our attack because our goal is also to reconstruct the input data from the output result of the inference model. However, our attack method is different from autoencoders because the encoder also plays a role as the classification model in our problem setting. That is,

the encoder has to not only compresses input data into a meaningful representation but also classify the input data into a correct class. We experimentally found that the encoder can play a role as the inference model with the same accuracy as the normal inference model as described in Section 4.5.

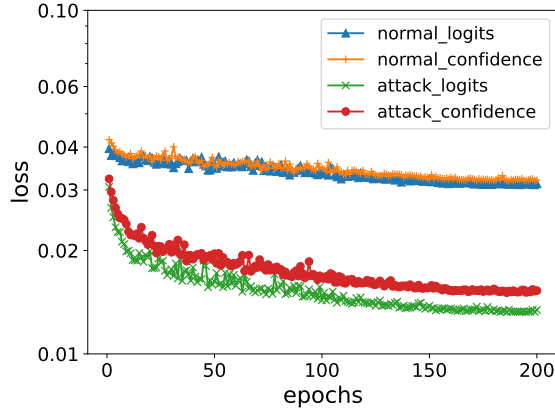
The quality of the data reconstructed by autoencoders depends on the number of dimensions in the compressed representation. To improve the quality of the reconstructed data, the number of dimensions should increase. However, in our problem setting, the number of dimensions in the compressed representation must be equal to the number of output dimensions of the inference model, i.e., the number of classes, so it is difficult to change the number of dimensions like autoencoders. Improving the quality of reconstructed data while keeping the number of dimensions constant remains future work.

4.6.5 Potential Countermeasures

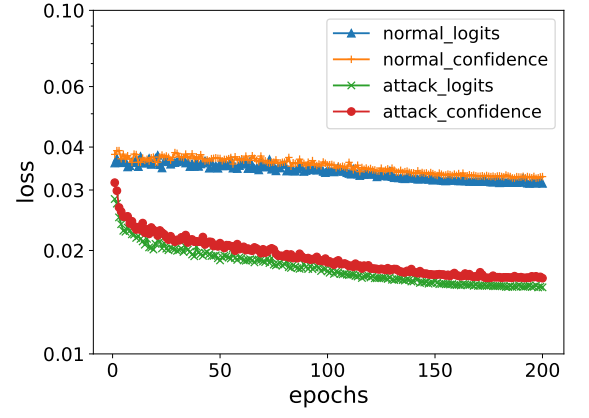
The difference between the distributions of output results of the normal inference model and the attacked inference model can be used as a countermeasure. Our attack is realized by embedding the reconstruction information about the input data in the output results. In other words, the distribution of the output results is expected to be significantly different from the normal distribution of the output results since some information is included in addition to the output results obtained by the usual training process. For example, since the entropy of the output results may be larger than that of the normal output results in the case of the attack, it is highly likely that the inference model has been attacked if the entropy of the output results exceeds a certain value. On the other hand, how to set an appropriate threshold for detecting our attack is still an issue. Another countermeasure is to keep the high complexity of each class in a dataset used for training. As discussed in Section 4.6.2, it is difficult to reconstruct images with high similarity to input images by our attack against a dataset in which each class has high complexity. However, if we make a dataset with the high complexity of each class, the classification for a dataset may become difficult and the accuracy may decrease, so the trade-off is an issue to be addressed.

4.7 Conclusion

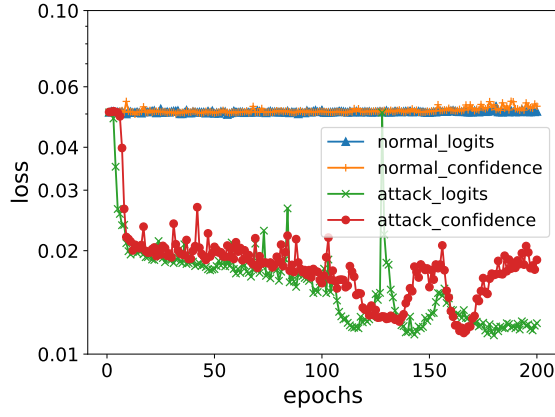
In this chapter, we presented an attack that can obtain images with high similarity to the input data by embedding the reconstruction information of the input data in the output result of machine learning models. We also found that the inference model is as accurate as the model trained normally, even when the input data reconstruction information is embedded in the output result. In fact, our attack is a threat to privacy violation in an environment where the model is used while keeping the input data secret because the adversary can reconstruct the input data if only the output result is obtained. We also introduced a measure of the complexity of the dataset and found a strong correlation between the complexity and the similarity. Future work includes improving our attack method so that it can achieve high similarity even with high-complexity datasets and extending our attack method so that it can be adapted to domains other than image classification.



