Building Trustworthy Artificial Intelligence of Things Systems in Adversarial Environments

Shanghao Shi

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Wenjing Lou, Chair
Yi Shi, Co-chair
Y. Thomas Hou
Jin-Hee Cho
Bo Ji
Yingying Chen

July 23, 2025
Arlington, Virginia

Keywords: Security and Privacy, Machine Learning, Internet of Things.

Building Trustworthy Artificial Intelligence of Things Systems in Adversarial Environments

Shanghao Shi

ABSTRACT

The recent decade has witnessed a great explosion of artificial intelligence and the Internet of Things technologies. They have revolutionized people's daily lives, improving convenience, efficiency, and connectivity in previously unimaginable ways. Among all broad topics related to AI and IoT, the Artificial Intelligence of Things, abbreviated as AIoT, focuses specifically on the convergence of these two exciting technologies, combining AI's intelligence with IoT's connectivity and data-gathering capabilities. Until now, numerous AIoT applications have been developed, such as smart homes, autonomous driving, smart wearable devices, and automated medical diagnosis. In this dissertation, we investigate the critical security and privacy problems in AIoT systems. Because the AIoT system comprises two fundamental components, including IoT networks and AI algorithms, we naturally decompose our research into two parts: IoT network security and AI security.

For IoT security, we explore new network attacks against the critical IoT network protocols and propose defense mechanisms to enhance the IoT infrastructure. In Chapter 2, we propose a novel network timing attack that desynchronizes and disables the chosen victim nodes in the IoT networks. Our attack compromises the precision time protocol, which is the de facto network timing protocol in time-sensitive IoT networks. In this chapter, we also introduce a defense mechanism based on network redundancy to prevent minority malicious nodes. In Chapter 3, we present the design of a trustworthy and verifiable spectrum sharing

system leveraging blockchain technology. This system aims to defend against malicious participants and securely record their behaviors. We focus on spectrum sharing as it promotes more efficient utilization of spectrum resources, thereby enhancing the communication infrastructure of IoT networks.

For AI security, we first focus on federated learning, or FL, a leading distributed learning paradigm built upon decentralized networks. We investigate privacy attacks against FL from a red team perspective to better understand and expose potential system vulnerabilities. We then explore adversarial attacks on multimodal diffusion models, motivated by the growing popularity of generative AI technologies. In Chapter 4, we introduce our customized model inversion attack against the medical FL systems. Our attack can reconstruct sensitive real-life COVID-19 X-ray images, brain tumor MRI images, and clinical text records, demonstrating its applicability and severity on practical medical systems. In Chapter 5, we present our novel model inversion attack named Scale-MIA against secure FL systems. This attack can reverse the shared model updates between the FL server and clients back to local training samples, challenging the fundamental privacy-preserving property of the FL systems. In Chapter 6, we introduce our novel adversarial attacks against multimodal diffusion models. Our attack adds customized imperceptible perturbations to the image prompts and can mislead the diffusion model from generating any attacker-chosen content, including NSFW content.

We hope this work can offer insights into the fundamental security and privacy research of the AIoT systems.

Building Trustworthy Artificial Intelligence of Things Systems in Adversarial Environments

Shanghao Shi

GENERAL AUDIENCE ABSTRACT

This work presents a Ph.D. dissertation on building trustworthy Artificial Intelligence of Things, or AIoT, systems in adversarial environments. Over the past decade, the AIoT, an emerging paradigm that integrates AI's decision-making capabilities with IoT's data collection and communication infrastructure, has led to innovations that are transforming our everyday life through improved convenience, efficiency, and connectivity.

However, as AIoT systems become more prevalent, ensuring their security and privacy becomes increasingly critical. This research addresses these concerns by examining two foundational components of AIoT: IoT network infrastructure and AI algorithms.

For IoT security, the work investigates novel attack vectors targeting essential communication protocols and proposes defense mechanisms to enhance system resilience. Chapter 2 introduces a new timing-based network attack that disrupts synchronization in time-sensitive IoT systems, along with a defense mechanism using network redundancy. Chapter 3 presents a blockchain-based spectrum sharing system designed to detect and deter malicious participants, ensuring transparent and efficient use of wireless spectrum resources.

For AI security, the first focus is on federated learning, or FL, a distributed machine learning framework widely adopted in decentralized AIoT environments. Chapter 4 demonstrates a targeted attack on medical FL systems, successfully reconstructing sensitive information such as COVID-19 chest X-rays, brain tumor MRIs, and clinical text records. Chapter 5 describes Scale-MIA, a model inversion attack that reveals private training data from shared

model updates, undermining the privacy guarantees of FL systems. Then we investigate the security of the emerging generative AI technology. Chapter 6 introduces our proposed adversarial attack against the multimodal diffusion models. Our attack can mislead the diffusion model from generating any attacker-chosen content, including NSFW images. Overall, this research aims to deepen the understanding of security and privacy risks in AIoT systems and to provide practical solutions for mitigating those risks.

Dedication

Dedicated to my mom and dad.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor – Prof. Wenjing Lou. Prof. Lou is not only a world-renowned researcher but also the most dedicated and inspiring educator I have ever met. Prof. Lou led me to the fascinating field of cybersecurity and provided me with her generous help and thoughtful guidance throughout my Ph.D. journey. Prof. Lou has taught me how to identify impactful research problems, solve them with rigorous methods, and finally deliver them as excellent research works. Working with Prof. Lou is the most invaluable experience of my academic life, and this experience has motivated me to pursue cybersecurity research as a lifelong commitment.

I would like to extend my sincere gratitude to my co-advisor – Prof. Yi Shi. Prof. Shi is a remarkable researcher and wise mentor who kindly offered me thoughtful insights and generous help during my Ph.D. study. I am also deeply grateful to my committee members, Prof. Y. Thomas Hou, Prof. Bo Ji, Prof. Jin-hee Cho, and Prof. Yingying Chen, for their valuable comments and constructive suggestions on my research.

I would like to thank my colleagues at the Complex Network and Security Research (CNSR) lab at Virginia Tech for the exceptional research discussions and collaborations. I learned a lot from our engaging discussions.

Last but not least, I would like to thank my parents. Their unwavering support and unconditional love have made it possible for me to pursue and complete my Ph.D. journey.

Funding Acknowledgments

This work was supported in part by the Office of Naval Research under grants N00014-24-1-2730 and N00014-19-1-2621, the National Science Foundation under grants 2331936, 2154929, 1916902, 1837519, and the Virginia Commonwealth Cyber Initiative (CCI).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

AIoT, or Artificial Intelligence of Things, refers to the integration of Artificial Intelligence (AI) with the Internet of Things (IoT) technologies. AIoT leverages the connectivity of IoT devices and enhances their functionalities through AI-driven data analytics and intelligent decision-making. Such a combination has cultivated many promising technologies, such as federated learning [122] and has been widely used in many applications such as smart homes [132, 193], healthcare [141, 188, 209], and autonomous driving [109, 128].

In this dissertation, we focus on the critical security & privacy aspect of the AIoT systems. In recent years, there has been a huge number of security attacks against various AIoT systems. Based on the attack interfaces, we categorize the adversarial attacks against the AIoT systems into two types: IoT networking attacks and machine learning attacks.

For the IoT networking attacks, in 2016, a powerful Botnet attack named Mirai attack [181] occurred in the United States. This attack focused on jeopardizing the network connection within the U.S. and successfully disconnected about half of the IoT devices in the U.S. In 2018, the researchers proposed an attack named BlackIoT that manipulates the power demand within the smart grid IoT network, potentially causing a power outage [170]. In 2022, a follow-up work named MadIoT 2.0 shows that the attacker with more knowledge about the target can launch more sophisticated and catastrophic attacks against the power system [160]. In 2020, an attack named ReVoLTE was proposed to break the LTE system and eavesdrop on encrypted LTE calls between users [155]. In 2023, an evasion attack

was proposed to compromise the smart home devices' physical fingerprints to control their physical operations [136]. All these attacks have demonstrated the urgent need to investigate and address the vulnerabilities inherent in IoT networks.

Therefore, in the first part of this dissertation, we focus on addressing the IoT network security challenges. We work from both the attack and defense perspectives. From the attack perspective, we identify new attack vectors against the IoT networks from the red team's perspective, aiming to find unexplored system vulnerabilities. From the defense perspective, we design new defense mechanisms to enhance the IoT network infrastructure against network adversaries.

In Chapter 2, we propose a novel secure network timing mechanism named MS-PTP to enhance the current de facto network timing protocol – the Precision Time Protocol (PTP) [163]. We first propose an insider-time-shifting attack launched by a compromised insider node. This attack enables a malicious insider node to self-elect as the time master and desynchronize the time of an arbitrary victim node. We implement the attack on real PTP implementations, including PTPD and LinuxPTP, to validate its effectiveness on real systems. The results show that the attack can successfully desynchronize the victim nodes and jeopardize the operation of time-sensitive IoT applications such as robots. As a countermeasure, we propose Multi-Source Precision Time Protocol (MS-PTP), a defense mechanism leveraging redundant time sources to counter minority malicious nodes. The core of MS-PTP is a Byzantine resilient input aggregation method that takes both honest and malicious timing measurements as inputs and calculates an aggregated value close to the honest measurements. This aggregated value is mathematically guaranteed to remain within a deterministic error bound relative to the ground-truth timing measurement, provided that the number of compromised time sources does not exceed one-third of the total population. We evaluate our proposed defense against the time-shifting attack, and the results show that

our proposed mechanism achieves excellent performance for defending against these attacks.

In Chapter 3, we focus on investigating the security threats of the spectrum sharing paradigm [164]. The spectrum sharing paradigm enables more efficient usage of the spectrum resources, boosting the fundamental communication capability of the IoT infrastructure. In this work, we identify that the inter-SAS coordination procedure is particularly vulnerable within the whole paradigm. We find that the current industrial standard coordination procedure has no built-in security mechanism and is almost open to malicious and selfish attackers. For example, a selfish attacker can easily lie about its spectrum demands and gain excessive spectrum resources. To address this threat, we propose a blockchain-based, decentralized inter-SAS coordination mechanism. Our mechanism guarantees that all servers reach a consistent spectrum allocation result, even in the presence of up to one-third compromised Byzantine participants. To ensure verifiability and integrity, all final allocation outcomes and intermediate execution values are recorded on an immutable ledger. In addition, our mechanism offers real-time feedback to spectrum requests, significantly enhancing the efficiency of the spectrum sharing systems. We implement a prototype of our mechanism on a distributed network across the U.S., and the results show that our mechanism achieves a secure spectrum sharing service under low latency and high throughput.

For AI security, many attacks have been proposed to compromise the AI system from different aspects. Adversarial examples are proposed to mislead the machine learning classifier into making wrong predictions through adding imperceptible perturbations to the input images [33, 68, 119]. Backdoor attacks are proposed to insert attacker-chosen backdoors in machine learning models [38, 183, 202]. Model-stealing attacks are proposed to extract the model owner's proprietary models via the predictive API [34, 89, 135, 178]. Membership inference attacks aim to infer whether certain members are contained in the training dataset [126, 167]. From the defense perspective, adversarial training [63] and adversarial purification methods

[123, 131, 174] are proposed to defend against adversarial examples. Byzantine resilient model aggregation methods are proposed to defend against data and model poisoning attacks in federated learning [25, 56, 204]. Model watermarking and fingerprinting methods are proposed to defend against model stealing methods [8, 88, 175]. Differential privacy is introduced to defend against membership inference attacks [6, 51].

In this part, we first focus on federated learning (FL) [96], a distributed learning paradigm that enables a set of clients to collaboratively train machine learning models within a distributed network (e.g., the IoT networks). During the FL training process, all client data remains local, and only the local training gradients and model updates are shared between the training server and clients. In this context, FL is considered to be a privacy-preserving mechanism. However, in this part, we investigate model inversion attacks [61, 65, 138, 190, 205, 211, 212], which aim to reverse the shared model updates back to local training samples, challenging the fundamental privacy-preserving capability of FL systems.

In Chapter 4, we design a model inversion attack against the medical FL systems. Due to its privacy-preserving capability, FL has been widely used in the healthcare domain to enable different hospitals to collaboratively train machine learning models without sharing sensitive patient information [9, 29, 44, 127, 129, 139, 152, 161]. However, in this work, we propose a new attack demonstrating that the medical FL is vulnerable to model inversion attacks. Our attack introduces two specialized modules: the Linear Leakage Module and the Zero Gradient Module. The Linear Leakage Module comprises two carefully designed linear layers capable of reversing gradients back to the inputs. We insert this module before the global model sent to the target client, enabling the reconstruction of its local training data. The second component, the Zero Gradient Module, suppresses gradient propagation by zeroing out its gradients. This module is deployed in front of all non-target clients, ensuring that only the target client's gradients are revealed during model aggregation. As a result,

the attack selectively exposes the training samples of the target client. We implement our attack on both medical images and text datasets. The results demonstrate the effectiveness of our attack in achieving perfect performance in reconstructing real medical records, such as X-Ray images, brain tumor MRI images, and patient text descriptions.

In Chapter 5, we introduce a novel model inversion attack against secure FL systems. Secure FL systems mean that the systems are protected by state-of-the-art secure aggregation protocols [23, 27], which can hide individual model updates and only expose the aggregated model updates to the server. This defense can effectively address the current optimization-based model inversion attacks because they take the individual model updates as the attack inputs, which are hidden under the secure aggregation protocols. However, our proposed attack can reverse even the aggregated model updates back to local training samples, rendering these secure aggregation protocols useless. We investigate the architecture of the machine learning models and find that the linear layers in the latent space are pivotal layers to launch the model inversion attack. Based on this, we decompose the complex inversion task into two steps, including first reversing the parameter updates of the two linear layers back to latent space representations via the linear leakage attack module, and then further reversing the latent space representations back to local samples via pre-trained decoders. Both attack steps only involve closed-form or feed-forward neural network operations, making our attack super efficient. We implement our attack on widely used machine learning datasets, and the results show that our attack can reconstruct hundreds of samples simultaneously with high quality in only a few seconds. Our attack represents the state-of-the-art model inversion attack against secure FL systems and calling for more effective defenses against such an advanced attack.

Our next focus is the security challenges of the emerging generative AI technologies. Particularly, diffusion models such as Stable Diffusion [153], DALL-E [145], Kandinsky [19], and

Midjourney [102] have gained widespread popularity due to their capability of generating high-quality synthesis images. They currently support multimodal input prompts that integrate text with other modalities, such as images and audio, for more accurate and expressive prompt specifications. However, the introduction of new modalities also introduces new attack vectors, making multimodal diffusion models vulnerable to adversarial attacks that can manipulate them into generating attacker-specified content.

In Chapter 6, we introduce a novel adversarial attack named SWAP. SWAP identifies the multimodal adapter [58, 203] as the pivotal attack component and exploits image as the attack modality, avoiding the involvement of the complex diffusion component to reduce the attack overhead. SWAP adds imperceptible perturbations to the image prompts to steer the multimodal adapter toward generating malicious representations close to the targets' representations while keeping them away from NSFW concepts such as adultery and violence via contrastive learning. This helps to bypass the black-box safety filter with a few queries and naturally leads to the generation of the attacker-desired content. We implemented our attack on real Stable Diffusion 1.5 and Stable Diffusion XL models [172]. The experiment results demonstrate that SWAP can bypass the built-in safety filters [146] and mislead both models into generating the attacker-chosen NSFW content with high attack success rates (ASRs) and excellent generation quality.

We summarize the dissertation in Chapter 7. We hope this dissertation can provide readers with insights about how to build secure and trustworthy AIoT systems.

Chapter 2

Protecting Network Timing from Byzantine Attacks

(Copyright Notice[1])

2.1 Introduction

Recently emerged time-sensitive applications, such as autonomous driving and smart grids, usually require a microsecond or sub-microsecond level synchronization between different nodes and failure to achieve these requirements can lead to significant consequences. The Precision Time Protocol (PTP), originally developed by the IEEE 1588 working group [55], is widely regarded as the de facto solution to provide highly precise network synchronization. PTP achieves much higher synchronization accuracy compared to the Network Time Protocol (NTP), the incumbent synchronization service for the Internet. It also offers better flexibility than high-precision GPS-based synchronization, which can only work reliably with outdoor GPS antennas.

[1]This chapter previously appeared as a part of a conference paper published in ACM WiSec 2023. ©2023 Copyright held by the owner/author(s). Reprinted, with permission, from Shanghao Shi, Yang Xiao, Changlai Du, Md Hasan Shahriar, Ao Li, Ning Zhang, Y. Thomas Hou, and Wenjing Lou, "MS-PTP: Protecting Network Timing from Byzantine Attacks," in Proceedings of the 16th ACM conference on security and privacy in wireless and mobile networks (WiSec'23), pages 61-71, 2023 [163].

Real-world Applications of PTP. With the help of PTP, devices in a local network can be synchronized with sub-microsecond accuracy. Currently, PTP has been documented in many industry standards including the 3GPP 5G standard [5], IEEE TSN standard [4], and IEEE smart grid standard [2], and has been used by various time-sensitive networks such as data center networks [133], and industrial automation networks [156]. In telecommunication, PTP is deployed in the backhaul networks to transfer timing information from accurate time servers in the mobile core networks to the base station to achieve the 1.5 μs stringent time synchronization requirements of 5G standards [5, 47]. In high-performance data centers such as Meta's data centers, PTP guarantees that data replicas are perfectly synchronized [133] to gain performance improvement as much as 100 times. For CPS, PTP is also employed in autonomous driving vehicles to synchronize ECUs and sensors in the intra-vehicle networks [1].

Gaps in Existing Efforts on Securing PTP. The original PTP [55], however, was designed two decades ago and did not come with any security mechanism against an adverse network environment. It has been shown that earlier versions of PTP are susceptible to various attacks launched by any node in the network, such as message spoofing and replay attacks [11, 45, 84], for its lack of proper authentication mechanism. In response, the latest version of PTP [3] recommends using symmetric key-based authentication mechanisms to counter network adversaries. This mechanism is supported by the latest IETF's PTP key management standard that helps distribute and manage secret keys for PTP nodes [105]. Recent work further argues that symmetric key-based authentication mechanisms are not able to address identity spoofing attacks and need to be replaced by an efficient elliptic-curve and public-key-based cryptography mechanism [84].

While these authentication mechanisms can effectively protect against network packet manipulations using secure communication channels, they cannot defend against malicious or

Byzantine insiders. This is because, regardless of how well individual nodes are protected, no system is perfectly secure. In many cases, especially IoT devices or swarm robots, it is quite challenging to ensure all of them are secure, especially since some of them can be physically captured and tampered with by the adversary [15]. Worse yet, the aforementioned cryptographical mechanisms are rarely implemented in popular PTP implementations such as PTPD [82] and linuxPTP [111], introducing more opportunities for attackers.