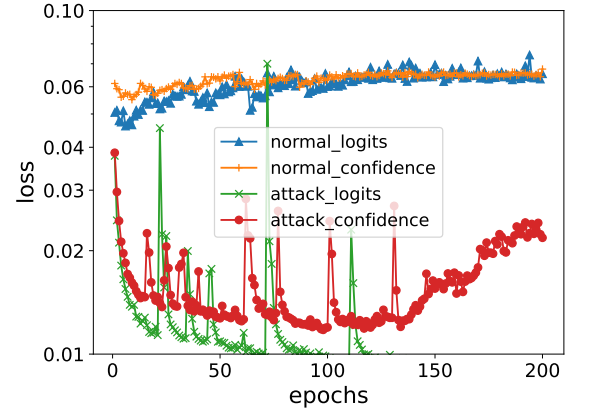
(a) MNIST



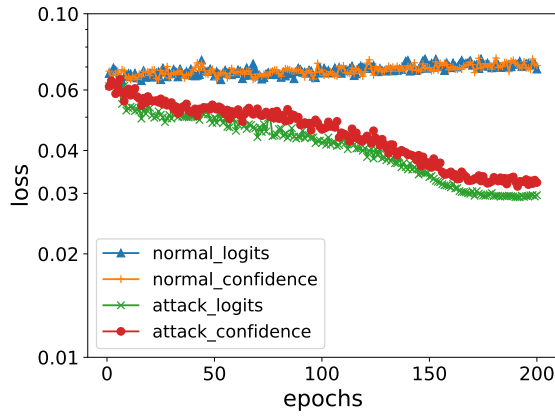
(b) FashionMNIST



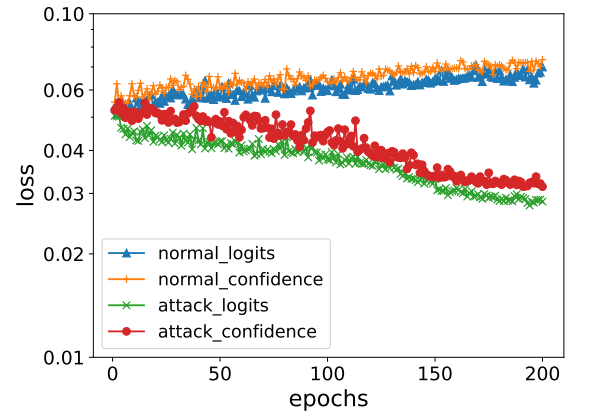
(c) SVHN



(d) GTSRB



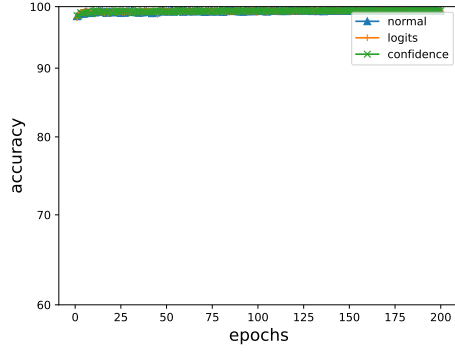
(e) CIFAR10



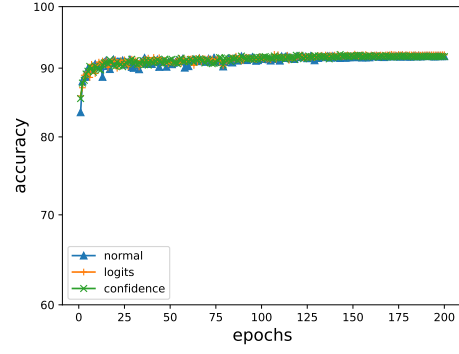
(f) CIFAR100

Figure 4.3: Loss of the attack model A .

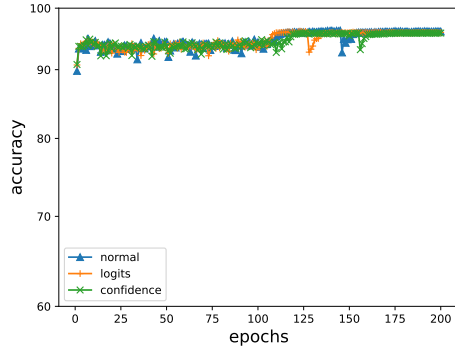
4.7 Conclusion



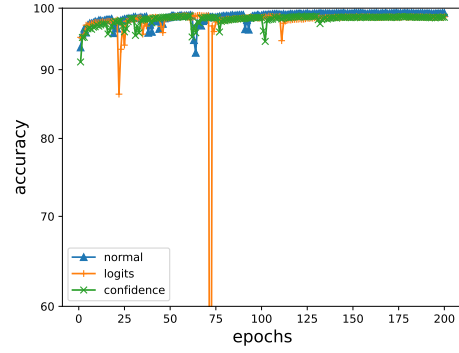
(a) MNIST



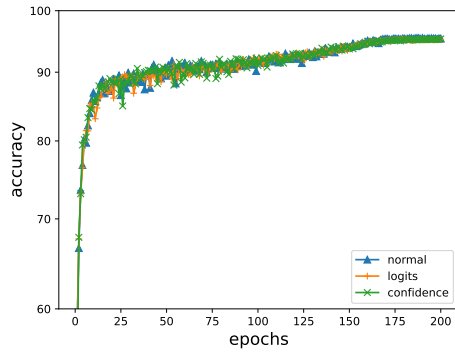
(b) FashionMNIST



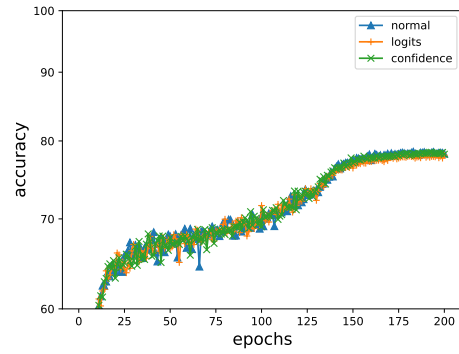
(c) SVHN



(d) GTSRB



(e) CIFAR10



(f) CIFAR100

Figure 4.4: Test accuracy of the inference model f .



Figure 4.5: Reconstructed image samples in accordance with the type of the output results. “Original” row shows the original images, “Logit” row shows the images reconstructed by using the logit as the output, and “Conf” row shows the images reconstructed by using the confidence as the output.

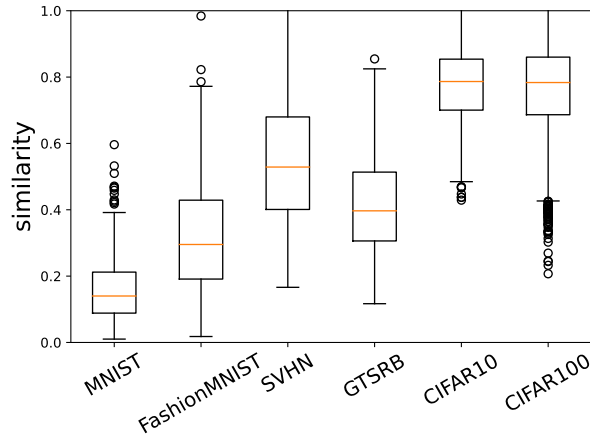


Figure 4.6: The values of $1 - (\text{SSIM})$ for each dataset.

4.7 Conclusion

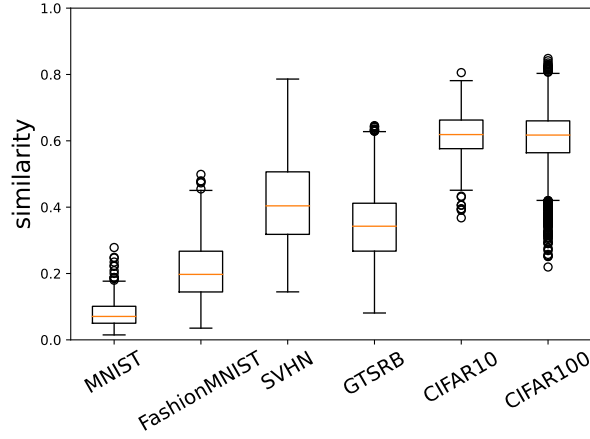


Figure 4.7: The values of LPIPS for each dataset.

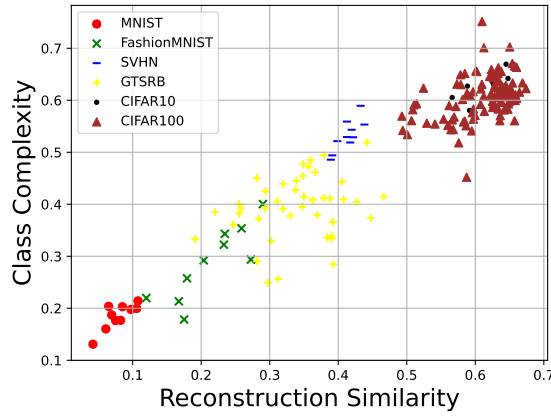


Figure 4.8: Reconstruction Similarity and Class Complexity in each dataset. The horizontal axis indicates the value of Reconstruction Similarity and the vertical axis indicates the value of Class Complexity calculated by Equations 4.1 and 4.2.

Chapter 5

Conclusion

The AI supply chain in this thesis is an end-to-end workflow that represents the full lifecycle of an AI system, covering data collection, model training, deployment, and operation, and involving multiple entities such as data providers, model trainers, model providers, and model users. This thesis has examined fundamental security and privacy challenges that arise from malicious model trainers in the AI supply chain systems and has presented a unified attack and defense perspective when the model trainers act maliciously in distributed and centralized learning settings.

In the first part (Chapter 2) of this thesis, we focused on security and privacy risks when local model trainers act maliciously in a distributed learning setting. In this scenario, we proposed the Secure and Private Gradient Computation (SPGC) framework, which integrates Secure Multi-party Computation (MPC) and Differential Privacy (DP) to simultaneously achieve security against backdoor attacks and privacy guarantees against data reconstruction attacks under malicious local model trainers. Our experimental results across multiple datasets showed that SPGC significantly reduces vulnerability to these attacks while maintaining higher model accuracy compared to existing approaches, demonstrating that strong security and privacy guarantees can coexist with practical model utility in collaborative learning environments.

The second part (Chapter 3) of this thesis explored a fundamentally different and substantially more powerful adversarial setting: the case in which the model trainer in a centralized learning setting behaves maliciously to conduct backdoor attacks. Under this stronger adversary setting,

revealing potential threats of backdoor attacks enables us to proactively develop corresponding defense mechanisms. We introduced TALPA, a backdoor attack that directly manipulates latent representations to embed highly persistent and stealthy malicious behaviors, and discussed the potential defense methods against TALPA. This attack demonstrated strong backdoor removal resilience to common defenses such as pruning and transfer learning, illustrating that a malicious model trainer can realize backdoor behaviors far beyond the capabilities assumed in traditional research.

The third part (Chapter 4) of this thesis further investigated the threat of data reconstruction under a malicious model trainer and proposed a novel attack strategy in which input-dependent information is intentionally encoded into model outputs during training. Like Chapter 3, under this stronger adversary setting, identifying the potential threats of data reconstruction attacks allows us to proactively develop appropriate defense mechanisms. Our attack enables the reconstruction of user inputs even in confidential inference settings where input data remain hidden from the model trainer, revealing a fundamental privacy vulnerability that cannot be mitigated by confidential inference settings alone. Further analysis showed that the class diversity of the training dataset significantly influences reconstruction difficulty, providing insight into how the training dataset structure contributes to privacy risk in our attack.