Analyzing the Impact of Compromised Nodes in Secure Time Synchronization Network. To develop an effective defense against malicious insiders, it is often necessary to have insights into the attack mechanisms. Therefore, the first half of the contribution focuses on the security analysis of the time synchronization network from the perspective of a malicious insider. We found that, even if the communication channel is secure, it remains possible for the adversary to arbitrarily shift the clock of any targeted victim node within the network, using only a single malicious insider. The key idea is to exploit the weakness in the election mechanism for the unique grandmaster (GM) to self-elect as the master to attack the rest of the nodes in the network. To validate our finding, we demonstrated the attack on the two most popular PTP implementations, PTPD [82] and linuxPTP [111], in an IoT testbed. To further show the potentially catastrophic sequence of the attack, we demonstrate the consequence of our attack on a Turtlebot 3 robotic platform. Turtlebot 3 robot relies on synchronized sensor inputs to realize its localization and control functions. When sensors are de-synchronized, the physical world perceptions from different modalities also become desynchronized, leading to errors in the localization and path planning process.

Our Proposed Defense. Our security analysis discovers a key vulnerability that can be exploited by a malicious insider—existing PTP protocols only make use of a single time source. To defend against the attack, we propose MS-PTP, a Byzantine-resilient network time synchronization mechanism to safeguard the dependability and accuracy of PTP timing

against a malicious insider who attempts to dis-synchronize clocks of honest clients. MS-PTP leverages redundancy (i.e., multiple time sources) to increase the PTP system's resiliency (i.e., its tolerance to Byzantine failures up to a certain threshold). The redundant time sources provide each client with additional measurements for calculating its clock drift/offset in each synchronization round. However, naively adding time sources (such as taking the average) does necessarily improve the accuracy, since the impact from the malicious input is not bounded. To address this problem, we propose a novel Byzantine-resilient measurement aggregation scheme for the client to obtain a robust estimate of its clock drift/offset, given that f out of the n measurements are potentially Byzantine and $n \geq 3f + 1$. The estimation error of MS-PTP is bounded by $\sqrt{2}$ times the measurement uncertainty of honest PTP sessions.

Evaluation. We implemented a prototype MS-PTP system on our IoT testbed. We validated its resilience against different Byzantine attacks and evaluated its computational efficiency. The results show that MS-PTP is able to retain microsecond level time synchronization accuracy even in the presence of an adaptive attacker with the full knowledge of our defense mechanism. MS-PTP maintains this accuracy when the network size grows to 30, which covers all the typical deployment settings of PTP networks. Moreover, we implemented MS-PTP over network time protocol (NTP), GPS-based time synchronization method, and a mixture of them to complement the PTP-only study. We observed similar Byzantine resiliency performance of MS-PTP for all these protocols, showing the generality of decentralized design across different time synchronization technologies of different scales. Lastly, to understand the theoretical guarantee of the proposed protocol, we've developed proven bounds on the time synchronization error. In summary, this chapter makes the following contributions:

- We analyze PTP's vulnerability from an insider adversary perspective. To show the

feasibility of the attack, we demonstrate the attack on two popular open software implementations of the PTP protocol in an IoT network testbed. We further demonstrate this attack on a physical indoor robot.

- To thwart the threat of malicious insiders, We propose MS-PTP, a Byzantine-resilient network time synchronization scheme, as an extension to the existing PTP. MS-PTP features a robust measurement aggregation scheme that leverages time crowdsourcing to produce a robust time estimation with a small bounded error.

- We developed a rigorous proof of the correctness of our aggregation mechanism and showed that its accuracy is non-parametric of the population of Byzantine time sources (i.e., fixed error bound). MS-PTP can scale to larger networks and outperforms the current state-of-the-art mechanisms.

- We implemented a proof-of-concept MS-PTP system and evaluated its performance in various network and attack scenarios. The results show that MS-PTP achieves excellent synchronization accuracy and is compatible with different time synchronization protocols.

2.2 Research Motivation

2.3 PTP: Overview and Vulnerabilities

2.3.1 PTP Overview

PTP establishes a tree-structured, master-slave network hierarchy, as shown in Fig. 2.1, to fulfill the time synchronization function. In this architecture, one node, supposedly with

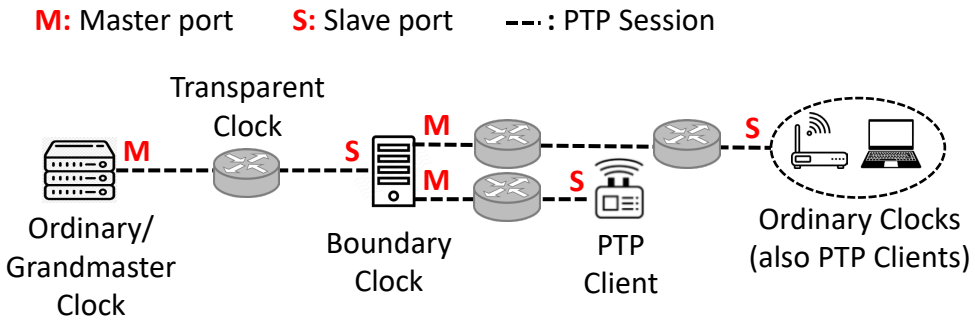


Figure 2.1: PTP network hierarchy.

the most accurate clock, is elected to serve as the unique time server—the *grandmaster clock* (GM). Other intermediate routers and servers are known as *boundary clocks* (BCs) or *transparent clocks* (TCs). The clients located in the end-leaf nodes are *ordinary clocks* (OCs) in PTP terminology. Under normal operating conditions, the GM periodically delivers timing information to downstream clients through the two-way time transfer (TWTT) mechanism.

Elect the Best Time Source. Upon initialization, the PTP standard specifies all or at least all master candidate devises to continuously broadcast a specific type of PTP message, named the *ANNOUNCE message* on UDP port 320, that contains essential clock accuracy and stability information in its payload. When these messages are received, PTP nodes use the *best master clock algorithm* (BMCA), a clock quality comparison algorithm specified by the PTP standard to decide pairwise master-slave relationship. The master candidate device will stop broadcasting their *ANNOUNCE messages* when they hear from a better clock in the same domain. By doing mutual comparison recursively, the best time source who beats all the other clocks survives to be the grand-master (GM) clock, which becomes the only one that is still broadcasting his *ANNOUNCE messages*. When new nodes join the network, they can obtain current GM's information through reading the GM's *ANNOUNCE messages* and if they have a better clock, can start a new round of this election process.

Two-way Time Transfer. After the time source is elected (i.e. the GM uniquely and stably sends *ANNOUNCE messages* for a certain period), timing information will be periodically transferred from the server to clients following the *two-way time transfer* protocol flow, as fig. 2.2 has depicted. Within one synchronization round, four accurate hardware timestamps, t_1 to t_4, and two cumulative residence times, c_1 and c_2 are accurately recorded to measure the clock offset θ and clock drift rate δ of the slave node. t_1 refers to the sending time of the SYNC message at the master port and t_2 refers to its reception time at the slave port. Similarly, t_3 is the time that DELAY_REQUEST is sent by the slave port and t_4 refers to the time of its reception at the Master. c_1 and c_2 are measured by down-link and up-link TCs with each one adding its residence time cumulatively to the correction fields of on-the-fly messages. PTP assumes a symmetric path delay δ unless the asymmetry between up-link and down-link path delay is known. The offset θ and path delay δ can be derived as:

$$\begin{aligned} \theta &= \frac{(t_2 - t_1 - c_1) - (t_4 - t_3 - c_2)}{2} \\ \delta &= \frac{(t_2 - t_1 - c_1) + (t_4 - t_3 - c_2)}{2} \end{aligned} \tag{2.1}$$

For the clock drift rate $\beta - 1$, PTP assumes a constant clock rate β and leverages the offset measured by the current round $\theta(t_{m_1})$ and l^{th} round later $\theta(t_{m_l})$ to get it. l is an adjustable parameter chosen by the clients.

$$\beta = \frac{\theta(t_{m_l}) - \theta(t_{m_1})}{t_{m_l} - t_{m_1}} \tag{2.2}$$

2.3.2 PTP Vulnerabilities

The vanilla version of PTP was designed decades ago and has no built-in security mechanism. Recently a revised version of PTP [3] discusses several cryptographic authentication

mechanisms, including group key-based direct authentication and TELSA-based delayed authentication [140] to support message authentication. However, these security mechanisms are unable to address malicious insiders [84] and they are not widely adopted and implemented, making PTP almost completely open to all kinds of attacks. In this work, we categorize the attackers into two main classes: network adversaries, who can observe and modify PTP traffics but do not have essential secret keys or credentials to break the cryptography primitives; and malicious insiders, who are compromised legitimate participants of PTP networks.

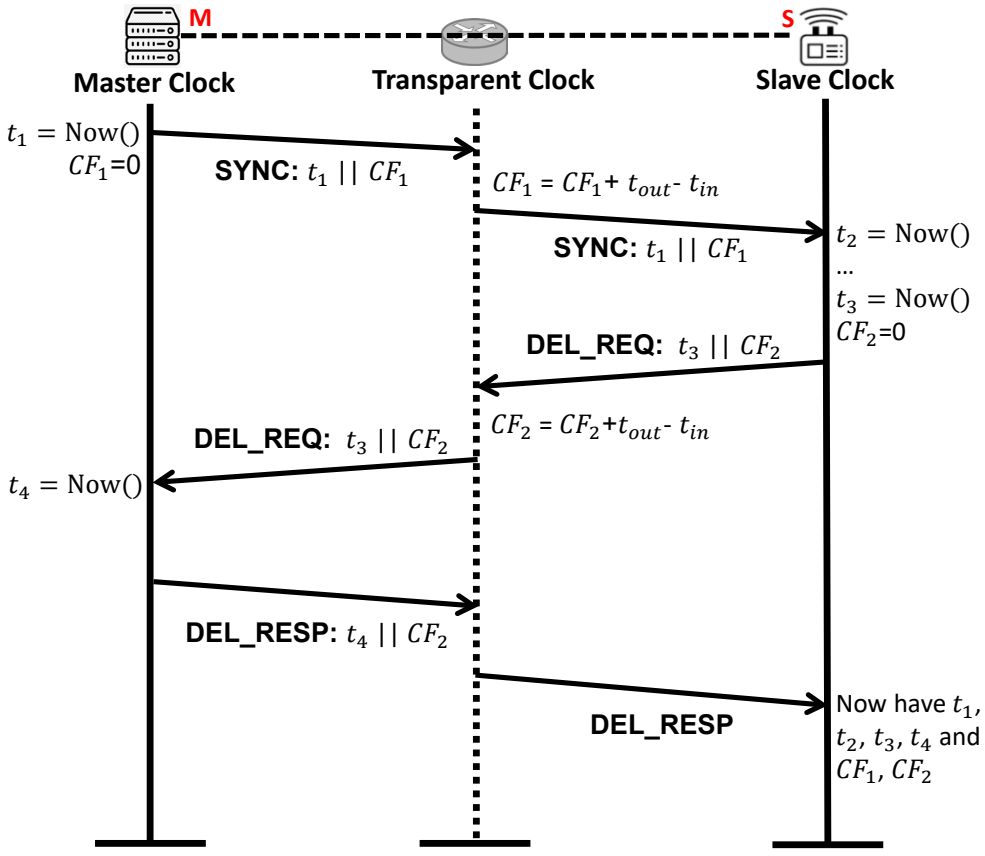


Figure 2.2: PTP two-way time transfer (TWTT) with one transparent clock.

Network Adversaries. Upon initial design, PTP adopted no security mechanism and can be easily disrupted by simple network-level attacks such as message spoofing, intercep-

tion, and modification. [10, 45] introduce several practical fatal network-level time-shifting attacks and validate their feasibility over real-life PTP networks. The symmetric key-based authentication mechanism proposed by the revised standard is also vulnerable to identity spoofing attacks such as rogue grand master attacks [84]. To address all these issues, [84] proposes an elliptic curve-based and public-key-based authentication scheme to establish the authenticity of network clocks. This mechanism can effectively rule out network adversaries by adding small additional overhead to the system.

Malicious Insiders. We consider the malicious insiders to be compromised legitimate participants that possess enough secret keys to bypass the cryptography authentication mechanisms. They may be compromised servers that generate malicious messages from the very beginning or compromised man-in-the-middle (MitM) attackers that modify on-the-fly PTP packets. They cannot be prevented by cryptography methods and pose a great threat to reliable PTP operation. Under the current single-source, tree-structured architecture, it is nearly impossible to detect these attackers cause the downstream clients give full trust to upstream servers and intermediate nodes and there is no way to detect them if they become malicious. In this work, we focus on addressing the most challenging malicious insiders.

2.4 Attack Demonstration

2.4.1 Experiment Setting

In this section, we demonstrate our experiment about how to shift the time of a victim node on a real IoT testbed. The testbed, as shown in figure 2.3, contains three nodes including one server node and two client nodes. The server node consists of one Raspberry Pi 4

device and one plug-in GPS hat. The GPS hat can synchronize its clock with the satellites and transfer this nano-second level accurate timing to the Raspberry Pi board, making it a proper time source. The two client nodes are standard Raspberry Pi 4 boards and all three nodes are interconnected via Ethernet. For PTP software, we choose the commonly used implementations – PTPD [82] and LinuxPTP [111] as the victim implementations. PTPD is a classic and well-received PTP daemon on the Linux operating system and supports an older version of PTP standard [55]. LinuxPTP is an implementation of newer PTP standard [3] and supports more profiles. In this work, we launched our attack over the default, unicast, and telecommunication profiles of LinuxPTP.

Attack Threat Model. The attacker is assumed to compromise one node in the time synchronization network, either the server node or a slave node. The attacker shall be able to monitor inbound PTP traffic, get access to the security credentials he received or assigned, and generate and transmit malicious PTP packets. To better understand PTP time-shifting attacks, we adopt a rather weak attack model as the compromised device is a client node because the compromised server and MitM attackers are too strong and it is straightforward and trivial to launch time-shifting attacks with such strong attackers, since these attackers can directly generate malicious packets. We are going to investigate the following four questions in this section: *(a)* Can a compromised client node jeopardize the operation of others in the current PTP networks? *(b)* If so, to what extent can they infect the time of the victim node? *(c)* Can the current defense mechanisms counter these attackers? *(d)* Will our timing attack cause consequences to a real system?

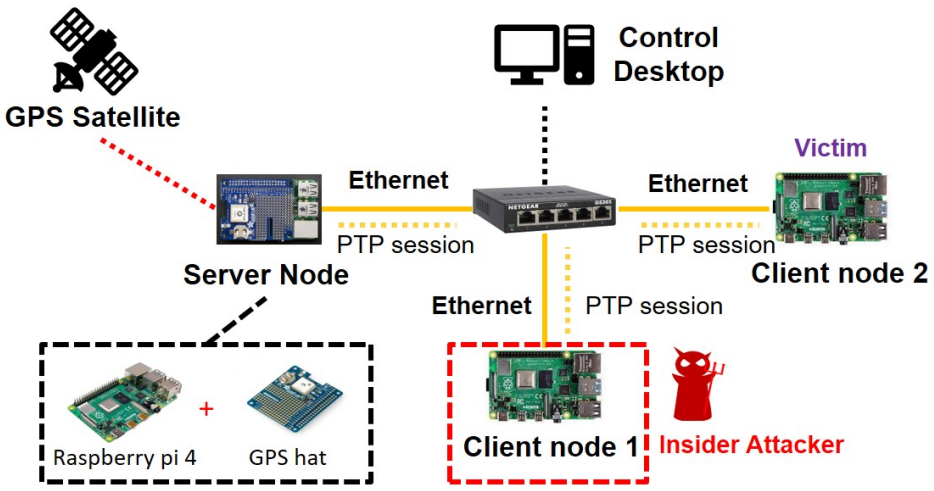


Figure 2.3: PTP testbed.

2.4.2 Insider Time Shifting Attack

We present an insider time-shifting attack launched by a compromised client (the client node 1 in the testbed). The attacker's general attack strategy is first to elect himself to build up a fake network hierarchy and then craft malicious PTP packets to shift the time of a chosen victim node. The attack can be carried out in the following four steps:

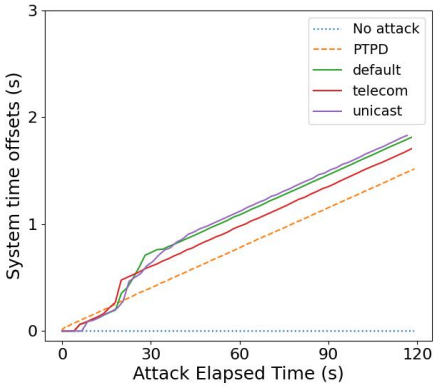
- **Phase 1: Clock Information Extraction.** To know the current GM's clock quality, the attacker n_{adv} reads the current transmitting ANNOUNCE messages ANN_h on its UDP port 320 and extracts the current GM's clock quality information q_h in message payloads.

- **Phase 2: Priority Inversion.** The attacker generates a fake clock quality payload q_{adv} that has higher clock quality than the legitimate q_h according to the BMCA-defined clock quality comparison rules. The attacker can increase the clock priority by one or label the clock type as an ultra-high precision atomic clock. The attacker then ensembles a malicious ANNOUNCE message ANN_{adv} according to the authentication mechanism used in the system. The message contains a PTP header, the fake clock

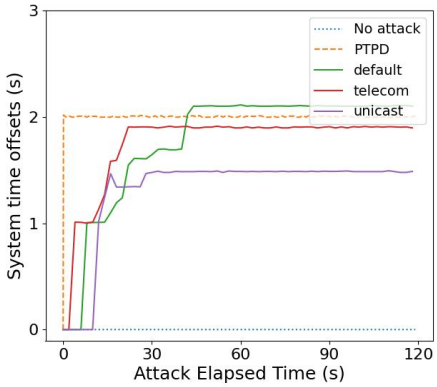
quality payload q_{adv}, and a necessary signature or message authentication code used to pass the authentication process. If the system uses digital signature-based method, $ANN_{adv} = pk_{n_i}||q_{adv}||sig_{sk_{n_i}}$. If uses the symmetric key -based method, $ANN_{adv} = q_{adv}||MAC_{sk_{n_i}}$.

- **Phase 3: Injection and Confirmation.** The attacker broadcasts ANN_{adv} periodically through UDP port 320 at the same speed as a normal grand-master clock. The correct transmitting interval can be obtained either through the PTP configuration files or through continuous monitoring of PTP grandmaster's behaviors. The attacker sniffs the incoming messages on this port at the same time and if the attacker fails to receive any incoming ANNOUNCE message for a certain time interval T_{adv}, there is a high probability that nodes in the network have already taken n_{adv} as the new grand-master. In practice, T_{adv} is not a long time period and we usually observed it to be within 10 seconds in our experiment before the attacker seized the grand-master position.

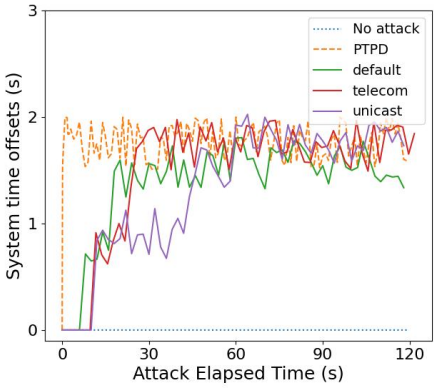
- **Phase 4: Time Shifting.** After the confirmation, the attacker starts the PTP engine to send erroneous information. It follows the normal PTP workflow (i.e. TWTT) but modifies the timestamp field of the SYNC message only to its victim by adding a delay d to t_1. By protocol, the offset measured at the victim device is shifted by $d/2$. This malicious $SYNC_{adv}$ message is also attached with a proper digital signature or message authentication code generated by the shared credentials between the attacker and the victim node.