Taken together, the contributions of this thesis provide a comprehensive characterization of security and privacy across the AI supply chain systems. Under malicious model trainers, we have established effective defenses in a distributed learning setting and revealed new attack threats in a centralized learning setting. These findings underscore the necessity of expanding threat scenarios of security and privacy to include the model trainer as a potentially adversarial entity, particularly as AI models are increasingly shared, reused, and integrated across complex supply chains.

A further research direction to realize secure and privacy-preserving AI supply chain systems is to develop theoretically grounded defenses that remain robust even against adversaries with full control over the training pipeline. We presented one example of a threat that can arise under such an adversary model in this thesis, but many other potential threats may still exist. Therefore, it is important to theoretically characterize the range of attacks that defense mechanisms can protect against. Another research direction is to enable each entity to verify the security and privacy of the AI supply chain systems through the introduction of auditability and certification mechanisms for

data and models. This verifiable trust is crucial for building trustworthy AI supply chain systems that can safely support future societal and industrial needs.

Through the investigation of both attack and defense perspectives under malicious model trainers, this thesis contributes to the foundation for constructing secure and privacy-preserving AI supply chain systems. We hope that the insights gained here will advance the development of trustworthy AI supply chain systems and foster a deeper understanding of the intricate interplay between security, privacy, and the structure of the AI supply chain systems.

Bibliography

- [1] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, “Badnets: Evaluating backdooring attacks on deep neural networks,” *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.
- [2] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, “Latent backdoor attacks on deep neural networks.” Association for Computing Machinery, 2019, pp. 2041–2055.
- [3] K. Doan, Y. Lao, W. Zhao, and P. Li, “Lira: Learnable, imperceptible and robust backdoor attacks,” in *Proc. of ICCV 2021*, 2021, pp. 11 966–11 976.
- [4] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *Proceedings of the 2015 ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.
- [5] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, “Neural network inversion in adversarial setting via background knowledge alignment,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 225–240.
- [6] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz, “Dreaming to distill: Data-free knowledge transfer via deepinversion,” in *The IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [7] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, “The secret revealer: Generative model-inversion attacks against deep neural networks,” in *Proc. of CVPR*. IEEE Computer Society, 2020, pp. 250–258.

BIBLIOGRAPHY

- [8] N. Haim, G. Vardi, G. Yehudai, O. Shamir, and M. Irani, “Reconstructing training data from trained neural networks,” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 22 911–22 924.
- [9] S. An, G. Tao, Q. Xu, Y. Liu, G. Shen, Y. Yao, J. Xu, and X. Zhang, “Mirror: Model inversion for deep learning network with high fidelity,” in *Proceedings of the 29th Network and Distributed System Security Symposium*, 2022.
- [10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.
- [11] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, “A survey on federated learning,” *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.
- [12] C. Xie, K. Huang, P.-Y. Chen, and B. Li, “Dba: Distributed backdoor attacks against federated learning,” in *International conference on learning representations*, 2019.
- [13] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, “How to backdoor federated learning,” in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 2938–2948.
- [14] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, “Attack of the tails: Yes, you really can backdoor federated learning,” *Advances in neural information processing systems*, vol. 33, pp. 16 070–16 084, 2020.
- [15] H. Yang, M. Ge, K. Xiang, and J. Li, “Using highly compressed gradients in federated learning for data reconstruction attacks,” *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 818–830, 2022.
- [16] H. Liu, B. Li, C. Gao, P. Xie, and C. Zhao, “Privacy-encoded federated learning against gradient-based data reconstruction attacks,” *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 5860–5875, 2023.

- [17] J. C. Zhao, A. Sharma, A. R. Elkordy, Y. H. Ezzeldin, S. Avestimehr, and S. Bagchi, “Loki: Large-scale data reconstruction attack against federated learning through model manipulation,” in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 1287–1305.
- [18] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *Proc. of CCS*. ACM, 2016, pp. 308–318.
- [19] C. Dwork, “Differential privacy,” in *Proc. of ICALP*, ser. Lecture Notes in Computer Science, vol. 4052. Springer, 2006, pp. 1–12.
- [20] L. Miao, W. Yang, R. Hu, L. Li, and L. Huang, “Against backdoor attacks in federated learning with differential privacy,” in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 2999–3003.
- [21] M. Naseri, J. Hayes, and E. D. Cristofaro, “Local and central differential privacy for robustness and privacy in federated learning,” *Proceedings 2022 Network and Distributed System Security Symposium*, 2020.
- [22] D. Xiao, J. Li, and M. Li, “Privacy-preserving federated compressed learning against data reconstruction attacks based on secure data,” in *International Conference on Neural Information Processing*. Springer, 2023, pp. 325–339.
- [23] S. Liu, Z. Wang, Y. Chen, and Q. Lei, “Data reconstruction attacks and defenses: A systematic evaluation,” *arXiv preprint arXiv:2402.09478*, 2024.
- [24] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, “Local privacy and statistical minimax rates,” in *Proc. of FOCS*. IEEE, 2013, pp. 429–438.
- [25] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, “Targeted backdoor attacks on deep learning systems using data poisoning,” *arXiv preprint arXiv:1712.05526*, 2017.
- [26] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, “Trojaning attack on neural networks,” in *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018.