(a) d as a cumulative delay



(b) d as a constant delay



(c) d as a random delay

Figure 2.4: Time shifting attack results over PTP implementation PTPD and LinuxPTP. The overall trend shows the effective manipulation of the victim clock offset.

2.4.3 Attack Results

We implemented our attack on the testbed with the Python Scapy tool [150], which allows us to sniff and generate arbitrary network packets. We assembled PTP packets according to both the PTP standard and attack methods including crafting proper IP, UDP, and PTP headers, as well as generating PTP payloads and appended signatures/MACs. To evaluate the capability of the attacker, we added the malicious delay d in three different ways—constantly (2s), randomly (mean 1.5s, standard deviation (std) 500ms), and cumulatively

(1.8s per 120s). Figure 2.4 shows the clock offsets of the victim node (client node 2) under different delay-adding methods. We can find that the time (offset) of the victim node was significantly shifted consistently with the way they were manipulated under different PTP implementations and profiles. We observed that the offsets of some LinuxPTP profiles were not immediately shifted when the attacks were launched. There were about 5-10s of delays before they were shifted. But the overall trend of their performance was in line with how they were manipulated. Because the delays we introduced were very large (seconds level) compared to the no-attack clock offsets (microseconds level), the no-attack system offsets became nearly invisible in the figure.

2.4.4 Case Study: Attack Consequence on a Real Robotic Platform.

To further evaluate the effect of time de-synchronization on real systems, we set up a cloud-based indoor delivery system based on the Amazon RoboMaker [12]. The system consists of an Amazon EC2 cloud server and a Turtlebot 3 robot [180] (as shown in Figure. 2.5(a)). The robot receives task missions and sensor inputs from the cloud, allowing the robot to navigate in the physical environment according to the commands from the server. The robot takes inputs from multiple sensors such as a camera, inertial measurement unit, and LiDAR. Each sensor is implemented as a separate node in the network and delivery the data via UDP packets. The sensor data is appended with timestamps, which are used to synchronize among different sensors, achieving a consistent sensing result. To emulate the consequence of our PTP timing attack, we added malicious delays to the sensing inputs. This simulates the situation in which inter-vehicle network synchronization is compromised and the sensor inputs are temporally misaligned. Our goal is to investigate whether the timing misalignments, more precisely submillisecond-level or even subsecond-level misalignments,

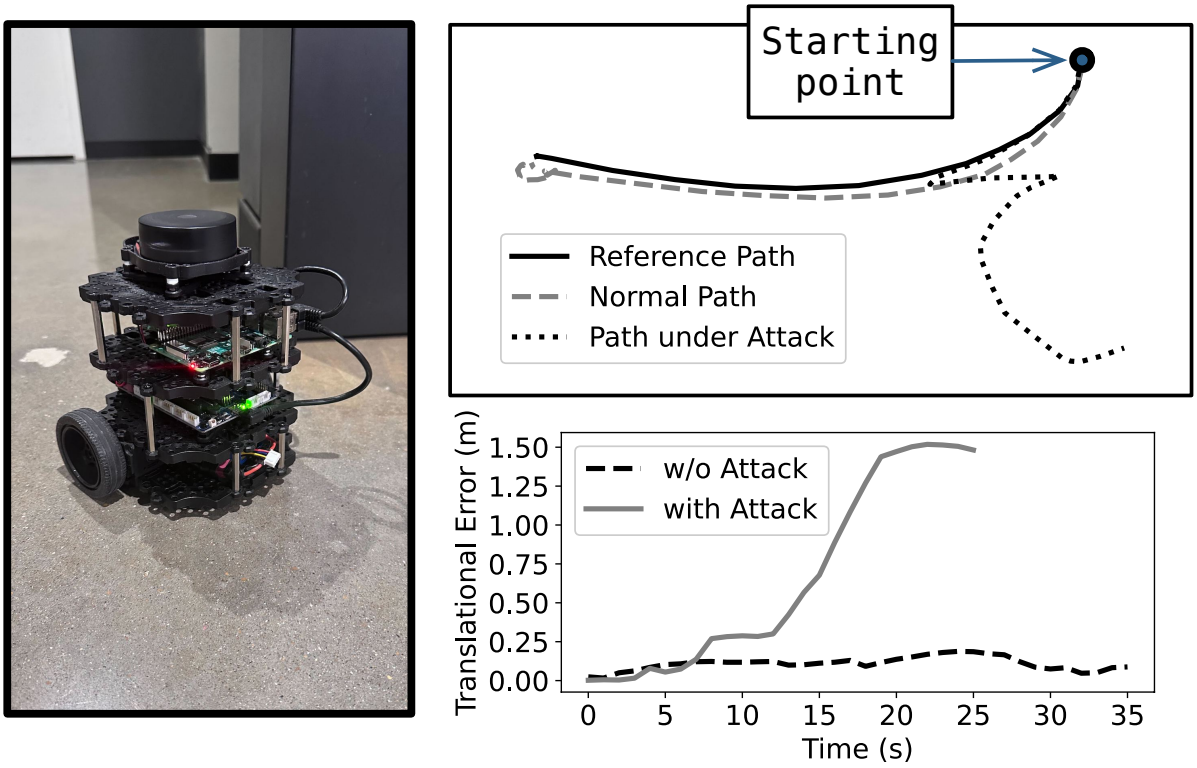
can cause interference with the operation of a real robot.

Figure 2.5(b) shows the trajectory under normal circumstances as well as the trajectory under attack. It can be observed that the robot navigates well without attack. Conversely, the adversary node can easily trick the desynchronization for up to 2 seconds, leading the robot to fail in localizing itself and navigating to the desired location. As shown in figure 2.5(b), with a falsely estimated position, the robot arbitrarily moved around the environment and hit the obstacles.

2.4.5 Attack Analysis and Discussion

With these results, we can answer the questions we have raised before. The compromised client node can shift the time of any victim node arbitrarily by changing the way how it adds delay without any limitation. The reason behind this time-shifting attack is that there lacks a proper clock quality verification mechanism and any node can claim itself to be a high-priority clock source and win the grand-master election process. The current centralized, tree-structured network hierarchy makes things worse for it faces single-point-of-failure and one falsely elected grand master can disrupt the operation of the whole network. While in theory, the system can employ a customized designed public key-based authentication and certificate management system to rule out the spoofing and Sybil attacks [84], they are costly and not used in practice. Furthermore, if we further consider the time servers and MitM nodes are compromised, this defense mechanism cannot help.

Our defense mechanism defends against insider attacks from a new perspective and can be used in parallel with the current cryptography-based authentication mechanism to further boost PTP's security and reliability. We turn to establish a fault-tolerant and robust time synchronization mechanism, yet some insiders are compromised to exhibit arbitrary behav-



(a) Turtlebot3. (b) Control deviations on trajectories.

Figure 2.5: The consequence of de-synchronization attack on Turtlebot3.

iors, we assume the majority of the timing devices to be honest. We adopt the fundamental idea of using multiple sources to counter Byzantine sources from the very famous Byzantine fault tolerance state machine replication (BFT-SMR) scheme. This idea is in line with PTP's developing trend – the newest version has already dictated the use of redundant servers to build robust synchronization services [3]. Fortunately, the current PTP leaves room for the simultaneous existence of multiple servers. This can be achieved by properly configuring the PTP software's configuration file. Each PTP client can open up multiple PTP sessions, with each one having its own domain number. By design, different PTP sessions in different domains do not cause interference to others and within each domain, there can be a unique time server. The remaining critical question is how to select a reliable measurement from a set of potentially malicious ones. We will discuss this problem in the next section.

2.5 System Design

2.5.1 System Model

Countering Byzantine failure is a fundamental problem in distributed networks. A lot of literature has introduced plenty of Byzantine fault-tolerant (BFT) schemes. The most famous BFT schemes may be the *state machine replication* (SMR) solution such as Practical BFT (PBFT) [35] and Tendermint [30]. The fundamental idea of these schemes is to use **redundant nodes** to counter Byzantine minority nodes. We adopt this fundamental idea by designing our Byzantine resilient PTP network using multiple time sources. We also take the threshold assumption from the BFT-SMR schemes, i.e. the number of malicious nodes does not exceed a certain portion of the total population. However, there are key differences in the basic assumptions between classic BFT mechanisms and the problem we are investigating, as we will discuss in the following parts.

Network Model. We assume there are m synchronization sources in the network and each client can connect to n ($n \leq m$) of them to initiate the synchronization procedure. As a result, an individual client is able to receive n independent measurements $(d_1, d_2, \cdots, d_n)$ from n sources in one synchronization round T. Considering the random measurement errors and uncertainty introduced along the communication paths, which is unavoidable in the communication channels, we assume the honest measurements follow the normal distribution of $d_i \sim \mathcal{N}(g, \sigma_i^2)$, where g refers to the correct measurement result under ideal conditions and σ_i^2 refers to the uncertainty level [54]. The key difference between our assumption and the classic BFT-SMR schemes is that the clients in the synchronization problem are not assumed to know the underlying correct measurement g. Therefore, instead of receiving multiple identical correct measurements and a few malicious ones, a time synchronization client is more likely to receive a set of measurements that are different from each other,

making it a difficult and non-trivial problem to counter Byzantine measurements among them.

Uncertainty Level. The standard deviation σ_i of an honest time source is considered significantly smaller than the system's required accuracy level r. In fact, the probability that a measurement from an honest source violates the requirement r is $P(|d_i - g| > r) = 1 - \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-r}^{r} e^{-\frac{x^2}{2\sigma_i^2}} \mathrm{d}x$ or in the form of Gaussian error function $P(|d_i - g| > r) = \mathbf{erfc}(\frac{r}{\sqrt{2}\sigma_i})$. This probability is considered as the unreliability rate of the system and is considerably small, otherwise, there will be a non-negligible probability that the requirements are broken. For example, to achieve 5G URLLC's 10^{-7} error rate, r satisfies $r > 5.3\sigma_i$.

MS-PTP Threat Model. We consider insider attackers to exhibit arbitrary behaviors such as delaying and manipulating messages sent to the victim. They may also collude with each other. However, due to the protection from the secure communication channel and honest non-ad-hoc networks, since this scenario provides the best reflection of existing network architecture, one malicious node cannot alter the messages from other nodes in the network. As a consequence, any time synchronization session involving an insider attacker becomes a Byzantine session where a certain number of arbitrary timing measurements are delivered to the clients. The only limitation of the adversary's capability is that they are the minority of the total population and the number of them is smaller than a threshold f. We suppose the compromised measurements do not exceed one-third of the total population.

System Goal. The ultimate goal of our system is to ensure high-precision time synchronization across the PTP network. Given that more than 2/3-majority of measurements are honest ($n \geq 3f+1$) while others are Byzantium, PTP clients shall be guaranteed to have the final synchronization error smaller than a certain bound that is independent of the number of server population n and Byzantine sessions f.

Algorithm 1 MS-PTP

Input: The number of reachable servers n and the desired number of faults to counter f ($f \leq \lfloor \frac{n}{3} \rfloor$).

Output: System time updated with robust aggregated result o.

procedure Periodical Calibration
  function Get measurements
    for i `in` $\{1, 2, \cdots, n\}$ **do**
      Get v_i from `server` n_i.
    end for
    return $v_1, v_2, \cdots, v_n$
  end function
  function Measurements Aggregation
    for i `in` $\{1, 2, \cdots, n\}$ **do**
      $S(v_i) = \sum_{j \in NM(v_i)} \|v_i - v_j\|^2$
    end for
    $G = (f+1) \arg\min_{i \in \{1,2,3,\cdots,n\}} S(v_i)$
    $o = \frac{1}{f+1} \sum_{w \in G} w$
    return o
  end function
  function Timing Updation
    Update `system time` with o.
  end function
end procedure

2.5.2 MS-PTP-Byzantine Resilient Measurement Aggregation

Based on our assumption, a PTP client receives n measurements in one synchronization round T and f of them are compromised and follow arbitrary distributions. We denote these measurements as $(d_1, d_2, \cdots, d_{n-f}, b_1, b_2, \cdots, b_f)$, where $(d_1, d_2, \cdots, d_{n-f})$ refer to the honest measurements and $(b_1, b_2, \cdots, b_f)$ refer to the Byzantine measurements. The general representation of measurements (without knowing its honesty or not) are $v_i (i = 1, 2, \cdots, n)$. We first formally define a robustness criterion for the non-parametric accuracy requirement (in system goal) and introduce our aggregation algorithm $\mathcal{A}^*$ that satisfies this goal.

Definition 2.1 (Aggregation Robustness)**.** We define that an aggregation rule $\mathcal{A}$ is r-robust when its output result $o = \mathcal{A}(d_1, d_2, \cdots, d_{n-f}, b_1, b_2, \cdots, b_f)$ satisfies $\|\mathbb{E}[o] - g\| < s < r$,

where s is a determined bound independent of f and n. r is the system's synchronization requirement.

Byzantine-resilient Aggregation Algorithm $\mathcal{A}^*$. We define a score $S(v_i) = \sum_{j \in NM(v_i)} \|v_i - v_j\|^2$, where $NM(v_i)$ includes the $2f$ nearest measurements of v_i. Based on this definition, our aggregation algorithm $\mathcal{A}^*$ can be expressed as:

$$\begin{aligned} G &= (f+1) \underset{i \in \{1,2,3,\cdots,n\}}{\arg\min} S(v_i) \\ o &= \frac{1}{f+1} \sum_{w \in G} w \end{aligned} \tag{2.3}$$

where $(f+1) \arg\min_{i \in \{1,2,3,\cdots,n\}} S(v_i)$ refers to the set of v_i with the $f+1$ smallest scores; o denotes the output. Algorithm 1 demonstrates the workflow of MS-PTP. MS-PTP takes two parameters including the total number of servers n and the number of failures f as the system input. The clients do not know the exact number of f and usually need to select their desired number of failures to counter. To maximize the fault-tolerant capability, PTP clients usually select $f = \lfloor \frac{n}{3} \rfloor$.

2.5.3 Theoretical Proofs

Theorem 2.2 (Correctness)**.** *The Byzantine-resilient aggregation algorithm $\mathcal{A}^*$ is $\sqrt{2}\sigma_{max}$-robust, where $\sigma_{max} = \max\{\sigma_1, ..., \sigma_{n-f}\}$, in which $\sigma_i (i \in \{1, 2, \cdots, n-f\})$ is the standard deviation of honest measurement d_i.*

To prove this result, the following two lemmas are necessary. The proof of **lemma 2.3** is trivial and we present the proof of **lemma 2.4** in the appendices. We focus on the mathematical proof of our main result: theorem 2.2.

Lemma 2.3. *Define* $D(d_i) = \sum_{j \neq i} \|d_i - d_j\|^2$ *as the sum of the distances between* d_i *(for* $i \in (1, 2, \cdots, n-f)$*) and the other honest measurements. It is obvious to have* $S(d_i) \leq D(d_i)$ *for not all the honest measurements are always within* $NM(d_i)$*.*

Lemma 2.4. *There exists at least* $f+1$ *honest measurements, denoted as* $d_k \in \mathcal{B}$*, that satisfies* $\mathbb{E}[D(d_k)] \leq (2f+2)\sigma_{max}^2$*.*

[Proof of Theorem 2.2] For a measurement v_i, we define the set of honest measurements in its $2f$-nearest neighborhood as $H(v_i)$, with $\|H(v_i)\| = \delta_h(v_i)$ and the set of Byzantine measurements in its $2f$-nearest neighborhood as $B(v_i)$, with $\|B(v_i)\| = \delta_b(v_i)$, Obviously, $\delta_h(v_i) + \delta_b(v_i) = 2f$.

$$
\begin{aligned}
\|\mathbb{E}o - g\|^2 &= \|\mathbb{E}[\frac{1}{f+1}\sum_{i=1}^{f+1} w_i] - g\|^2 \\
&= \|\mathbb{E}[\frac{1}{f+1}\sum_{i=1}^{f+1} w_i] - \frac{1}{f+1}\sum_{i=1}^{f+1}\mathbb{E}\sum_{j\in H(w_i)}\frac{1}{\delta_h(w_i)}d_j\|^2 \\
&= \|\frac{1}{f+1}\sum_{i=1}^{f+1}\mathbb{E}[w_i - \sum_{j\in H(w_i)}\frac{1}{\delta_h(w_i)}d_j]\|^2 \\
\text{(AM-QM Ineq.)} &\leq \frac{1}{f+1}\sum_{i=1}^{f+1}\|\mathbb{E}w_i - \sum_{j\in H(w_i)}\frac{1}{\delta_h(w_i)}d_j\|^2 \qquad (2.4) \\
\text{(Jessen's Ineq.)} &\leq \frac{1}{f+1}\sum_{i=1}^{f+1}\mathbb{E}\|w_i - \sum_{j\in H(w_i)}\frac{1}{\delta_h(w_i)}d_j\|^2 \\
&= \frac{1}{f+1}\sum_{i=1}^{f+1}\mathbb{E}\|\frac{1}{\delta_h(w_i)}\sum_{j\in H(w_i)}(w_i - d_j)\|^2 \\
\text{(AM-QM Ineq.)} &\leq \frac{1}{f+1}\sum_{i=1}^{f+1}\frac{1}{\delta_h(w_i)}\mathbb{E}\sum_{j\in H(w_i)}\|(w_i - d_j)\|^2
\end{aligned}
$$

If w_i is an honest measurement, $\mathbb{E}\frac{1}{\delta_h(w_i)}\sum_{j\in H(w_i)}\|(w_i-d_j)\|^2 \leq \frac{\delta_h(w_i)2\sigma^2_{\max}}{\delta_h(w_i)} = 2\sigma^2_{\max}$. Else, if w_i is a Byzantine measurement, $\frac{1}{\delta_h(w_i)}\mathbb{E}\sum_{j\in H(w_i)}\|(w_i-d_j)\|^2 \leq \frac{1}{\delta_h(w_i)}\mathbb{E}[S(w_i)]$, which by Lemma 2.4 satisfies $\frac{1}{\delta_h(w_i)}\mathbb{E}\sum_{j\in H(w_i)}\|(w_i-d_j)\|^2 \leq \frac{2(f+1)\sigma^2_{\max}}{\delta_h(w_i)}$. For a Byzantine measurement, $\delta_h(w_i) \geq f+1$ and $\frac{1}{\delta_h(w_i)}\mathbb{E}\sum_{j\in H(w_i)}\|(w_i-d_j)\|^2 \leq 2\sigma^2_{max}$.

Combine all the results together, we have:

$$\begin{aligned}\|\mathbb{E}o-g\|^2 &\leq \frac{1}{f+1}\sum_{i=1}^{f+1}\frac{1}{\delta_h(w_i)}\mathbb{E}\sum_{j\in H(w_i)}\|(w_i-d_j)\|^2 \\ &\leq \frac{1}{f+1}(f+1)2\sigma^2_{max} = 2\sigma^2_{max}\end{aligned} \tag{2.5}$$

In summary, we can prove that $\|\mathbb{E}[o]-g\| \leq \sqrt{2}\sigma_{max}$, which is considered significantly smaller than the synchronization requirements r.

2.5.4 Analysis

Byzantine Resilience. We have provided a rigorous mathematical proof for the Byzantine resilience of our defense mechanism. The attackers, no matter what kind of behaviors they are doing, are not supposed to break the deterministic error bound. One potential concern the readers raise may be: Can the attackers shift the error bound of the aggregated outputs if they know the defense mechanism and send erroneous measurements accordingly? We investigate the following attack case to provide an intuitive answer.

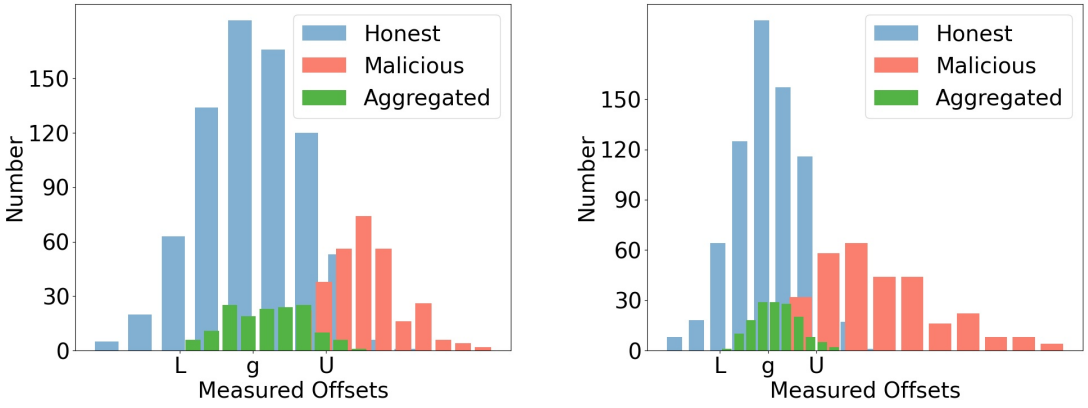
Case Study: Adaptive Attack. The attackers collude with each other and send malicious measurements near the proved error bound rather than typical random, constant, and cumulative values. The attacker's goal is to gradually break the error bound and shift the victim's time. The attack can be conducted in two steps:

- **Phase 1: Standard Deviation Estimation.** The attackers conclude with each other and thus know all the honest measurements. The attackers estimate the standard deviation of the honest measurements as $std_{adv} = \sqrt{\frac{\sum_{i=1}^{n-f}(d_i-\bar{d})^2}{n-f-1}}$.

- **Phase 2: Adding Malicious Measurements.** The attackers introduce malicious measurements trying to shift the error bound. The malicious measurements can be $\sqrt{2}std_{adv}+\epsilon$, where $\sqrt{2}std_{adv}$ refers to the attacker's estimation of the error bound and ϵ refers to the attacker's desired time shifting direction. Note that ϵ shall be gradually increased from a small value for it will be easily detected and discarded if they are too far away from the error bound.

We took a simulation to investigate and visualize the impact of the adaptive attack and our defense mechanism. The attackers launched the adaptive attack by having ϵ as a gradually increasing value (from 0 to $5\sigma_{max}$) that tries to shift the aggregated result slowly or just having ϵ as a constant small value ($\epsilon = 2 - \sqrt{2}\sigma_{max}$) that tries to shift the distribution of the aggregated results out of the upper or lower error bound. We took our simulation 150 times with the parameters from actual implementations of MS-PTP ($g = 13.95\mu s$ and $\sigma_i = 4.36\mu s$). Figure 2.6 is our simulation result, where a histogram map is plotted to show the distribution of honest, malicious, and aggregated measurements. We can observe that the attackers failed to do so and only a few points locate out of the error bounds. The simulation result is consistent with our theoretical proof, validating our mathematical results.

Scalability. MS-PTP ensures a deterministic error bound irrelevant to the number of participants. As a result, the theoretical error bound of MS-PTP does not change when the network size becomes larger. Table 2.1 demonstrates MS-PTP's performance under the adaptive attack in our simulations. We took the state-of-the-art Byzantine fault-tolerant gradient (data) aggregation mechanisms adopted from the federated learning frameworks as



(a) ϵ as a constant small value.

(b) Gradually increased ϵ.

Figure 2.6: Adaptive attack over MS-PTP. U refers to the upper bound and L refers to the lower bound.

a comparison. We took the adaptive attack because it adds malicious measurements near the theoretical bound and can be used to explore the error bound in practice. When there is no attack, the system measures the offset as $g = 13.95\mu s$ and $\sigma_i = 4.36\mu s$. We increased the size of the network from n=4 to n=28 and checked MS-PTP's performance under the attack. We can observe that MS-PTP does not achieve the best performance. But when the network size becomes larger, MS-PTP achieves the best performance among all mechanisms. The output result of MS-PTP remains to be stable and does not increase significantly with a larger network size.

Complexity. MS-PTP requires multiple (n) redundant servers in the network. As a result, the communication overhead is increased by $n\times$ on the client side. On the server side, MS-PTP does not impose a lot of extra communication overhead for each server that operates in its own domain as a usual one. MS-PTP is a lightweight protocol and does not introduce a lot of computation overhead. The algorithm can be executed within several milliseconds. This is crucial because the algorithm is conducted periodically and a large execution time

will pose a significant overhead to the system.

Backward Compatibility. MS-PTP is fully compatible with the current PTP standard. In the cases there are not enough PTP servers such as in the CPS systems, MS-PTP may also resort to alternative external redundancy such as GPS sources and NTP sources. There have already been some cross-protocol synchronization management tools such as Chronyd [151] to help fetch from these available sources.

2.6 Evaluation

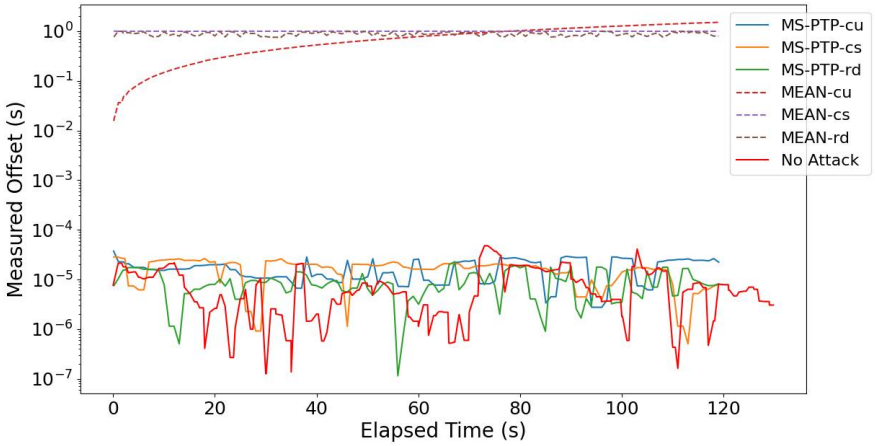
Table 2.1: MS-PTP scalability performance (measured offset in microseconds) under attack.

Scheme	Fault Tolerance	Proved Bound	$n = 4$	$n = 10$	$n = 16$	$n = 22$	$n = 28$
Krum [25, 56]	$n \geq 2f + 3$	$\sim \sqrt{2n - 2f}\sigma_{max}$	15.407	18.11	20.537	22.478	24.002
Bulyan [56, 70]	$n \geq 4f + 3$	Same as Krum	-	-	-	-	-
Median [56, 204]	$n \geq 2f + 1$	$\sqrt{n - f}\sigma_{max}$	15.880	18.231	19.403	20.167	20.706
Trim Mean [204]	$n \geq 2f + 1$	$\sim \sqrt{\frac{2n}{n-f}}\sigma_{max}$	16.205	18.454	19.457	20.153	20.607
Phocas [56, 198]	$n \geq 2f + 1$	$\sim \sqrt{4 + \frac{12n}{n-f}}\sigma_{max}$	18.667	27.413	36.142	44.827	54.447
MS-PTP	$n \geq 3f + 1$	$\sqrt{2}\sigma_{max}$	**15.606**	**16.746**	**17.233**	**17.474**	**17.552**

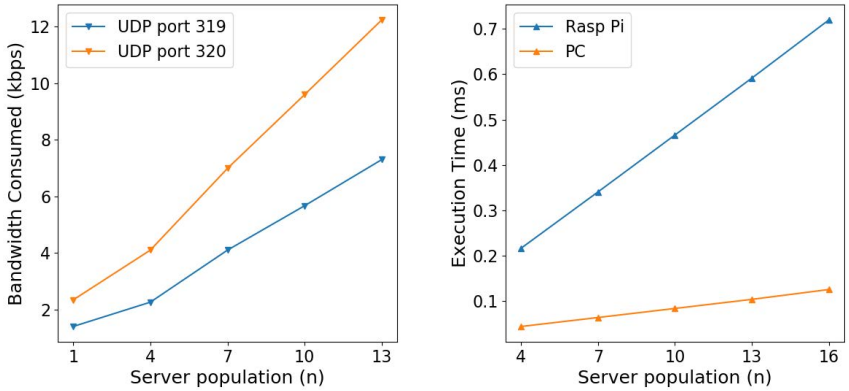
2.6.1 Byzantine Resilience

We implemented MS-PTP on our PTP testbed to evaluate its performance on a real IoT network, where the experimental settings are the same as the attack experiments in Section 2.4. We added an additional Raspberry Pi to the network and configured all four nodes as server nodes to meet the network redundancy required by MS-PTP. We selected a desktop as a PTP client and configured it to receive measurements from different devices simultaneously. The PTP client captured and took the IP address (in LAN) of all four servers

and configured its profile to establish four independent sessions with each server. We built up MS-PTP on top of PTP engines with Python code by taking measurements from PTP sessions and aggregating these measurements to obtain a robust timing estimation. This testbed supports hardware validation when the malicious population $f = 1$ and the number of servers $n = 3f + 1 = 4$. We conducted time shifting attack on the compromised node and observed the synchronization performance of the PTP client who employed MS-PTP as a countermeasure.



(a) MS-PTP accuracy performance under attack (*cu* – cumulative delay, *cs* – constant delay, *rd* – random delay)



(b) MS-PTP communication overhead.

(c) MS-PTP computation overhead.

Figure 2.7: MS-PTP performance on real IoT testbed.

Fig. 2.7 shows MS-PTP's performance against time shifting attack, in comparison with the MEAN aggregation method that simply takes the average of measurements as a benchmark. In the experiment, the malicious delay d was added in constant (2s), random (mean 1.5s, standard deviation (std) 500ms), and cumulative (1.8s per 120s) ways inconsistent with the attack settings in section 2.4. Note that the y-axis is plotted in a logarithm scale to show the results' accuracy level. We can observe that without MS-PTP, the MEAN aggregation's clock offset is shifted by several seconds (10^0). With MS-PTP, the system's measured offsets are maintained at a level of 10 μs (10^{-5}), nearly the same as the results without attack.

2.6.2 Communication and Computation Overhead

We configured each device to emulate multiple servers in order to testify to the overhead introduced by MS-PTP when the network size becomes larger. For each Raspberry Pi, we launched four PTP sessions and pinned each of them on an independent CPU core respectively. Since Raspberry Pi 4 used in our experiments has a quad-core CPU, we were able to emulate up to 16 PTP sessions/servers in this way. Fig. 2.7 shows the communication and computation overhead introduced by MS-PTP. For the communication overhead, we monitored the bandwidth consumed by the UDP ports 319 and 320 used by PTP during our experiment on the testbed. We observed tens of kilo bytes-level bandwidth consumption in our experiments. The bandwidth consumption is linearly increasing with respect to n and we take it as a small overhead when n is not large. For the computation overhead, we checked MS-PTP's execution time on one PC and Raspberry Pi. We can also observe a linear increase in the execution time with respect to the server population. MS-PTP's execution time is very small (< 1 ms) on both Raspberry Pi and PC.

Table 2.2: Measured offset with NTP and GPS servers

Scenarios	Accuracy mean	Accuracy std
Measurements without attack	$13.95\mu s$	$4.36\mu s$
No redundant servers	$37.86ms$	$2.31ms$
3 PTP servers	$12.80\mu s$	$1.32\mu s$
3 NTP servers	$0.974ms$	$0.551ms$
2 NTP, 1 GPS servers	$0.576ms$	$0.578ms$

2.6.3 Compatibility

Another important advantage of MS-PTP is that it can not only be used by PTP, but also by the other popular time synchronization mechanisms. We validated MS-PTP's compatibility with NTP and GPS synchronization methods by introducing NTP servers and GPS servers as redundant time sources. We considered the existence of one Byzantine insider, but with either 3 NTP servers or 2 NTP servers plus 1 GPS server serving as the honest redundancy. In these cases, MS-PTP is not only built upon PTP engines but also upon NTP and GPS engines. Fortunately, all of these synchronization mechanisms provide users with convenient interfaces. Any user with *sudo* priority can get access to them and implement MS-PTP on top of them. Table 2.2 shows the accuracy performance on our real testbed with these settings when a constant 100 ms delay d is added. We can observe that redundant NTP servers do provide a fault-tolerant guarantee for the synchronization service. The synchronization accuracy level was reduced from $10ms$ level after the attack, to a normal NTP server's $< 1ms$ level. However, because NTP servers' accuracy was worse than PTP servers, the aggregation performance was reduced to NTP accuracy. When a GPS server was added, the aggregation accuracy became better. This indicates that the more accurate servers we are using, the better accuracy MS-PTP can achieve.

2.7 Discussion

MS-PTP uses network redundancy to counter Byzantine failures. However, some Man-in-the-middle (MitM) attackers are too strong and can not be addressed by MS-PTP. For example, suppose the attacker controls a choke point, where all the communication sessions between the servers and clients pass through. In that case, he can add malicious delays, and no matter how many redundant servers MS-PTP uses, still can not find him. Therefore, we recommend that network administrators shall deploy servers in separate locations with as few joint communication paths as possible to avoid such MitM attackers. An individual client should also try to select servers in different geological areas and node-disjoint communication paths from different sources.

2.8 Conclusion

In this chapter, we focus on addressing the Byzantine insider attacks in time synchronization systems. We first demonstrated through hardware experiments that the current PTP implementations are susceptible to a practical insider time-shifting attack, in which only one malicious insider is able to bring catastrophe to the time synchronization service. As a countermeasure, we devise a novel Byzantine-resilient aggregation scheme to generate a high-fidelity, error-bounded measurement under the assumption that fewer than one-third of the measurements are Byzantine-influenced. We provide rigorous proof and thorough analysis for the theoretical guarantees that MS-PTP ensures. To evaluate the feasibility and performance of our new mechanism, we implemented a proof-of-concept MS-PTP with a combination of hardware experiments and software implementations in various Byzantine-ridden network scenarios. The result shows that MS-PTP achieves excellent synchronization

accuracy for client clocks under different practical attacks. Timing is an important attack vector against the CPS/IoT systems, which have and will be gradually exploited in many time and safety-critical systems. We hope this chapter provides a good solution for these time-sensitive systems to set up robust time synchronization services and can motivate more research about the timing vulnerabilities of CPS/IoT systems.

Chapter 3

Building up Trustworthy and Verifiable Inter-SAS Coordination

(Copyright Notice[1])

3.1 Introduction

The radio spectrum is an important and scarce resource for wireless communications. To accommodate the ever-increasing volume and variety of spectrum users in the commercial setting, spectrum regulatory agencies in the US, notably the Federal Communications Commission (FCC) and National Telecommunications and Information Administration (NTIA) have opened up previously exclusive spectrum bands for more flexible shared use by the public [98], culminating in a paradigm of dynamic spectrum sharing (DSS). The Spectrum Access System (SAS), mandated by the FCC for managing spectrum sharing in the 3.55- 3.7 GHz Citizens Broadband Radio Service (CBRS) band [134], is currently the most promising DSS-enabling technology and has started commercial deployment in 2020. At its core, SASs

[1]This chapter previously appeared as a part of a conference paper published in ACM ASIACCS 2024. ©2024 Copyright held by the owner/author(s). Reprinted, with permission, from Shanghao Shi, Yang Xiao, Changlai Du, Yi Shi, Chonggang Wang, Robert Gazda, Y. Thomas Hou, Eric Burger, Luiz DaSilva, and Wenjing Lou, "TriSAS: Toward Dependable Inter-SAS Coordination with Auditability," in Proceedings of the 19th ACM Asia Conference on Computer and Communications Security (ASIACCS'24), pages 399-411, 2024 [165].

are implemented and operated by certified service providers, called *SAS administrators*, to allocate spectrum usage resources (location, time, bands, power) to commercial spectrum users per request while ensuring the pre-emptive access right of incumbent users (e.g., military radars, satellite stations, and other federal users). Multiple SAS administrators have been approved by the FCC for the CBRS ecosystem, including Google, Federated Wireless, Sony, Amdocs, and CommScope.

3.1.1 Inter-SAS Coordination Problem

The SAS administrators have been certified to manage the CBRS band usage of their customers (spectrum users) through proprietary SAS servers. They overlap in geographical service areas, a situation similar to that of current mobile network operators (MNOs). However, unlike the current MNOs where each MNO operates in its own licensed spectrum bands, all SAS administrators serve their customers in the *same* CBRS band. This brings a fundamental inter-SAS coordination problem—the SAS administrators should be coordinated so that their customers do not receive the same or significantly overlapping allocations (location, time, bands, power). Failing to address this problem leads to service de-synchronization and allocation conflicts, which could result in harmful interference between spectrum users. Unfortunately, this is strictly prohibited by the current WInnForum Standards [191] because any interference that exceeds the pre-defined threshold can disrupt the spectrum sharing service of the whole area.

However, during the operation of the SAS, some SAS servers may inevitably deviate from normal routines because of server crashes, regular system upgrades, or even malicious hacking. As a result, these SAS servers exhibit arbitrary behaviors and can be treated as Byzantine nodes. For example, SAS servers may become offline and stop service for a certain period

because of regular maintenance. Malicious SAS servers may purposely modify or lie for some shared information to gain excessive spectrum resources in the shared CBRS band. To provide a sound and sustainable spectrum sharing service, the SAS system shall be able to overcome these issues and ensure that the system operates smoothly even if a portion of the server fails. More specifically, the already committed spectrum allocation transactions shall not be revoked or modified even if their proposed servers crash down, and the malicious servers shall not jeopardize the coordination procedure of the honest servers. Moreover, when these crashed or malicious servers go online, they shall be able to recover their operation status and retrieve the records from the honest servers.

The Coordinated Periodic Activities for SASs (CPAS) procedure, as specified in the Wireless Innovation Forum (WInnForum) standard [60], is the only incumbent inter-SAS coordination mechanism used in practice. This mechanism requires SAS administrators to perform a full-dump-style synchronization of SAS service states on a daily basis, from 3 AM to 6 AM U.S. Eastern Time, so that they can provide non-conflicting spectrum allocations to users the next day. However, this overnight message-exchange protocol neither considers the possible presence of malicious SASs nor meets with real-time performance expectations of the dynamic spectrum sharing service.

To tackle these issues, recent literature has proposed using the blockchain and distributed ledger technology to construct fault-tolerant SASs and inter-SAS coordination schemes [18, 69, 196, 210]. They either use the blockchain to bootstrap a distributed spectrum database [18, 69], or leverage smart contracts' flexible programming interfaces to implement spectrum allocation and other business logic [196, 197, 210]. The blockchain-based solutions can also provide auditability by maintaining an immutable ledger of spectrum operations across all participants. In [196, 197] particularly, the fault-tolerant inter-SAS coordination is fulfilled by the blockchain system's built-in Byzantine fault-tolerant (BFT) consensus, which essen-

tially realizes a state machine replication (SMR) in that all SAS servers follow the same spectrum allocation algorithm and curate a unified ledger for all spectrum users. In this way, the non-faulty servers can synchronize and process new requests sequentially, instead of exchanging all service data in a full-dump style (as in the case of CPAS).