BIBLIOGRAPHY

- [27] K. Liu, B. Dolan-Gavitt, and S. Garg, “Fine-pruning: Defending against backdooring attacks on deep neural networks,” in *Proc. of RAID 2018*, ser. LNCS, vol. 11050. Springer, 2018, pp. 273–294.
- [28] S. Wang, S. Nepal, C. Rudolph, M. Grobler, S. Chen, and T. Chen, “Backdoor attacks against transfer learning with pre-trained deep learning models,” *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1526–1539, 2022.
- [29] M. Fredrikson, E. Lantz, S. Jha, S. M. Lin, D. Page, and T. Ristenpart, “Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing,” in *USENIX Security Symposium*. USENIX Association, 2014, pp. 17–32.
- [30] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, “Exploiting unintended feature leakage in collaborative learning,” in *Proc. of IEEE S&P*. IEEE, 2019, pp. 691–706.
- [31] M. Nasr, R. Shokri, and A. Houmansadr, “Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning,” in *Proc. of IEEE S&P*. IEEE, 2019, pp. 739–753.
- [32] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Proc. of TCC*, ser. LNCS, vol. 3876. Springer, 2006, pp. 265–284.
- [33] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *Proc. of CCS*. ACM, 2015, pp. 1310–1321.
- [34] M. Pettai and P. Laud, “Combining differential privacy and secure multiparty computation,” in *Proceedings of the 31st annual computer security applications conference*, 2015, pp. 421–430.
- [35] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, “A hybrid approach to privacy-preserving federated learning,” in *Proc. of AISec*. ACM, 2019, pp. 1–11.

- [36] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, “Hybridalpha: An efficient approach for privacy-preserving federated learning,” in *Proc. of AISec.* ACM, 2019, pp. 13–23.
- [37] M. Chase, R. Gilad-Bachrach, K. Laine, K. Lauter, and P. Rindal, “Private collaborative neural network learning,” *Cryptology ePrint Archive*, 2017.
- [38] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.
- [39] A. C. Yao, “Protocols for secure computations,” in *Proc. of FOCS.* IEEE, 1982, pp. 160–164.
- [40] A. Shamir, “How to share a secret,” *Communication of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [41] E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, T. Schneider, and F. Koushanfar, “Tinygarble: Highly compressed and scalable sequential garbled circuits,” in *Proc. of IEEE S&P.* IEEE, 2015, pp. 411–428.
- [42] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *Proc. of IEEE S&P.* IEEE, 2017, pp. 3–18.
- [43] E. Kushilevitz, Y. Lindell, and T. Rabin, “Information-theoretically secure protocols and security under composition,” *SIAM Journal on Computing*, vol. 39, pp. 2090–2112, 2010.
- [44] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang, “Membership inference attack against differentially private deep learning model,” *Transactions on Data Privacy*, vol. 11, no. 1, pp. 61–79, 2018.
- [45] C. Park, D. Hong, and C. Seo, “An attack-based evaluation method for differentially private learning against model inversion attack,” *IEEE Access*, vol. 7, pp. 124 988–124 999, 2019.
- [46] F. D. McSherry, “Privacy integrated queries: An extensible platform for privacy-preserving data analysis,” in *Proc. of SIGMOD.* ACM, 2009, pp. 19–30.

BIBLIOGRAPHY

- [47] P. Kairouz, S. Oh, and P. Viswanath, “The composition theorem for differential privacy,” *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 4037–4049, 2017.
- [48] I. Goodfellow, “Efficient per-example gradient computations,” 2015.
- [49] P. Mohassel and Y. Zhang, “Secureml: A system for scalable privacy-preserving machine learning,” in *Proc. of IEEE S&P 2017*. IEEE, 2017, pp. 19–38.
- [50] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, “Differentially private model publishing for deep learning,” in *IEEE S&P 2019*. IEEE, 2019, pp. 332–349.
- [51] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our data, ourselves: Privacy via distributed noise generation,” in *Proc. of EUROCRYPT*, ser. LNCS, vol. 4004. Springer, 2006, pp. 486–503.
- [52] A. Smith, A. Thakurta, and J. Upadhyay, “Is interaction necessary for distributed private learning?” in *Proc. of IEEE S&P*. IEEE, 2017, pp. 58–77.
- [53] J. Bohler and F. Kerschbaum, “Secure sublinear time differentially private median computation,” in *Proc. of NDSS*. Internet Society, 2020.
- [54] V. Rastogi and S. Nath, “Differentially private aggregation of distributed time-series with transformation and encryption,” in *Proc. of SIGMOD*. ACM, 2010, pp. 735–746.
- [55] E. Shi, T. H. Chan, E. Rieffel, R. Chow, and D. Song, “Privacy-preserving aggregation of time-series data,” in *Proc. of NDSS*. Citeseer, 2011, pp. 1–17.
- [56] S. Goryczka and L. Xiong, “A comprehensive comparison of multiparty secure additions with differential privacy,” *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 5, pp. 463–477, 2017.
- [57] F. Eigner, A. Kate, M. Maffei, F. Pampaloni, and I. Pryvalov, “Differentially private data aggregation with optimal utility,” in *Proc. of ACSAC*. ACM, 2014, pp. 316–325.

- [58] M. Naor, B. Pinkas, and E. Ronen, “How to (not) share a password: Privacy preserving protocols for finding heavy hitters with adversarial behavior,” in *Proc. of CCS*. ACM, 2019, pp. 1369–1386.
- [59] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, “A generic framework for privacy preserving deep learning,” in *Proc. of PPML with NeurIPS*, 2018.
- [60] D. Boneh, A. Sahai, and B. Waters, “Functional encryption: Definitions and challenges,” in *Proc. of TCC*, ser. LNCS, vol. 6597. Springer, 2011, pp. 253–273.
- [61] I. Damgård and M. Jurik, “A generalisation, a simplification and some applications of Paillier’s probabilistic public-key system,” in *Proc. of PKC*, ser. LNCS, vol. 1992. Springer, 2001, pp. 119–136.
- [62] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, “Differentially private model publishing for deep learning,” in *Proc. of IEEE S&P*. IEEE, 2019, pp. 332–349.
- [63] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, “Scalable private learning with PATE,” in *Proc. of ICLR*, 2018.
- [64] P. Mohassel and P. Rindal, “Aby3: A mixed protocol framework for machine learning,” in *Proc. of CCS 2018*. ACM, 2018, pp. 35–52.
- [65] S. Wagh, D. Gupta, and N. Chandran, “Securenn: 3-party secure computation for neural network training,” *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 26 – 49, 2019.
- [66] N. Agrawal, A. Shahin Shamsabadi, M. J. Kusner, and A. Gascón, “Quotient: Two-party secure neural network training and prediction,” in *Proc. of CCS 2019*. ACM, 2019, pp. 1231–1247.
- [67] A. Dalskov, D. Escudero, and M. Keller, “Secure evaluation of quantized neural networks,” *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 4, pp. 355–375, 2020.