Limitations of Blockchain/SMR-based SAS: While providing consensus-driven security and immutable records of spectrum access, the above blockchain-based SASs and SMR-based inter-SAS coordination schemes do not represent a practical solution in the recently deployed SAS ecosystem. First and foremost, the SAS administrators are competitive commercial entities and manage their own service subscribers and adopt proprietary allocation algorithms which could be commercial secrets. As a result, there is no unified allocation algorithm among different SAS administrators, and their managed SAS servers cannot be treated as replicated state machines. Moreover, implementing complex allocation algorithms in blockchain smart contracts and executing them could face prohibitive on-chain costs, since the spectrum allocation algorithm can be highly complex, such as using graph coloring-based [206], reinforcement learning-based [7] or optimization-based approach [85] to derive a fair and effective spectrum allocation. Therefore, instead of replicating the same spectrum sharing service across different entities, it is imperative for the SASs to adopt a coordination mechanism that can ensure a unified and fair spectrum allocation among all participating SASs without having all parties pre-agree on any allocation algorithm. To avoid situations in that malicious or selfish SAS plays favoritism towards its own customers, security properties including fault-tolerance, immutability, and auditability of spectrum allocations should also be observed.

3.1.2 Our Solution

In this work, we propose TriSAS: a trustworthy, robust, and efficient inter-SAS coordination mechanism that achieves the above goals while being backward compatible with the existing SAS framework. TriSAS's core mission is to facilitate independent SAS servers to generate fair and auditable spectrum allocations for their users in an efficient manner while ensuring correctness against selfish and even malicious SAS servers. Catering to the heterogeneity of spectrum allocation algorithms among different SAS servers, we decompose the inter-SAS coordination into a two-phase process—the input data synchronization phase and the decision finalization phase. We use a blockchain database as a key component for the secure and efficient handling of spectrum requests from all users.

For the input synchronization phase, we require all servers to store their input requests in the blockchain database proactively. The blockchain database automatically converts the requests to input transactions and uses a reliable broadcast mechanism to deliver the transactions to all other SAS servers securely. This phase ensures that all participants share a common input set. In the decision finalization phase, each SAS server first pulls the common input set from the blockchain database and computes its allocation proposal locally via its proprietary allocation algorithm, which is stored in the blockchain database and reliably broadcasted to other parties. To avoid any SAS from playing favoritism towards its own users, we design a voting sub-phase to allow all SAS servers to agree on the best allocation proposal. We propose an allocation evaluation algorithm (AEA), which balance between the spectrum utilization rate and allocation fairness, for the SASs to evaluate different allocations following pre-agreed evaluation criteria. An allocation receiving a high score needs to boost spectrum usage efficiency and ensure allocation fairness at the same time. The servers vote for their preferred allocation according to the scores obtained from AEA. The allocation that is voted by the majority of the servers is elected as the final allocation result. TriSAS

stores all the transactions in an immutable ledger for potential audits. In general, TriSAS can tolerate one-third of SAS servers being Byzantine.

We implemented TriSAS on the AWS cloud computing platform with the Bigchaindb [120] serving as the blockchain database. We deployed a network of servers in different areas across the U.S. to simulate real-world deployments of the SAS. We evaluated the performance of TriSAS with respect to different input rates and pulling intervals. We focused on the throughput and latency performance of TriSAS and the experiment results show that TriSAS can achieve scalable performance under practical settings.

In summary, this chapter makes the following contributions:

- We identify the inter-SAS coordination problem and its unique challenges of achieving operation security while accommodating the diversity of allocation algorithms across distributed SAS servers in the current CBRS ecosystem.
- We propose TriSAS, a new inter-SAS coordination mechanism to address these challenges. TriSAS is resilient to Byzantine SAS servers when their number does not exceed one-third of the total population and at the same time keeps high throughput and low latency. TriSAS extends the current SAS service model and essentially ensures the safety, fairness, auditability, and efficiency of spectrum allocations and SAS operation.
- We propose a novel allocation evaluation algorithm (AEA) to evaluate the fairness and spectrum utilization efficiency of spectrum allocations from different SAS servers and help the system decide on a final selection. This algorithm can also serve as a benchmarking tool for general SAS performance evaluation and may be of independent interest.
- We implemented a prototype of TriSAS. Extensive experiment results show that TriSAS

Table 3.1: A comparison between different blockchain-based SAS.

Scheme	Blockchain's role	Who curate blockchain	Fault-tolerant allocation	CBRS-SAS compatibility
CPAS [60]	N/A	N/A	No	Yes
[18]	Rule	Third Party	No	No
[210]	Allocation	Prior Users	Yes	No
TrustSAS [69]	Record Keeping	GAA Users	No	Yes
BDSAS [196, 197]	Allocation	SAS Servers	Yes	Yes
TriSAS	Record Keeping	SAS Servers	Yes	Yes

Scheme	Inter-SAS coordination	Allocation generation	Algorithm diversity
CPAS [60]	Yes	Off-chain	Yes
[18]	No	On-chain	No
[210]	No	On-chain	No
TrustSAS [69]	Yes	Off-chain	Yes
BDSAS [196, 197]	Yes	On-chain	No
TriSAS	Yes	Off-chain	Yes

is a practical inter-SAS coordination solution in handling large input volumes with a small processing latency.

3.2 Research Background

3.2.1 Spectrum Access System

The FCC has designated the Citizens Broadband Radio Service (CBRS) band [134], a 150 MHz frequency band between 3.55 GHz to 3.7 GHz, to accommodate the shared spectrum usage between federal and non-federal users [134]. The users get access to this spectrum resource according to a three-tiered framework approved by the FCC, with the incumbent federal users being positioned at the highest tier, the non-federal Priority Access License (PAL) holders at the middle tier and the non-federal General Authorized Access (GAA)

users at the lowest tier. At the heart of this new shared spectrum usage paradigm is an automatic spectrum coordinator, named the Spectrum Access System (SAS), which dynamically allocates spectrum to users at various tiers and controls the operation and management of the spectrum usage. The SAS ensures that lower tier users cannot cause harmful interference to higher tier users, and at the same time boosts the spectrum utilization efficiency. SASs are implemented by different SAS administrators designated by the FCC and deployed on their SAS servers. Currently, different SAS administrators manage their own subscribers and allocation algorithms. They take spectrum access requests from their spectrum users and respond with transmission grants indicating whether the requests have been approved as the result of their spectrum allocation algorithms.

3.2.2 Blockchain-based SAS

The unique properties of blockchain, i.e., decentralization, transparency, and consensus-based security, make it a potential enabler to future spectrum management [189]. Recent works have proposed several blockchain-based, decentralized SASs to realize secure and verifiable dynamic spectrum access.

Ariyarathna et al. [18] propose a smart contract-based SAS that mainly focuses on creating and trading "spectrum tokens" based on Ethereum. This work adopts the SAS admins as a centralized party and does not consider the problems (coordination, fault tolerance, etc.) introduced by the current decentralized SAS settings. Zhang et al. [210] propose an enhanced smart contract-based dynamic spectrum sharing system, as well as a novel consensus mechanism. This work also incorporates the privacy-preserving consideration of users into their design. However, although this explores the use of a smart contract to implement the allocation, which results in a fault-tolerant allocation among participants, it

still adopts the single SAS administrator model and fails to address the challenges introduced by decentralized SAS settings. Grissa et al. [69] introduce TrustSAS, a secure and privacy-preserving SAS that combines state-of-the-art cryptography with the blockchain technique. However, for inter-SAS coordination, TrustSAS only uses the blockchain (the global chain in their design) as a record-keeping board to accept any allocation proposals proposed by a cluster of users, who are usually governed by a single SAS administrator. In a situation where different clusters of users (SAS administrator) have overlapped serving areas, which unfortunately is the current fact, cannot prevent a selfish cluster from proposing unfair or even fake allocations in favor of himself. Xiao et al. [197] propose BD-SAS, a fault-tolerant, decentralized SAS that uses a two-layer blockchain system where the global chain is used for spectrum regulation compliance and smart contract-based local chains are used in individual spectrum zones for automating spectrum allocation. However, BD-SAS still relies on smart contracts to perform on-chain spectrum allocation, which is neither efficient nor enables allocation algorithm diversity.

3.2.3 Blockchain Database

The blockchain database is an emerging technology that combines blockchain's decentralization, Byzantine fault tolerance, and data immutability properties, with distributed databases' query, high throughput, and low latency properties. Blockchain databases are mostly deployed in small-scale private networks consisting of multiple or tens of peers who know each other's identities. These networks enforce strict access control policies and only authorized entities can participate in the network. Examples include Bigchaindb [120] and Couchdb [13]. Compared to classical cryptocurrency and smart contract systems, blockchain databases are not purposed for realizing a full-fledged payment system. Instead, they are customized to handle a larger volume of data transmission and storage across different nodes, while at

the same time maintaining the fault-tolerance property for common database operations. Whenever a data processing operation (e.g., insert, modify, and query) is received by a peer, it leverages the underlying reliable broadcast mechanism to forward it to all peers to ensure that every node maintains the same immutable ledger.

3.3 System Model

3.3.1 System Architecture

Geographical Concepts: Following the definition of the WInnForum Standard [191] and previous works [196, 197], we consider spectrum sharing zone as a unit spectrum management geo-location areas in which SAS administrators can realize all spectrum sharing functions including spectrum request, response, allocation, and inter-SAS coordination. In practice, a zone usually refers to a US county. Different spectrum sharing zones operate independently and spectrum allocations in one zone will not affect the allocations of others. But all spectrum zones are subject to global regulations where the global refers to the entire spectrum jurisdiction such as the US continent. In this chapter, we focus on the spectrum sharing system within one zone as spectrum sharing is conducted on a per-zone basis. We define five types of participants in the spectrum sharing system, as shown in Figure 3.1.

Spectrum Users operate one or more CBSDs (base stations) in the system. They register themselves to their SAS administrators when they first join the system and later send spectrum usage requests to get access to spectrum resources. They are expected to receive transmission grants, i.e. the decision about whether their requests are granted or not. Currently, one spectrum user only subscribes to and receives service from one SAS administrator. Within one zone, there are hundreds of CBSDs, denoted by $CBSD_k, (k = 1, 2, \cdots, m)$.

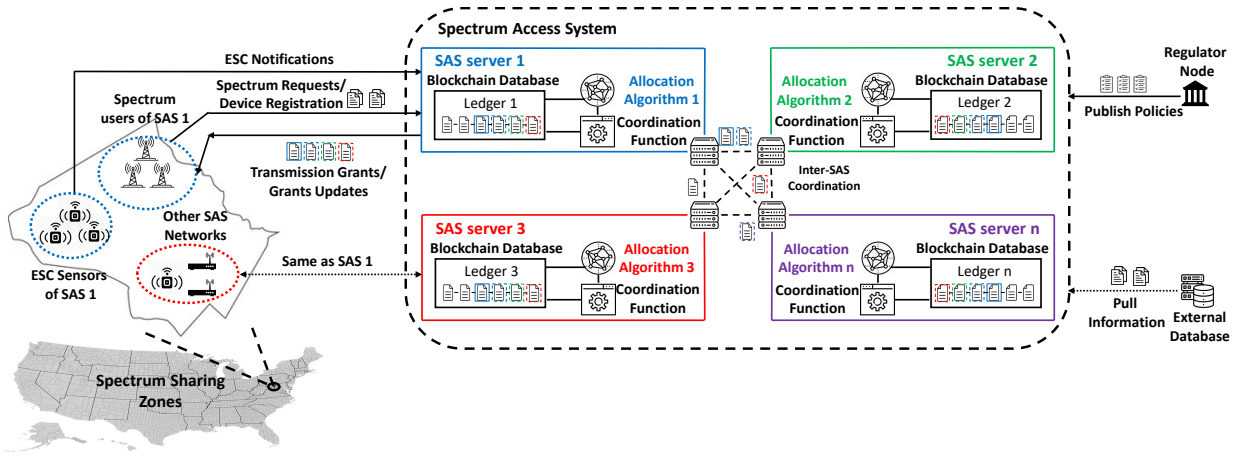


Figure 3.1: TriSAS system model.

ESC Sensors is a set of radio sensors deployed by either the SAS administrators or other appointed entities that aim to detect the presence of incumbent spectrum users such as naval radars. After they detect the presence of incumbent users, they send ESC notifications to SAS servers and the servers suspend the transmission grants of CBSDs that might cause interference to the incumbents. ESC notifications usually have strict latency requirements and are processed by each SAS server locally.

SAS Servers are deployed by the SAS administrators to coordinate the spectrum usage of spectrum users. We assume there is only one server in charge within one spectrum sharing zone for each SAS administrator, denoted by $SAS_i, (i = 1, 2, \cdots, n)$, where n refers to the number of SAS administrators in the zone. In one zone, there are typically about 4-7 SAS servers/administrators and each SAS server has its own list of subscribers. The SAS servers are the core components of the spectrum access systems. They contain several key functions including database functions, coordination functions, and proprietary allocation algorithms. Within one server, the database function stores all the essential records including system inputs, intermediate results, and final results. For a blockchain database, the database function stores all the transactions in an immutable ledger. The allocation algorithm, which

is considered to be owned by a SAS administrator who is unwilling to expose its details because of commercial and intellectual property reasons, is in charge of generating spectrum allocations by taking input sets from the database function. The coordination function is in charge of keeping state synchronization across SAS servers. It interacts with the other two functions within its server and the other coordination functions within other servers. It is the core component of our inter-SAS coordination mechanism.

Regulatory Entities include the NTIA and FCC. They are the trusted entities in the system and regulate the operation of SAS administrators. They do not participate in the spectrum sharing service operations and only publish and enforce regulation rules in the system. The frequency of policy upgrades is relatively low, in the order of days and months.

External Databases provide supplemental information to SAS servers, such as geographic information. The data size of the pulled records from them is usually very large and the frequency of updates is very low. For some of them, the SAS server may only pull the records once to its cloud for the whole life cycle unless there are significant changes. Therefore, we do not consider these records necessary for our real-time coordination procedure.

3.3.2 Threat Model

The WInnForum standard stipulates the 5G spectrum sharing system to establish a CBRS public key infrastructure (PKI) [192]. As a result, each node in the network, including a spectrum user and a SAS server, can sign its messages with its own private key. The other nodes can verify the signatures with the corresponding public key retrieved from the CBRS PKI. This prevents a network-level attacker from eavesdropping, modifying, and forging messages (or transactions) in the system. Unfortunately, this built-in security mechanism cannot distinguish the messages sent by malicious SAS servers, who are considered insider

attackers and can sign the malicious messages with their own secret keys to bypass the cryptographic checks.

Because of crashes down, regular maintenance, and even malicious hacking, we consider a portion of the SAS server may deviate from the normal operation routine for a certain period of time. But we assume they can eventually be fixed. Furthermore, some selfish SAS administrators may purposely sabotage the system by having its SAS servers send conflicting information to other SAS servers under the current framework for their own benefit. For example, the attacker can lie about the status of specific spectrum transmission grants to cause a de-synchronization of the service state between different SAS servers and further trigger allocation conflicts, ultimately jeopardizing shared spectrum use. The attacker when acting as a selfish administrator may configure its SAS server to propose unfair allocations via its proprietary algorithm to give its own clients extra spectrum resources when it is in turn.

In this work, we use "Byzantine behavior" to cover all the possible SAS server behaviors that deviate from the normal routine (i.e., arbitrary behaviors that do not follow the correct coordination procedure). We assume the Byzantine servers are fewer than one-third of the total population, which follows the convention in classical Byzantine fault tolerance problems in distributed systems [30, 35, 104]. Moreover, in current SAS ecosystems, the SAS administrators and their managed SAS servers are certified entities. They make a profit in the spectrum sharing market and are incentivized to follow the FCC rules for the operation of SAS administrators, i.e., 47 CFR Part 96 [134]; failure to comply can have serious legal ramifications. The certified and commercial nature of SAS administrators makes the 2/3 honest majority assumption achievable in practice.

Spectrum users may also behave in certain malicious ways. They can launch DoS attacks or send falsified spectrum requests to acquire excessive allocations. This requires the SAS

servers to enforce strict access control. For this work, we regard this as the responsibility of individual SAS administrators and out of the scope of the inter-SAS coordination problem.

3.3.3 System Goal

The inter-SAS coordination mechanism takes input sets $\mathcal{R}_i(t)(i = 1, 2, \cdots, n)$ from all SAS servers that contain all active requests and can ensure the following properties:

- **State Synchronization:** The inter-SAS coordination mechanism is conducted periodically and after each round the states $\mathcal{S}_i(t)$ of SAS servers shall be synchronized.
- **Byzantine Resilience (Operation Security):** The coordination mechanism needs to counter Byzantine servers. It shall ensure that the honest servers function normally and will not be affected by Byzantine servers when they constitute less than 1/3 of the total population. Once the crashed or malicious servers are recovered, they can synchronize themselves with the honest ones easily.
- **Uniform Allocation and Record Audibility:** The distributed SAS servers should agree on a uniform allocation for each user's spectrum request. The input sets and spectrum allocations (along with how the agreement is reached) need to be stored in a public immutable ledger. Every node can get access to this ledger for audibility purposes. As a result, any malicious behaviors can be identified and tracked in the immutable ledger and the corresponding malicious party cannot deny doing so because all records are cryptographically signed and verifiable.
- **Fairness and Efficiency:** The coordination mechanism needs to be performed in real-time without adding too much overhead and latency. The processing latency shall be smaller than the periodical communication interval between CBSDs and servers (300

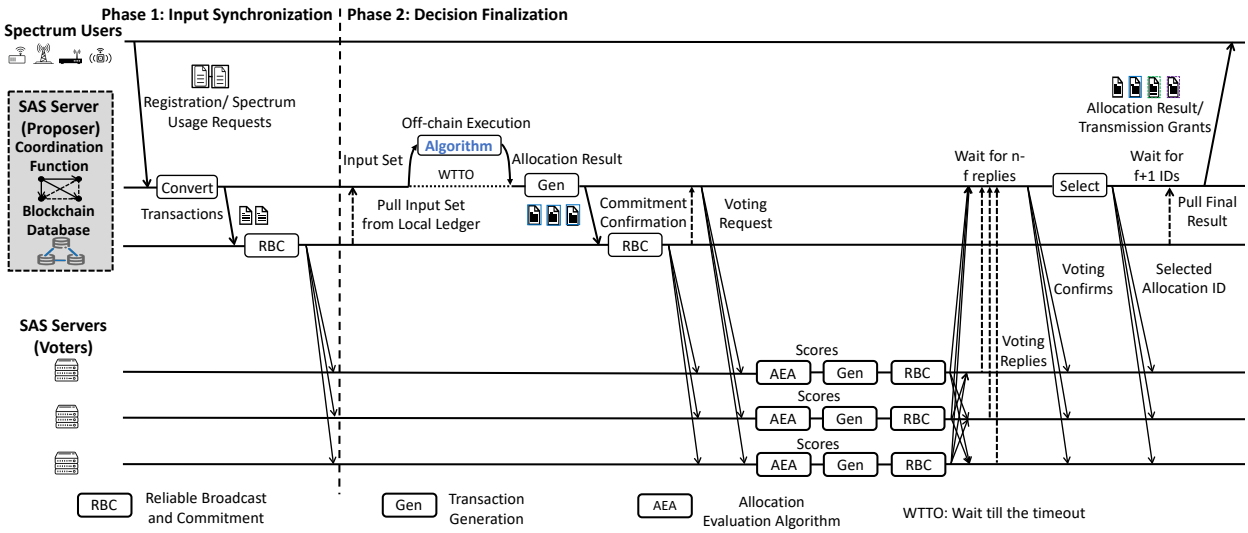


Figure 3.2: TriSAS workflow.

seconds) under practical input rates (about 30 requests per minute). The coordination mechanism also needs to ensure that the final allocations are voted by a 2/3 majority of servers to avoid potential unfair allocations. To avoid imposing a large on-chain execution overhead, the proprietary allocation algorithms shall be executed locally.

3.4 Approach

3.4.1 Mechanism Overview

TriSAS is a two-phase mechanism including the input synchronization and decision finalization phases, as shown in Figure 3.2. For each SAS server, the input synchronization phase is triggered whenever there are new device registration requests or spectrum usage requests. The input synchronization phase needs to ensure that all SAS servers share a common input set $\mathcal{R}(t)$, which is the union of all the SAS servers' individual input sets,

i.e. $\mathcal{R}(t) = \bigcup_{i=1}^{n} \mathcal{R}_i(t)(i = 1, 2, \cdots, n)$. This set is stored in an immutable ledger in the blockchain database and can be used in the decision-finalization phase to generate spectrum allocations.

The decision finalization phase is in charge of generating spectrum allocations and finalizing one of them as the valid allocation that is favored by a 2/3 majority of SAS servers. This phase requires the coordination function within the servers to periodically (with interval T) check whether there are new transactions in the common input set $\mathcal{R}(t)$. The decision finalization phase is triggered when these proposers find new input transactions in the common input set. Within one interval, there can be one or more proposers, depending on how many SAS administrators are proposing spectrum allocations. The decision finalization phase needs to be finished within the interval T and can accomplish periodical synchronizations of SAS server states $\mathcal{S}_i(t)$.

SAS server's state $\mathcal{S}_i(t)$ contains three types of records, the common input set $\mathcal{R}(t)$, the proposed allocation set $\mathcal{A}(t)$, and the vote set $\mathcal{V}(t)$. These are stored in the immutable ledger of the blockchain database in the form of transactions and data blocks. All participants can get access to the ledger for audit purposes. Each data block contains a hash value of previous blocks and any modification of already committed blocks can be detected.

3.4.2 Input Synchronization

The SAS servers take the requests from the spectrum users as the inputs to the system. Different SAS administrators have their own input set $\mathcal{R}_i(t)$. The input set consists of requests signed by CBSDs, denoted by $r_{CBSD_k}^{t_{gen}}$, where t_{gen} refers to request generation time. The number of requests increases monotonically with respect to time t, i.e. $|Input_i(t)| \propto t$.

For each server, when the coordination function receives a request from spectrum users, it

first transfers the request to the corresponding input transaction by formatting the request into the blockchain database's standard and signing the transaction with its own private key. As a result, the device registration requests are transferred to registration transactions that contain the device identity, antenna characteristics, power level, and location information. The transactions are signed by their origin CBSDs and the enrolled SAS server, denoted by $Reg_{tx} =< id_{device}, antenna, power, loc, sig_{CBSD_k}, sig_{SAS_i} >$.

Similarly, the spectrum usage requests are transferred to request transactions that contain device identity, request spectrum ranges, request time and duration, and are also signed by their origin CBSDs and the enrolled SAS server. We denote them as $Req_{tx} =< id_{device}, range, t_{req}, duration, sig_{CBSD_k}, sig_{SAS_i} >$.

After the transactions are generated, the coordination function sends the transactions to the blockchain database. The coordination function can either send the transactions immediately whenever new ones arrive or can buffer the transactions into batches before sending them. The blockchain database, upon receiving the transactions, organizes the messages into data blocks. The underlying consensus mechanisms of the blockchain database, such as PBFT [35] or Tendermint [30], ensure that these blocks are reliably broadcasted and stored by all honest SAS servers, i.e. the blocks are committed. The coordination function receives a commit confirmation message when all data blocks are committed, indicating that the input synchronization phase is finalized and the requests have been stored in an immutable ledger. Notably, the SAS servers cannot modify or forge the registrations and spectrum usage transactions because they cannot forge the signatures of the CBSDs.

Algorithm 2 TriSAS-Phase 1: Input Synchronization

Requirement: The number of servers is n, and the desired number of faults to counter f satisfies $f \leq \lfloor \frac{n}{3} \rfloor$.

Ensure: All SAS administrators have the same input set.

1: **procedure** INPUT SYNCHRONIZATION
2: **function** REQUEST CONVERSION AND BROADCAST
3: **for** i `in` $\{1, 2, \cdots, n\}$, each `server` SAS_i **do**
4: Get $Req_{tx}/Reg_{tx} = Convert(r^{t_{gen}}_{CBSD_k})$.
5: Broadcast Req_{tx} or Reg_{tx} to others.
6: Store received Req_{tx} or Reg_{tx} to the ledger.
7: **end for**
8: **end function**
9: **end procedure**

3.4.3 Decision Finalization

The decision finalization phase is launched with interval T by SAS servers (proposers) denoted by SAS_j where $j \in \{1, 2, \cdots, p\}$ and p is the number of proposers. This phase is asynchronous with the input synchronization phase. This phase has three sub-phases: allocation generation, allocation voting, and allocation decision.

Allocation Generation

The coordination functions of SAS proposers periodically pull a common input set $\mathcal{R}(t_{pull})$ from the blockchain database to launch the allocation generation process, where t_{pull} refers to the record pulling time. The pulled input set, $\mathcal{R}(t_{pull})$, contains all the active requests before the pulling time t_{pull}. With this common input set, the proposers generate spectrum allocations according to their own proprietary algorithms Alg_j. We define the set of allocations of a SAS proposer as $\mathcal{A}_j(t)$ and the newest allocations at t_{pull} is $\mathcal{A}_j(t_{pull})$. By definition, we have $\mathcal{A}_j(t_{pull}) = Alg_j(\mathcal{R}(t_{pull}))$. $\mathcal{A}_j(t_{pull})$ contains the server's replies to spectrum usage requests, indicating whether they are granted. If a spectrum usage request is not granted, the allocation may provide recommended operation parameters including the power level

and spectrum band(s).

There are many allocation algorithms customized for the spectrum allocation problem including simple first-come-first-served (FCFS) allocation, greedy approaches, graph coloring [64], reinforcement learning [7], and optimization-based algorithms [85].

Within each proposer, the allocation algorithm is usually a separate function from the coordination function. This enables the SAS servers to execute complex allocation algorithms offline without the need to consume on-chain computation resources. The coordination function obtains allocation result $\mathcal{A}_j(t_{pull})$ from the algorithm and generates the allocation transactions $Alloc_{tx}$ accordingly. Each allocation transaction is bound with a request transaction, containing the pulling time, its generation time, the id of the request transaction it replies to, the decision (whether the spectrum usage request is granted and if not, some recommended alternative spectrum resources), the id of the allocation result it belongs to, and the signature of its generator. The allocation transaction can be expressed as $Alloc_{tx} =< t_{pull}, t_{gen}, id_{Req_{tx}}, decision, id_{\mathcal{A}_j(t_{pull})}, sig_{SAS_j} >$. The coordination function within each server sends the allocation transactions to the blockchain database in a batch and waits for the commit confirmations from it to conclude this sub-phase.

Allocation Voting

Upon receiving the commit confirmations of the proposed allocation transactions, the coordination function broadcasts voting requests to all SAS servers. The voting request messages do not contain any operation parameters and only serve as notifications. They are signed by the proposer's private key and the recipients can verify their authenticity. The other SAS servers, upon receiving the voting requests, serve as the voters and start the allocation voting sub-phase to generate replies.

Allocation Evaluation Algorithm. The Allocation Evaluation Algorithm (AEA) evaluates the performance of spectrum allocations. It takes the request set $\mathcal{R}(t_{pull})$ and allocations $\mathcal{A}_j(t_{pull})$ as the input and produces scores indicating the performance of different allocations, i.e. $s = \mathbf{AEA}(\mathcal{A}_j(t_{pull}), \mathcal{R}(t_{pull}))$.

For each spectrum user, $\mathcal{A}_j(t_{pull})$ contains the feedback about whether its request is granted. There could be spectrum usage conflicts between different spectrum requests, and AEA first checks whether $\mathcal{A}_j(t_{pull})$ resolves these conflicts, i.e. whether $\mathcal{A}_j(t_{pull})$ satisfies the interference and priority requirements. A proper allocation shall ensure that the lower tier users cause no harmful interference to higher tier users. If an allocation cannot pass this check, it will get a zero score. For the allocations that have passed the checking process, AEA evaluates their performance in terms of spectrum utilization rate and fairness. We assume the numbers of spectrum usage requests from SAS servers are $l_1, l_2, \cdots, l_n$. The numbers of granted spectrum usage requests in allocation $\mathcal{A}_j(t_{pull})$ are $g_1, g_2, \cdots, g_n$. Note that $g_i \leq l_i$ because not all the requests are necessarily granted.

We define the spectrum utilization rate as the ratio between granted requests and original requests:

$$u = \frac{\sum_{i=1}^{n} g_i}{\sum_{i=1}^{n} l_i} \tag{3.1}$$

For fairness measurement, we choose the well-established Jain's fairness index [86]. Considering $\mathbf{H} = (\frac{g_1}{l_1}, \frac{g_2}{l_2}, \cdots, \frac{g_n}{l_n})$ as the request grant rate of all SAS servers. We define the fairness metric as:

$$v = \frac{(\sum_{i=1}^{n} \frac{g_i}{l_i})^2}{n \sum_{i=1}^{n} (\frac{g_i}{l_i})^2} \tag{3.2}$$

In the ideal situation if the allocation is totally fair, the fairness metric $v = 1$. In the worse

case if the allocation is totally unfair, the fairness metric $v = \frac{1}{n}$.

The final evaluation score s can be expressed as a function of u and v, i.e. $s = f(u, v)$. Different application scenarios may apply different $f()$ to satisfy their unique requirements. For example, s can be the multiplication of the utilization rate u and fairness score v, i.e. $s = uv$, indicating the area the allocation covers in the (u, v) space, as well as serving as a metric to evaluate an allocation. Or the users can set a fairness threshold to only accept allocations that obtain a higher fairness score than the threshold th. In this case, $f(u, v) = \begin{cases} u & \text{if } v \geq th \\ 0 & \text{if } v < th \end{cases}$.

The voters generate voting transactions based on their local calculation of scores through the AEA. Each voting transaction contains the id of the allocation it votes for, the score, its generation time, and the signature from its generator. The transaction can be expressed as: $Vote_{tx} =< id_{\mathcal{A}_j(t_{pull})}, s, t_{gen}, sig_{SAS_i} >$. The coordination function within each voter sends the voting transactions to the blockchain database, which are stored in the ledger and reliably broadcasted to others. After the coordination function gets commit confirmations from the blockchain database, it replies to the proposer with a voting reply message. The proposer waits for $n - f$ replies and upon receipt, broadcasts a voting confirmation message to conclude this sub-phase.

Within each server, the allocation generation sub-phase and the allocation voting sub-phase take place concurrently. This means a server can both serve as a proposer to generate new allocation proposals and serve as a voter to vote for others' allocation proposals at the same time. The number of proposers p is a known parameter for every SAS server. If the number p equals n, all SAS servers propose allocations to compete with each other; and if equals one, there is only one proposer and all the others serve as voters to approve the allocations proposed by it.

Allocation Decision

The final allocation decision step waits for the voting confirmation replies from all committed ($\leq p$) allocations, i.e. those that had sent a voting request message before, until a timeout threshold. The SAS servers select the valid allocation with the highest score $\mathcal{A}_{final}(t_{pull})$ as the final selected allocation. In this context, valid means there are at least $f+1$ identical votes from different servers stored in the ledger. The servers broadcast the id of the final selected allocation $id_{\mathcal{A}_{final}(t_{pull})}$ in the voting decision message and upon receiving $f+1$ identical results signed by different servers, pull $\mathcal{A}_{final}(t_{pull})$ from the database and reply to the users, which conclude the whole coordination process.

In summary, from the spectrum users' perspective, a valid transmission grant corresponds to the following list of transactions: a registration transaction Reg_{tx} containing device and physical operation information, a request transaction Req_{tx} containing the spectrum usage request, an allocation transaction $Alloc_{tx}$ showing the spectrum usage decision, at least $f+1$ identical vote transactions $Vote_{tx}$ showing the validity of the allocation, and at least $f+1$ identical voting decision messages showing SAS servers' endorsements. Note that the voting decision messages do not necessarily need to be stored in the immutable ledger because the stored allocation and voting transactions are enough for checking the integrity and correctness of the voting process. The honest servers will get correct results even if the last id broadcasting and verification is not there i.e. they directly pull the final results back to clients after the selection step. We keep it in the coordination process to prevent Byzantine servers from forcing their subscribers to transmit in the unauthorized band(s) to jeopardize the operation of others because with this step the clients shall only consider a transmission grant to be valid if and only if there are at least $f+1$ endorsements from the SAS servers which Byzantine nodes cannot accomplish.

Algorithm 3 TriSAS-Phase 2: Decision Finalization

Requirement: The number of servers is n, the number of proposers is p, and the desired number of faults to counter f satisfies $f \leq \lfloor \frac{n}{3} \rfloor$.

Ensure: The SAS system is robust against f Byzantine faults and all SAS admins obtain a unified, fair, and verifiable spectrum allocation plan.

procedure Decision Finalization
  function Allocation Generation
    for j `in` $\{1, 2, \cdots, p\}$, each `server` SAS_j **do**
      Pull common input set $R(t_{pull})$.
      Get $A_j(t_{pull}) = Alg_j(R(t_{pull}))$.
      Get $Alloc_{tx} = Convert(A_j(t_{pull}))$.
      Broadcast $Alloc_{tx}$ to others.
      Store received $Alloc_{tx}$ to the ledger.
    end for
  end function
  function Allocation Voting—Proposers
    for j `in` $\{1, 2, \cdots, p\}$, each `server` SAS_j **do**
      Broadcast voting requests to voters.
      Wait for $n - f$ voting transactions $Vote_{tx}$.
      Broadcast voting confirmations to voters.
    end for
  end function
  function Allocation Voting—Voters
    for i `in` $\{1, 2, \cdots, n\}$, each `server` SAS_i **do**
      Calculate score $s = \mathbf{AEA}(\mathcal{A}_j(t_{pull}), \mathcal{R}(t_{pull}))$ upon receiving voting request.
      Broadcast the voting scores to others.
    end for
  end function
  function Allocation Decision
    for i `in` $\{1, 2, \cdots, n\}$, each `server` SAS_i **do**
      Wait for p confirmations from all proposers.
      Select the best allocation $\mathcal{A}_{final}(t_{pull})$ from p proposed allocations.
      Broadcast the identity $id_{\mathcal{A}_{final}(t_{pull})}$.
      Wait for $f + 1$ identical identities.
      Pull $\mathcal{A}_{final}(t_{pull})$ from the ledger and replies final results to the clients.
    end for
  end function
end procedure

With all these transactions and messages, the users know that the spectrum usage decisions they have received are reliable and agreed upon by the majority of the servers. The users can verify the transactions and decisions in the blockchain ledger. We demonstrate the data structure of a valid transmission grant in Figure 3.3.

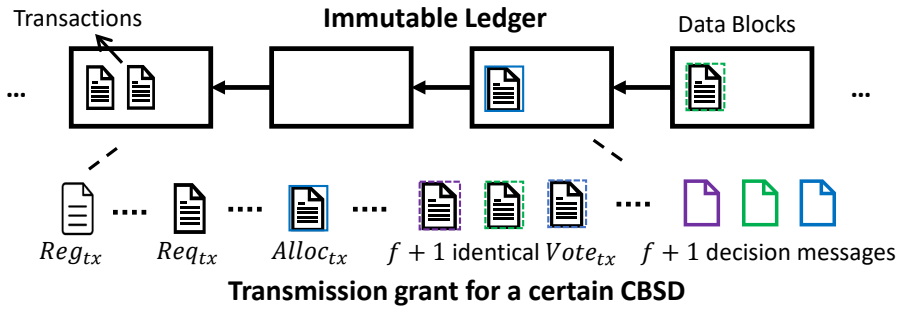


Figure 3.3: The data structure of a valid spectrum transmission grant.

3.4.4 Analysis

State Synchronization. The coordination function waits for the blockchain database to reply with a commit confirmation each time it sends a transaction. Because the blockchain database itself has an underlying Byzantine fault-tolerant consensus mechanism, which has been well investigated and proved such as PBFT and Tendermint, all the transactions are stored in distributed immutable ledgers across all servers. The state $\mathcal{S}_i(t)$, including the input records $R_i(T)$, allocation records, and vote records are synchronized when the coordination process is accomplished.

Decision Correctness. Each proposed allocation receives $n-f$ replies from the voters (including the proposer himself) before it proceeds. Among them at most f votes are malicious. When $n \geq 3f+1$, the remaining $n-2f$ honest votes are identical and take up the majority. Therefore, by majority voting, the Byzantine servers cannot affect voting scores of honest allocations. However, the proposers themselves can be Byzantine and propose allocations.

But a Byzantine allocation proposed by a Byzantine server cannot have a higher score s than normal allocations, otherwise, it is a correct allocation. As a result, Byzantine allocations cannot be selected as the final allocation and the decision correctness is ensured. Our mechanism has a timeout threshold, which means different servers need to be synchronized. We consider this assumption to be practical because current telecommunication networks usually have stringent time synchronization requirements and many protocols such as the network time protocol (NTP) [124] and precision time protocol (PTP) [55] are used for this purpose.

Complexity. We denote the number of servers as n and the number of CBSDs as m. The message complexity of our coordination mechanism is $\mathcal{O}(n^2)$. The memory consumption is $(n+2)m+n^2$, or considering $m >> n$ is $\mathcal{O}(mn)$. More specifically, within one coordination round the ledger stores m registration transactions Reg_{tx}, m request transactions Req_{tx}, n proposed allocations with each one containing m replies ($alloc_{tx}$) to the requests and n^2 vote transactions $Vote_{tx}$.