BIBLIOGRAPHY

- [68] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J. Bossuat, J. S. Sousa, and J. Hubaux, “POSEIDON: privacy-preserving federated neural network learning,” in *Proc. of NDSS*. Internet Society, 2021.
- [69] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, “Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing,” in *Proc. of USENIX Security*. USENIX Association, 2014, pp. 17–32.
- [70] A. Sanyal, M. Kusner, A. Gascon, and V. Kanade, “Tapas: Tricks to accelerate (encrypted) prediction as a service,” in *Proc. of ICML 2018*, ser. PMLR, vol. 80, 2018, pp. 4497–4506.
- [71] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. E. Lauter, and F. Koushanfar, “XONN: xnor-based oblivious deep neural network inference,” in *Proc. of USENIX Security 2019*. USENIX Association, 2019, pp. 1501–1518.
- [72] N. Dowlin, R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” in *Proc. of ICML 2016*, 2016, pp. 201–210.
- [73] J. Liu, M. Juuti, Y. Lu, and N. Asokan, “Oblivious neural network predictions via minionn transformations,” in *Proc. of CCS 2017*. ACM, 2017, pp. 619–631.
- [74] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, “Gazelle: A low latency framework for secure neural network inference,” in *Proc. of USENIX Security 2018*. USENIX Association, 2018, pp. 1651–1668.
- [75] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, “Fast homomorphic evaluation of deep discretized neural networks,” in *Proc. of CRYPTO 2018*, ser. LNCS, vol. 10993. Springer, 2018, pp. 483–512.
- [76] H. Chaudhari, R. Rachuri, and A. Suresh, “Trident: Efficient 4pc framework for privacy preserving machine learning,” in *Proc. of NDSS 2020*. The Internet Society, 2020, pp. 1–18.

- [77] A. Patra and A. Suresh, “Blaze: Blazing fast privacy-preserving machine learning,” in *Proc. of NDSS 2020*. The Internet Society, 2020, pp. 1–18.
- [78] M. Byali, H. Chaudhari, A. Patra, and A. Suresh, “Flash: Fast and robust framework for privacy-preserving machine learning,” *Proceedings on Privacy Enhancing Technologies*, vol. 2, pp. 459–480, 2020.
- [79] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, “Delphi: A cryptographic inference service for neural networks,” in *Proc. of USENIX Security*. USENIX Association, 2020, pp. 2505–2522.
- [80] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, “Cryptflow : Secure tensorflow inference,” in *Proc. of IEEE S&P 2020*. IEEE, 2020, pp. 1646–1663.
- [81] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, “Cryptflow2: Practical 2-party secure inference,” in *Proc. of CCS*. ACM, 2020, pp. 325–342.
- [82] B. Yang, I. Sato, and H. Nakagawa, “Bayesian differential privacy on correlated data,” in *Proc. of SIGMOD*. ACM, 2015, pp. 747–762.
- [83] C. Liu, S. Chakraborty, and P. Mittal, “Dependence makes you vulnerable: Differential privacy under dependent tuples,” in *Proc. of NDSS*. Internet Society, 2016, pp. 21–24.
- [84] I. Mironov, “Rényi differential privacy,” in *Proc. of CSF*. IEEE, 2017, pp. 263–275.
- [85] T. Murakami and Y. Kawamoto, “Utility-optimized local differential privacy mechanisms for distribution estimation,” in *Proc. of USENIX Security*. USENIX Association, 2019, pp. 1877–1894.
- [86] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, “Privacy risk in machine learning: Analyzing the connection to overfitting,” in *Proc. of CSF*. IEEE, 2018, pp. 268–282.
- [87] B. Jayaraman and D. Evans, “Evaluating differentially private machine learning in practice,” in *Proc. of USENIX Security*. USENIX Association, 2019, pp. 1895–1912.