3.5 Evaluation

3.5.1 Experiment Setting

We implemented a prototype of TriSAS on the Amazon AWS cloud computing platform. Our system consisted of {4, 7, 10, 13} EC2 instances serving as SAS servers and one desktop in our lab serving as spectrum users. Each server instance was an EC2 T2.Large node that had two vCPUs, one 8GB memory, and one 32GB disk. They were located in four different areas across the U.S., including northern Virginia, Ohio, Oregon, and northern California. This simulated the real-life deployment of SAS servers because in reality servers of different

SAS administrators are deployed in their own data centers across the country. The network latency between the desktop in our lab and cloud servers is shown in Table 3.2. We chose Bigchandb as the blockchain database. Each server installed a Bigchaindb implementation and we configured them together to form a Bigchaindb network. Bigchaindb is a representative and well-documented blockchain database. We leveraged the Python programming interface of Bigchaindb, i.e. the Python driver of Bigchaindb to implement our coordination mechanism. Bigchaindb provides two types of transaction templates including the *create* transaction template and *transfer* transaction template. We used the *create* template for the registration transaction Reg_{tx} and the *transfer* template for the other types of transactions, i.e. Req_{tx}, $Alloc_{tx}$ and $Vote_{tx}$. The contents of these transactions were included in the *asset* field and *metadata* field of the templates. Bigchaindb uses the Tendermint consensus mechanism as the underlying consensus protocol. Upon initialization, the Tendermint consensus mechanism generates a pair of elliptic curve-based public and private keys for each participant. We used this key pair to identify nodes and sign transactions.

Table 3.2: Round trip times (in milliseconds) among SAS nodes in different areas.

Node	Location	1	2	3	4	5
1	Lab (N.Virginia)	-	-	-	-	-
2	AWS (N.Virginia)	7	-	-	-	-
3	AWS (Ohio)	21.5	11.2	-	-	-
4	AWS (California)	84	60.6	51.4	-	-
5	AWS (Oregon)	72	63.5	48	20.2	-

The key evaluation metrics are the system's throughput and latency under a certain volume of input traffic. Within a real-life spectrum sharing zone, which is typically a county, there are about 400 CBSDs and each of them can send a request to the server per heartbeat interval, which by the WInnForum standard is a 300-second interval [191]. Together this contributes to 80 transactions per minute (TPM) input rate. This is the maximum input rate because most of the time the CBSDs just send a heartbeat message without any meaningful content.

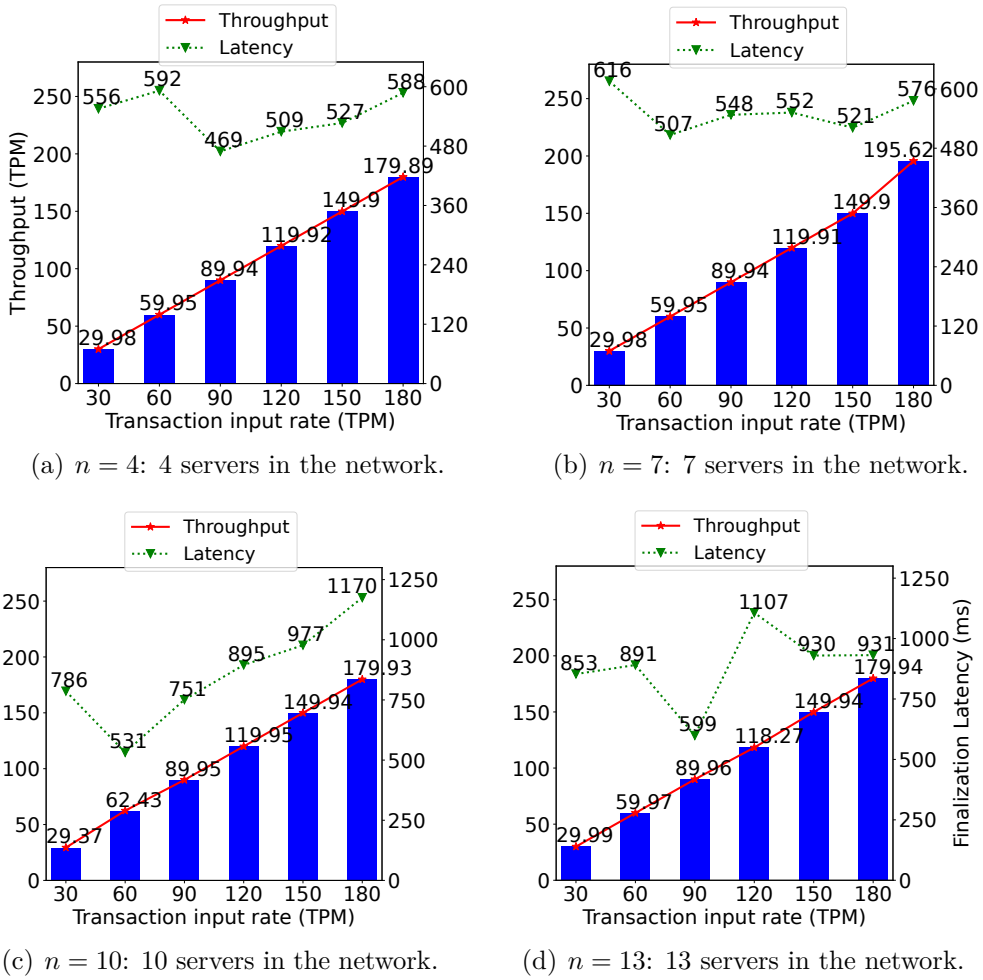
We can expect the usual traffic volume to be on a scale of 10 TPM. In our experiment, we conducted stress tests for the system and chose the following input rates: {30, 60, 90, 120, 150, 180} TPM. Each request contained its desired spectrum bands and location. We assumed there are 100 locations and 15 spectrum bands (one band is 10 MHz), and each time a request randomly asked for three to five bands and one location.

Latency is another important evaluation metric, especially for the second decision finalization phase. This is because the second phase is conducted periodically and must be finished within the interval T, otherwise, the coordination mechanism fails. The decision finalization interval is expected to be shorter than the heartbeat interval (300s) because if this is guaranteed, the users can get real-time replies in the next heartbeat communication with servers. In our experiment, we chose the interval as {30, 60, 90, 120} seconds under a fixed input rate.

3.5.2 Input Synchronization Performance

In Figure 3.4 we demonstrate the experiment results of the input synchronization phase. The figures show the throughput and latency performance when the network size increases from $n = 4$ to $n = 13$, which can tolerate 1 to 4 Byzantine faults. We can observe that their output throughput rates are monotonically increasing and follow a linear relationship with respect to the input rates. This implies that the system throughput has not yet reached its bottleneck and may increase even with larger input rates. We can also observe that, for all settings, the system throughput is almost identical to the input rate, showing that the request messages can be processed in real time without buffering.

The latency of the input synchronization phase, i.e. the time interval between the transactions proposal and commitment, is very stable when the network size $n = 4$ and $n = 7$. It is about 550 milliseconds for every input rate. But when the network becomes larger, the



(a) $n = 4$: 4 servers in the network.

(b) $n = 7$: 7 servers in the network.

(c) $n = 10$: 10 servers in the network.

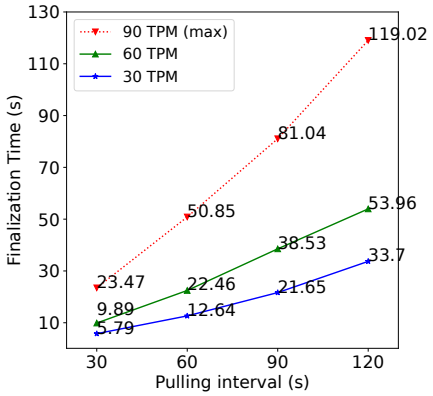
(d) $n = 13$: 13 servers in the network.

Figure 3.4: Input synchronization throughput and latency performance. When n={4, 7, 10, 13}, the system can tolerate {1, 2, 3, 4} Byzantine faults.

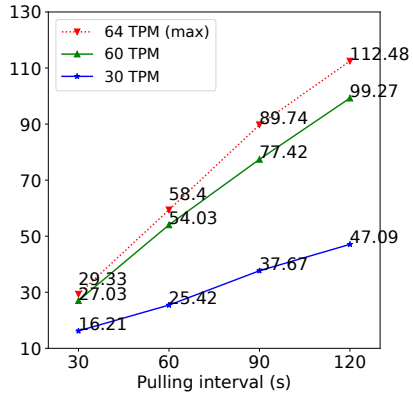
latency increases to 800 milliseconds to 1 second when $n = 10$ and $n = 13$. We consider this to be because larger networks impose more communication overhead for the system, resulting in longer processing delays.

3.5.3 Decision Finalization Performance

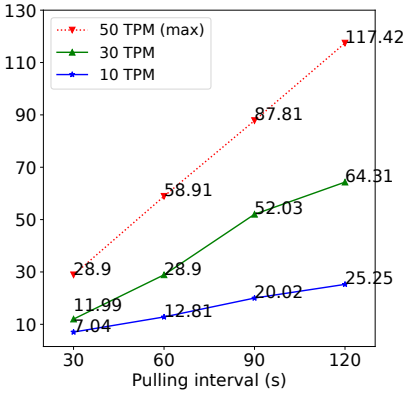
In Figure 3.5 we demonstrate the performance of the decision finalization phase. In the experiment, we mainly focused on the latency performance of this phase. We changed the



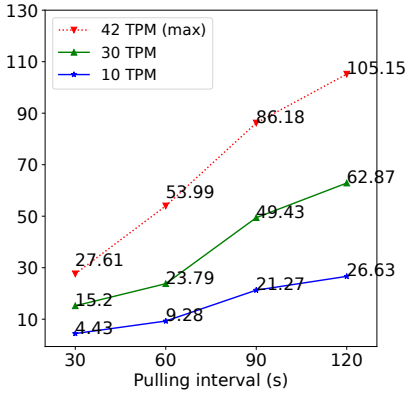
(a) $n = 4$: 4 servers in the network.



(b) $n = 7$: 7 servers in the network.



(c) $n = 10$: 10 servers in the network.



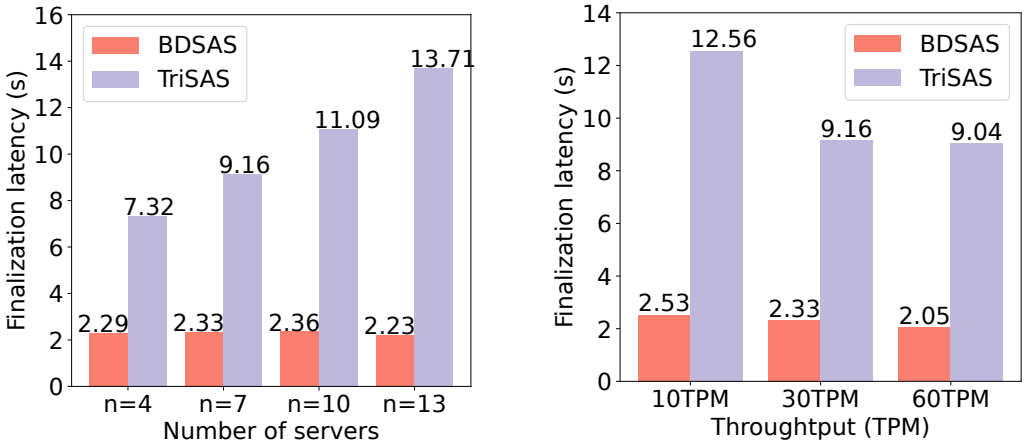
(d) $n = 13$: 13 servers in the network.

Figure 3.5: Decision finalization latency performance. When n={4, 7, 10, 13}, the system can tolerate {1, 2, 3, 4} Byzantine faults.

size of the network from $n = 4$ to $n = 13$. For each network size, we checked the system's performance with respect to a normal input rate (10 or 30 TPM) to its maximum stable input rate, which was obtained as the maximum input rate that keeps the processing latency smaller than the interval T, meaning that the decision finalization phase can be finished within the interval.

For each case, we can observe that the processing latency is significantly increased when the input rate changes from the normal rate to the maximum rate for a fixed polling interval.

This is because a higher input rate contributes to a larger number of request messages $\mathcal{R}(t)$, which consumes more processing time for the transaction commitments and allocation generations. For a fixed input rate, we can observe that the latency linearly increased with respect to the polling interval. This is because the system gets more requests for larger pulling intervals and needs more time to process them. When the network size becomes larger, we find that the performance decreases as the system has smaller maximum stable input rates and longer processing latency. For example, the maximum stable input rate decreases from 90 TPM to 64 TPM when n increases from 4 to 7; and if we fixed the input rate to 30 TPM, the processing latency when $n = 7$ is significantly larger than that of $n = 4$.



(a) Performance verses server numbers. (b) Performance verses throughput.

Figure 3.6: Latency performance of TriSAS and BD-SAS under different network sizes and input rates.

3.5.4 Benchmark Comparison

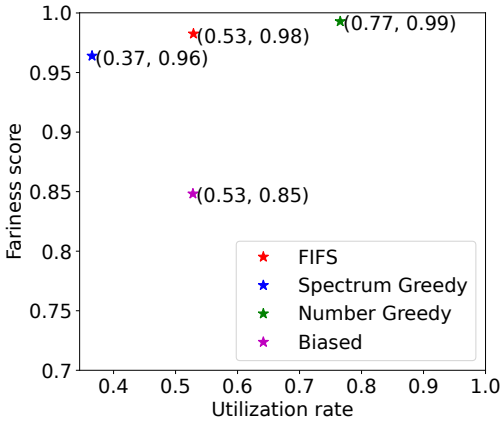
We compared the performance of our mechanism with BD-SAS [197], another blockchain-based decentralized SAS following the setting that the system processes the requests sequentially one by one instead of the current default periodical batch-based processing routine to make fair comparisons because BD-SAS does not support the second way. We demonstrate

the performance of TriSAS and BD-SAS in figure 3.6. We focused on the average latency of both mechanisms to process one request.

We observe that when the network size increases from $n = 4$ to $n = 13$, BD-SAS keeps a very stable processing latency at about 2.2 seconds, while TriSAS has a larger latency that increases from 7.32 seconds to 13.73 seconds under 30 TPM input rate. For a fixed $n = 7$ network, we find that BD-SAS's processing latency is still stable at about 2.3 seconds while TriSAS's latency varies from 9.04 seconds to 12.56 seconds. From the result, we find that TriSAS imposes a much larger overhead than BD-SAS. We consider this is because BD-SAS adopts a much simpler service model as it relies on smart contracts to fulfill the spectrum assignment and cannot implement complex allocation algorithms, execute them off-chain, or allow algorithm diversity, which is crucial for inter-SAS coordination.

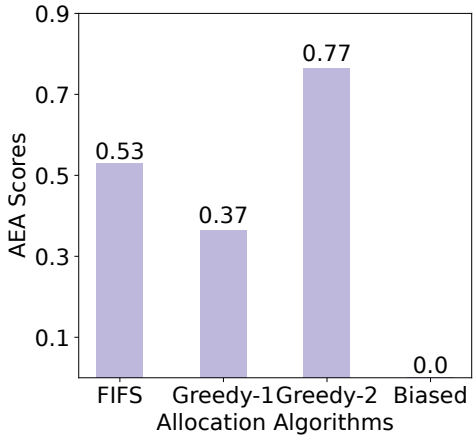
3.5.5 Fairness Performance

We conducted a simulation to evaluate the performance of several allocation algorithms including first-come-first-serve (FCFS), spectrum greedy, number greedy, and biased allocation algorithms with our AEA algorithm. FCFS means that if two requests ask for the usage of the same spectrum band in one location, the earlier request is approved and the latter one is rejected. Spectrum greedy means that if two requests have conflicts, the one with larger spectrum bands wins. Number greedy means that the algorithm tries to maximize the number of granted requests. The biased allocation algorithm tries to maximize the total number of grants for a certain server. We assumed there were 300 spectrum usage requests toward 100 locations with each one asking for a random amount of spectrum band(s). We repeated the simulation for 100 trials and within each trial, every allocation algorithm generated one allocation to obtain an AEA score. We considered the average performance of the 100 trails



(a) U (utilization)- V (fairness) map.

(b) Performance when $s = uv$.



(c) Performance when $th = 0.9$.

Figure 3.7: Evaluation performance of different spectrum allocation algorithms.

as the final performance of each allocation algorithm.

We demonstrate the experiment result in figure 3.7. We first plot the (u, v) map, i.e. the utilization rate and fairness score map of the four allocation algorithms. For fairness performance, we can observe that the biased allocation algorithm achieves the lowest score, which is consistent with our intuition. For the utilization rate, the number greedy allocation algorithm achieves the best performance. To further provide overall unified and quantitative measurements for different algorithms, we calculate two different AEA scores as $s_1 = uv$ and $s_2 = \begin{cases} u & \text{if } v \geq th \\ 0 & \text{if } v < th \end{cases}$, with the first one indicating the area the algorithms covered in

the (u, v) map, and the second one enforcing a fairness threshold. We can observe that the number greedy algorithm achieves the best performance for both two scores. Therefore, if the SAS administrators use the aforementioned allocation algorithms to generate spectrum allocations, the one employing the number greedy algorithm shall win the voting phase with the highest probability. We clarify that in reality, the SAS administrators may use much more complex allocation algorithms than the four we mentioned. However, they can still use AEA to evaluate their performance and fulfill the decision-finalization phase in our design.

3.6 Discussion

Scalability. In this work, we mainly focus on the spectrum sharing problem within one zone. However, in reality, the SAS administrators may conduct spectrum scheduling and inter-SAS coordination in many zones across different areas simultaneously. In this case, the number of spectrum users and input volume is significantly increased. The bottleneck of the current implementation is that we use the Amazon EC2 computing instances and these nodes have limited computation and memory resources. Their computation and memory power is only comparable to a daily-used desktop and certainly can not handle very large user numbers and input volume. On the bright side, all the SAS administrators are major corporate entities with access to high-performance computing clusters. Therefore, we can expect them to have enough resources to handle large-scale and complex inter-SAS coordination problems.

Complex allocation algorithm. For the spectrum allocation algorithm, our current implementation only considered linear-complexity allocation algorithms. We observe that there exist more complex allocation algorithms that utilize optimization techniques to achieve high allocation efficiency and fairness [85]. In this regard, TriSAS can incorporate these complex algorithms in a straightforward fashion since it enables different SAS servers to

use proprietary algorithms. Nonetheless, how to accomplish the vote-and-selection process within a hard time limit, especially when different SAS servers finish allocation generation at disparate times, is an outstanding issue and we leave this problem to future work.

SAS client privacy. In our scheme, we assume each SAS server disseminates its local client requests to all other servers, allowing all honest SAS servers to receive the same super-set of client requests. In practice, however, different SAS providers may be reluctant to share their client data with each other, at least in its original form, to protect the competitive advantage of their algorithms and the privacy of locally subscribed clients. To address this privacy concern, we identify two potential extensions of TriSAS. First, a SAS server may anonymize or obfuscate sensitive information about client requests, such as device ID and location, while preserving a high level of allocation accuracy. Differential privacy techniques can be used to help each client establish a privacy budget [52]. Second, a SAS server may employ a server-level trusted execution environment (TEE) solution, such as Intel SGX [42] and AMD SEV [91], to compute over confidential client requests (to decrypt inside the TEE enclave with keys secretly provisioned from other SAS servers). A program integrity proof can be provided to other SAS servers through remote attestation. Meanwhile, how to manage the decryption keys among SAS servers would require a secure design.