BIBLIOGRAPHY

- [88] C. Gentry, “Fully homomorphic encryption using ideal lattice,” in *Proc. of STOC 2009*. ACM, 2009, pp. 169–178.
- [89] D. Bogdanov, S. Laur, and J. Willemson, “Sharemind: A framework for fast privacy-preserving computations,” in *Proc. of ESORICS*, ser. LNCS, vol. 5283. Springer, 2008, pp. 192–206.
- [90] D. Demmler, T. Schneider, and M. Zohner, “Aby - a framework for efficient mixed-protocol secure two-party computation,” in *Proc. of NDSS 2015*. Internet Society, 2015, pp. 1–15.
- [91] H. Chaudhari, A. Choudhury, A. Patra, and A. Suresh, “Astra: High throughput 3pc over rings with application to secure prediction,” in *Proc. of CCSW*. ACM, 2019, pp. 81–92.
- [92] N. Chandran, D. Gupta, A. Rastogi, R. Sharma, and S. Tripathi, “Ezpc: Programmable and efficient secure two-party computation for machine learning,” in *Proc. of IEEE EuroS&P 2019*. IEEE, 2019, pp. 496–511.
- [93] A. Patra, T. Schneider, A. Suresh, and H. Yalame, “Aby2.0: Improved mixed-protocol secure two-party computation,” in *Proc. of USENIX Security*. USENIX Association, 2021.
- [94] A. Khedr, G. Gulak, and V. Vaikuntanathan, “Shield: Scalable homomorphic implementation of encrypted data-classifiers,” *IEEE Transactions on Computers*, vol. 65, no. 9, pp. 2848–2858, 2016.
- [95] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in *Proc. of ASIACRYPT*, ser. LNCS, vol. 10624. Springer, 2017, pp. 409–437.
- [96] X. Jiang, M. Kim, K. E. Lauter, and Y. Song, “Secure outsourced matrix computation and application to neural networks,” in *Proc. of CCS 2018*. ACM, 2018, pp. 1209–1222.
- [97] C. He, M. Xue, J. Wang, and W. Liu, “Embedding backdoors as the facial features: Invisible backdoor attacks against face recognition systems,” in *Proc. of TURC 2020*. ACM, 2020, pp. 231–235.

- [98] M. Xue, C. He, J. Wang, and W. Liu, “Backdoors hidden in facial features: a novel invisible backdoor attack against face recognition systems,” *Peer-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1458–1474, 2021.
- [99] S. Li, H. Liu, T. Dong, B. Z. H. Zhao, M. Xue, H. Zhu, and J. Lu, “Hidden backdoors in human-centric language models,” in *Proc. of CCS 2021*. ACM, 2021, pp. 3123–3140.
- [100] Y. Matsuo and K. Takemoto, “Backdoor attacks to deep neural network-based system for covid-19 detection from chest x-ray images,” *Applied Sciences*, vol. 11, no. 20, pp. 1–10, 2021.
- [101] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, “Model-reuse attacks on deep learning systems,” in *Proc. of CCS 2018*. ACM, 2018, pp. 349–363.
- [102] Z. Liu, F. Li, Z. Li, and B. Luo, “Loneneuron: A highly-effective feature-domain neural trojan using invisible and polymorphic watermarks,” in *Proc. of CCS 2022*. ACM, 2022, pp. 2129–2143.
- [103] D. Wu and Y. Wang, “Adversarial neuron pruning purifies backdoored deep models,” in *Proc. of NeurIPS 2021*, vol. 34. Curran Associates, Inc., 2021, pp. 16 913–16 925.
- [104] K. Doan, Y. Lao, and P. Li, “Backdoor attack with imperceptible input and latent modification,” in *Proc. of NeurIPS 2021*, vol. 34. Curran Associates, Inc., 2021, pp. 18 944–18 957.
- [105] T. A. Nguyen and A. Tran, “Input-aware dynamic backdoor attack,” in *Proc. of NeurIPS 2020*, vol. 33. Curran Associates, Inc., 2020, pp. 3454–3464.
- [106] W. Sun, X. Jiang, S. Dou, D. Li, D. Miao, C. Deng, and C. Zhao, “Invisible backdoor attack with dynamic triggers against person re-identification,” *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 307–319, 2024.
- [107] J. Zhang, C. Dongdong, Q. Huang, J. Liao, W. Zhang, H. Feng, G. Hua, and N. Yu, “Poison ink: Robust and invisible backdoor attack,” *IEEE Transactions on Image Processing*, vol. 31, pp. 5691–5705, 2022.

BIBLIOGRAPHY

- [108] Z. Zhao, X. Chen, Y. Xuan, Y. Dong, D. Wang, and K. Liang, “Defeat: Deep hidden feature backdoor attacks by imperceptible perturbation and latent representation constraints,” in *Proc. of CVPR 2022*, 2022, pp. 15 213–15 222.
- [109] N. Zhong, Z. Qian, and X. Zhang, “Imperceptible backdoor attack: From input space to feature representation,” in *Proc. of IJCAI 2022*, L. D. Raedt, Ed. IJCAI Organization, 2022, pp. 1736–1742.
- [110] Z. Tian, L. Cui, J. Liang, and S. Yu, “A comprehensive survey on poisoning attacks and countermeasures in machine learning,” *ACM Computing Surveys*, vol. 55, no. 8, 2022.
- [111] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *Proc. of NIPS 2014*. MIT Press, 2014, pp. 3320–3328.
- [112] A. Schwarzschild, M. Goldblum, A. Gupta, J. P. Dickerson, and T. Goldstein, “Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 9389–9398.
- [113] S. Li, M. Xue, B. Z. H. Zhao, H. Zhu, and X. Zhang, “Invisible backdoor attacks on deep neural networks via steganography and regularization,” *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2088–2105, 2021.
- [114] R. Ning, J. Li, C. Xin, and H. Wu, “Invisible poison: A blackbox clean label backdoor attack to deep neural networks,” in *Proc. of INFOCOM 2021*. IEEE, 2021, pp. 1–10.
- [115] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, “Neural cleanse: Identifying and mitigating backdoor attacks in neural networks,” in *IEEE S&P 2019*. IEEE, 2019, pp. 707–723.
- [116] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proc. of CVPR 2018*. IEEE, 2018, pp. 586–595.

- [117] Y. Jo, S. Yang, and S. J. Kim, “Investigating loss functions for extreme super-resolution,” in *Proc. of CVPR Workshops 2020*. Computer Vision Foundation / IEEE, 2020, pp. 1705–1712.
- [118] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proc. of ICML 2015*, 2015, pp. 448–456.
- [119] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. M. Molloy, and B. Srivastava, “Detecting backdoor attacks on deep neural networks by activation clustering,” in *Proc. of SafeAI 2019*, 2019.
- [120] D. Tang, X. Wang, H. Tang, and K. Zhang, “Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection,” in *Proc. of Usenix Security 2021*. USENIX Association, 2021, pp. 1541–1558.
- [121] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, “Strip: A defence against trojan attacks on deep neural networks,” in *Proc. of ACSAC 2019*. ACM, 2019, pp. 113–125.
- [122] Y. Zeng, S. Chen, W. Park, Z. Mao, M. Jin, and R. Jia, “Adversarial unlearning of backdoors via implicit hypergradient,” in *International Conference on Learning Representations*, 2022.
- [123] F. R. Hampel, “The influence curve and its role in robust estimation,” *Journal of the American Statistical Association*, vol. 69, no. 346, pp. 383–393, 1974.
- [124] L. Zhu, R. Ning, C. Wang, C. Xin, and H. Wu, “Gangsweep: Sweep out neural backdoors by gan,” in *Proc. of MM 2020*. ACM, 2020, pp. 3173–3181.
- [125] S. U. Hussain, M. Javaheripi, M. Samragh, and F. Koushanfar, “Coinn: Crypto/ml codesign for oblivious inference via neural networks,” in *Proc. of CCS*. ACM, 2021, pp. 3266–3281.
- [126] C. Berry and N. Komninos, “Efficient optimisation framework for convolutional neural networks with secure multiparty computation,” *Computers & Security*, vol. 117, p. 102679, 2022.

BIBLIOGRAPHY

- [127] A. Kumar, R. Tourani, M. Vij, and S. Srikanteswara, “Sclera: A framework for privacy-preserving mlaas at the pervasive edge,” in *Proc. of IEEE PerCom 2022 Workshops*, 2022, pp. 175–180.
- [128] T. Shen, J. Qi, J. Jiang, X. Wang, S. Wen, X. Chen, S. Zhao, S. Wang, L. Chen, X. Luo, F. Zhang, and H. Cui, “SOTER: Guarding black-box inference for general neural networks at the edge,” in *Proc. of USENIX ATC*. USENIX Association, 2022, pp. 723–738.
- [129] European Commission, “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 (General Data Protection Regulation) (Text with EEA relevance),” 2016.
- [130] M. Malekzadeh, A. Borovykh, and D. Gündüz, “Honest-but-curious nets: Sensitive attributes of private inputs can be secretly coded into the classifiers’ outputs,” in *Proc. of CCS*. ACM, 2021, pp. 825–844.
- [131] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proc. of CVPR 2018*. IEEE Computer Society, 2018, pp. 586–595.
- [132] F. Tramér, F. Zhang, and A. Juels, “Stealing machine learning models via prediction apis,” in *Proc. of USENIX Security 2016*. USENIX Association, 2016, pp. 601–618.
- [133] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, “High accuracy and high fidelity extraction of neural networks,” in *Proc. of USENIX Security*. USENIX Association, 2020, pp. 1345–1362.
- [134] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *Proc. of IEEE S&P 2018*. IEEE Computer Society, 2017, pp. 3–18.

- [135] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, “MI-leaks: Model and data independent membership inference attacks and defenses on machine learning models,” in *Proc. of NDSS 2019*. The Internet Society, 2019.
- [136] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, “Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing,” in *Proc. of USENIX Security 2014*. USENIX Association, 2014, pp. 17–32.
- [137] B. Z. H. Zhao, A. Agrawal, C. Coburn, H. J. Asghar, R. Bhaskar, M. A. Kaafar, D. Webb, and P. Dickinson, “On the (in)feasibility of attribute inference attacks on machine learning models,” in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2021, pp. 232–251.
- [138] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, “Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers,” *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.
- [139] M. P. Parisot, B. Pejo, and D. Spagnuolo, “Property inference attacks on convolutional neural networks: Influence and implications of target model’s complexity,” *arXiv preprint arXiv:2104.13061*, 2021.
- [140] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” *Advances in neural information processing systems*, vol. 32, 2019.
- [141] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, “Inverting gradients-how easy is it to break privacy in federated learning?” *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 937–16 947, 2020.
- [142] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, “See through gradients: Image batch recovery via gradinversion,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 337–16 346.

BIBLIOGRAPHY

- [143] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, “Evaluating gradient inversion attacks and defenses in federated learning,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 7232–7241, 2021.
- [144] C. Song and V. Shmatikov, “Overlearning reveals sensitive attributes,” in *Proc. of ICLR*, 2020.
- [145] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, “Prada: Protecting against dnn model stealing attacks,” in *Proc. of EuroS&P 2019*. IEEE, 2019, pp. 512–527.
- [146] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [147] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, “Evaluating gradient inversion attacks and defenses in federated learning,” in *Advances in Neural Information Processing Systems*, 2021.
- [148] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, “See through gradients: Image batch recovery via gradinversion,” in *Proc. of CVPR*. IEEE Computer Society, 2021, pp. 16 332–16 341.
- [149] D. E. Rumelhart and J. L. McClelland, “Learning internal representations by error propagation,” in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, 1987, pp. 318–362.
- [150] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, “Unsupervised learning of invariant feature hierarchies with applications to object recognition,” in *Proc. of CVPR*. IEEE Computer Society, 2007, pp. 1–8.
- [151] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.