3.7 Related Work

State Machine Replication and BFT Consensus. When it comes to realizing a fault-tolerant distributed computing service that achieves uniform decision among participants, state machine replication (SMR) is heralded as the *de facto* paradigm [24, 35, 157], where a consortium of replicated servers provides consistent computation service in response to a sequence of client requests, despite a certain portion of faulty servers. Based on the type of

faults they address, the SMR-based consensus protocols can be classified into crash fault-tolerant (CFT) consensus protocols and Byzantine fault-tolerant (BFT) consensus protocols. Paxos [103], Raft [79] and Zab [148] are typical CFT consensus protocols. They can tolerate crashed faults when the majority of the servers behave. Many industrial distributed computing systems such as Apache Hadoop [168] and Apache Kafka [99] use them as the fundamental consensus mechanisms. The BFT consensus protocols address Byzantine failures, which exhibit arbitrary behaviors due to malicious hacks, device crashes, and network failures. Compared to CFT consensus protocols requiring crash faults to comprise less than half of the total population, BFT consensus protocols counter Byzantine faults when their population is less than one-third. PBFT [35], Zyzzyva [97], and Tendermint [30] are well-known BFT consensus protocols.

Blockchain-based SAS. Prior works have explored using block-chain smart contract to implement a spectrum allocation mechanism directly [18, 196] or to aid the current SAS server in query aggregation and allocation publication [69, 210]. It is further explored in TrustSAS [69] and BD-SAS [196] that a two-layer blockchain framework may provide further scalability benefits. However, as we have discussed in Section 3.2, blockchain-based SASs have certain limitations and cannot fully address the inter-SAS coordination problem.

3.8 Conclusion

In this chapter, we investigate the inter-SAS coordination problem in the 5G spectrum sharing systems. We identify the drawbacks of the WInnForum CPAS standard, as it cannot provide a secure and efficient inter-SAS coordination. We further analyze the existing blockchain-based SAS solutions and identify their limitations of being unable to allow diverse allocation algorithms owned by SAS servers, as well as failing to enable efficient off-chain

execution of them. To address this problem, we propose TriSAS, a two-phase coordination mechanism that not only provides security guarantees but also ensures high system throughput and low processing latency. We implemented a prototype of TriSAS on the Amazon AWS platform and conducted extensive experiments to evaluate its performance. The results demonstrate that TriSAS can be effectively deployed in real-world systems. This work contributes to the state-of-the-art inter-SAS coordination and communication, an important problem that is often ignored by the community. This problem may also involve commercial, public policy, and enforcement activities, which together contribute to a healthy spectrum-sharing market.

Chapter 4

Multimodal Data Leakage in Medical Federated Learning Systems

(Copyright Notice[1])

4.1 Introduction

Federated learning (FL) has developed as a key enabling technology for the future implementation of AI-powered medical diagnosis and treatment systems [9, 29, 44, 127, 129, 139, 152, 161]. FL allows medical centers to collaboratively train machine-learning models for various clinical tasks such as disease classification and clinical diagnosis, without sharing private patient information. This is crucial because medical centers are bound to preserve patient privacy and their data usage is strictly restricted in many clinical applications in accordance with regulatory guidelines. Under the FL framework, distributed training is set up in a way where hospitals that own private clinical data usually serve as clients, and a server – either hosted at one of the collaborating sites or maintained by a third party, in-

[1]This chapter previously appeared as a part of a conference paper published in IEEE/ACM CHASE 2025. ©2025 Copyright held by the owner/author(s). Reprinted, with permission, from Shanghao Shi, Md Shahedul Haque, Abhijeet Parida, Chaoyu Zhang, Marius George Linguraru, Y. Thomas Hou, Syed Muhammad Anwar, and Wenjing Lou, "MedLeak: Multimodal Medical Data Leakage in Secure Feder- ated Learning with Crafted Models," In ACM/IEEE International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE'25), pages 245-256, 2025 [166].

tegrates the model updates received from each client to orchestrate the federated learning paradigm. There are multiple open source (such as NVFLARE [154] and OpenFL [149]) as well as commercial platforms (such as Rhino FCP [173]), designed to streamline the implementation of FL. During the FL training process, only the model updates, which refer to either the gradients or parameter updates, are exchanged between the participant and the server, while the private training samples are kept securely at the clinical site. Therefore, when first introduced, federated learning was considered to be privacy-preserving and the model updates are regarded as safe vectors that hide training samples' private information [121, 141, 187, 188, 209].

Existing Privacy Attacks Recent privacy attacks challenged the privacy-preserving property associated with FL. It is demonstrated that a curious or malicious parameter server can extract information related to the training samples such as their labels, membership information, and even the whole training sample using the model updates [61, 62, 65, 116, 126, 185, 205, 211, 217]. Of particular interest, the model inversion attacks (MIAs) [61, 65, 138, 162, 205, 217] are a type of privacy attack aiming to recover the original training images. They take the individual model updates provided by the clients as inputs and reverse them back to the local training samples. This could be detrimental in medical applications, where such an attack can reconstruct patient-specific data. Existing MIAs usually formulate this reverse process as an optimization problem and have been shown to achieve good recovery performance in recovering high-fidelity training images from the model updates, when enough optimization iterations are performed. This completely exposes the information that the FL system has been designed to protect.

However, existing optimization-based MIAs are facing serious scalability and efficiency challenges. In practice, they need the server to consume extensive resources (usually hundreds

of computation seconds with large memory) to recover only a few images, making them virtually impractical for real-world systems. Further, such MIAs can also be prevented by a specialized multi-party computation (MPC) mechanism named secure aggregation (SA) protocol [27, 31, 142], whose fundamental idea is to use various cryptography primitives (e.g., secret sharing) to mask individual model updates with random values but keep their summation identical to the pre-masked value. In this way, the FL system can proceed to the training process without exposing *individual model updates*, preventing the MIAs from reversing them back to local training samples.

Our Attack In this chapter, we propose a novel attack name MedLeak that addresses the limitation of the existing works. Our attack design makes it very practical, which could be detrimental to privacy in existing FL systems. MedLeak is an efficient attack, capable of recovering hundreds of training samples in a batch from the victim client within just a few seconds. MedLeak can also break the SA protocols as it can recover the training samples directly from the *summation value* of the model updates even though the individual ones are cryptographically masked. In addition, MedLeak can be accomplished within one FL training round, making it very practical and stealthy. To further evade detection, the attacker can launch the attack in the initial FL training rounds to avoid hurting the FL training performance. A greedy attacker can even launch the attack multiple times during continuous training rounds to harvest as many local training samples as possible.

Technically, MedLeak is a two-phase attack including the attack preparation phase and the sample recovery phase. In the first phase, the attacker adds an additional two-linear layer module in front of the original model architecture and initializes the module with customized parameters before sending it to the clients. For the target victim, the attacker initializes the parameters of the two-layer module to form a "linear leakage" module with the help of an

auxiliary dataset that has the same data format and distribution as the training samples. This "linear leakage" module is a powerful mathematical tool that can perfectly reverse its gradients back to its inputs, which are identical to the training samples because we place this module as the first component of the model architecture. For other clients, their two-layer modules' parameters are crafted to form a "zero gradient" module, aiming to zero out their gradients and model updates. By doing so the aggregated model update is identical to the model update of the victim because all the others are set to zero, rendering the SA protocols useless.

As medical data is usually composed of both image and text records, we explore and extend MedLeak's capability to recover both data modalities. Particularly, the recovery of medical text records is largely ignored in the existing literature and we are proposing *the first* work in this direction to our knowledge. Compared to medical images, the recovery of medical text records is more challenging because they are discrete natural language words with different paragraph lengths in the input space, resulting in completely different FL model architectures and parameter calculation methodologies. To address this challenge, we slightly modify and customize our attack to insert the malicious module after the embedding layer rather than at the front of the whole model to first launch the recovery word embedding vectors, and then further reverse these word embedding vectors back to input tokens (words).

We evaluated MedLeak on MedMNIST [200], COVIDx CXR-4 [186], Kaggle Brain Tumor MRI [130] datasets for medical images, and on the MedAbstract dataset [158] for medical text records. For medical images, we evaluated our attack performance using the recovery rate, structural similarity (SSIM) score, peak signal-to-noise ratio (PSNR) score, and attack time. Our results show that MedLeak achieves excellent performance on these datasets to recover hundreds of images simultaneously with high recovery rates and quantitative scores, with only a few seconds of execution time. On visual inspection, the recovered images

are virtually indistinguishable from the original images. We compared the performance of MedLeak with three existing MIAs. Our results demonstrate that MedLeak achieves better quantitative scores and is comparatively much more efficient. We further fed the recovered images to downstream disease diagnosis tasks. Our results show that the recovered images achieved a classification performance close to the original images, validating the effectiveness of our attack. For medical text records, we evaluated our attack performance using the recovery rate, word error rate (WER), and attack time. The results show that MedLeak can accurately recover tens of long clinical data paragraphs (e.g., descriptions of patients' health conditions) spanning up to hundreds of words simultaneously. This highlights MedLeak's strong capability to recover non-image modality records effectively and efficiently.

In summary, in this chapter we present the following contributions:

1. We propose MedLeak, a novel and powerful MIA that is capable of recovering high-quality local training samples in large batches using model updates from FL clients efficiently, even when state-of-the-art cryptography-based defense mechanisms such as secure aggregation are employed.

2. Our attack represents a fundamental and practical privacy vulnerability of the medical FL system as it compromises individual clients' privacy. MedLeak can target both medical image and text data, demonstrating its broad applicability in the medical domain.

3. We provide rigorous mathematical analysis and proof for our attack. Our proposed attack design is closed-form, hence avoids incurring any computation-intensive optimization and significantly reduces the computational costs when compared to existing MIAs.

4. We implement MedLeak on medical images and text datasets under different practical

assumptions in the FL systems. The results show that our attack can nearly perfectly recover different types of local training samples of the target victim.

4.2 Background

4.2.1 Federated Learning

We consider for each training round t, there are n clients denoted by $\mathcal{C} = \{c_1, c_2, \cdots, c_n\}$ to be selected by the parameter server S to collaboratively train a global model $G = f_\theta : \mathcal{X} \to \mathcal{Y}$, with each client c_i holds a local dataset D_i. In detail, the parameter server S first publishes the global model parameters θ^t to the clients. Then each client trains the received global model G_t for L_i^t local rounds over D_i to generate its model update δ_i^t. Note that when $L_i^t = 1$, the model update δ_i^t can be replaced by the gradient g_i^t. After that, the client c_i sends the model update δ_i^t back to the server S and the server employs model aggregation (such as using the federated average (FedAVG) algorithm [121]) to conduct the training process:

$$\theta^{t+1} = \sum_{i=1}^{n} \alpha_i \delta_i^t, \tag{4.1}$$

where α_i is the weight assigned to client c_i. The summation of all weights $\{\alpha_i\}_{i:c_i \in \mathcal{C}}$ is 1 and can be adjusted according to the size of local datasets D_i^t to avoid training bias. The server may also employ alternative aggregation strategies (such as FedSGD algorithm [121]) for model aggregation. In the following text, we will omit the notation t, because our attack and analysis are all conducted in a single FL training round.

Federated learning has been widely used in the healthcare domain, enabling different healthcare providers to collaborate with each other towards accomplishing the training of machine

learning models for both natural language processing and imaging tasks [110, 113, 141]. Such clinical FL systems usually leverage the patients' private information as the local training datasets. These can be the patients' radiology scans, textual reports, and tabular records describing the patients' visits and health conditions. This information is considered to be highly private and sensitive and is protected by strict governance laws such as HIPPA and GDPR. The medical FL system is deployed to protect such privacy by ensuring that the sensitive data never leaves the firewalls within the healthcare sites during the model training process. However, herein we will demonstrate with MedLeak that the privacy-preserving property of current FL systems is under challenge.

4.2.2 Model Inversion Attacks

The model inversion attacks (MIAs) take the individual model updates δ_i as the inputs and aim to reconstruct them back to the local datasets D_i held by the clients. This reversion problem has been formalized as an optimization problem, represented as $\arg\min_{\hat{D}_i}[d(\nabla\hat{D}_i - \nabla D_i)]$, where $\hat{D}_i$ refer to randomly initialized dummy samples and $d()$ refers to a distance function such as the second norm distance. To solve this optimization problem, [217] utilizes the L-BFGS optimizer to reconstruct the dummy samples iteratively step by step until reaching a good optimization point. It is further improved by an analytical method that aims to recover the ground-truth labels of dummy samples from the gradients [211], which significantly eases the optimization task and helps accomplish better attack performance. Later works have improved the optimization tools and focus on recovering larger batches of images on more practical machine learning models such as the ResNet [65, 115, 190, 205]. However, their recovery sizes are still restricted to the scale of tens, representing a scalability challenge. Moreover, all the aforementioned MIAs require costly iterative optimization methods in their design, which introduce a very large overhead to recover each batch of input images.

Further, these existing methods also cannot bypass the current SA protocols.

4.2.3 Secure Aggregation

To enhance the privacy of the federated learning systems, Bonawitz et al. [27] proposed a new type of specialized MPC mechanism named the secure aggregation (SA) protocols to fulfill an abstract function of masking individual model updates δ_i to u_i with random bits, while keeping the summation of the masked values $\sum_{i=1}^{n} u_i$ identical to those of the pre-masked values $\sum_{i=1}^{n} \delta_i$. Therefore, despite variations of detailed cryptographic design, all SA protocols ensure that the server *cannot* obtain the individual model updates δ_i to launch any model inversion attacks, but can proceed with the FL training process with the aggregated model update $\sum_{i=1}^{n} u_i$, which is identical to $\sum_{i=1}^{n} \delta_i$. Since its initial introduction, the SA protocols have been continuously refined to incorporate other properties including communication efficiency, drop-out resilience [23, 40, 72, 90], and security against malicious clients [31, 142], making it the current state-of-the-art privacy protection mechanism for FL systems.

Secure Aggregation under Challenge: Under the honest-but-curious attack model, the SA protocols have proven to be secure against various MIAs. However, recent works adopt a stronger attack model to assume a *proactive* attacker that modifies the global model's parameters and even its architecture before publishing it to the clients. Under this assumption, [138] proposed a novel attack that retrieves a target individual model update from the aggregated result, breaking the SA protocols. The fundamental idea is to craft the model parameters to adversarial models and distribute different adversarial models to different clients strategically. The adversarial models are crafted to ensure that only the model update of the victim client is preserved while all the others are zeroed out. The limitation of this

attack is that it incorporates a costly optimization process in its design, which introduces too much attack overheads. [61] proposed another attack method to add crafted modules before the original model architecture. These additional modules are crafted with delicate mathematical designs to ensure that the model gradients can be perfectly reversed back to inputs whenever the server receives any model updates. The limitation of this attack is that the attacker cannot link the recovered images to their owners, which means the SA protocols still preserve a certain level of privacy, known as "privacy by shuffling". Later, [213] addressed this issue by designing a more complex adversarial module composed of convolutional and linear layers before the original model to identify the client associated with the recovered images. However, using a computer vision-related architecture prohibits the attack from being adopted to recover text records.

Table 4.1: A comparison between the existing MIAs and MedLeak with respect to their attack assumptions, efficiency, scales, capabilities, and attack generalizability.

Attacks	**Attack Model**	**Attack Efficiency**	**Scale**
[65, 115, 190, 205, 211, 217]	Honest-but-curious	Low (Optimization-based)	10^1
[138]	Malicious server	Low (Optimization-based)	10^1
[61, 213]	Extra attack module	High (Mostly closed-form)	10^2 to 10^3
MedLeak	Extra attack module	High (Closed-form)	10^2 to 10^3
Attacks	**Break SA?**	**Recover Text?**	**Targeted?**
[65, 115, 190, 205, 211, 217]	No	No (Image-only)	No
[138]	Yes	No (Image-only)	Yes
[61, 213]	Yes	No (Image-only)	No
MedLeak	Yes	Yes	Yes

4.2.4 Attack Summary and Comparison

In Tab. 4.1, we summarize and compare the existing MIAs according to their attack assumptions, efficiency, scales, capabilities, and generability. In our design, we adopt the active attacker assumption to break the SA protocol and simultaneously address various limitations

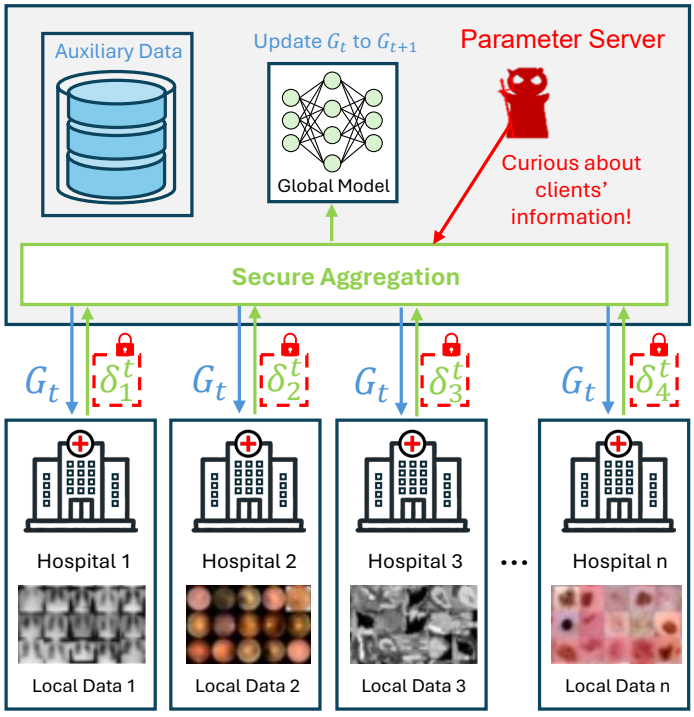


Figure 4.1: MedLeak threat model. The server is considered to be a malicious attacker. The secure aggregation protocol is in place to protect the individual model updates.

of the existing works. In particular, we aim to address the following three problems: 1) *attack efficiency*-our attack shall not employ any computation-intensive optimization process to incur large overheads; 2) *attack effectiveness*- our attack's capability of recovering high-quality local training samples from the *aggregated model updates* and attribute them back to individual clients even when SA protocols are incorporated, and 3) *attack generality*- our attack can be applied to traditional image recovery tasks as well as text recovery task (currently less explored).

4.3 Threat Model

In Fig.4.1 we demonstrate the threat model of our attack. We consider the parameter server S to be a malicious party that is curious about the training samples held by clients

(e.g., private patient images or text records). We consider the parameter server to be a proactive attacker and can actively modify model parameters and architectures from G to $\hat{G}$ to achieve the attack goals, following the same assumption as [61, 138, 162, 213]. We assume the communication channels between the server and clients are secured and all messages can be authenticated. We assume the state-of-the-art SA protocols are in place and the server can only get access to the *aggregated model updates* $\sum_{i=1}^{n} \delta_i$ without knowing anything about the individual values δ_i. We consider the attacker is able to collect or obtain an auxiliary dataset D_{aux} that has the same data format and can represent the target dataset well. The attacker can leverage various online resources such as publicly available datasets, image searching tools, and image generative tools to fulfill this requirement. For our use case, the availability of public chest X-rays and medical text datasets makes this task trivial. The goal of the attacker is to recover the local data samples of a target client c_{target} just from the aggregated model updates $\sum_{i=1}^{n} \delta_i$. This can be mathematically expressed as: $D_{target} = Reverse(\sum_{i=1}^{n} \delta_i, \hat{G})$.

4.4 Attack Method

4.4.1 Attack Overview

As we assume the attacker only possesses the already-masked aggregated model updates $\sum_{i=1}^{n} \delta_i$ under the protection of the SA protocol, it is very challenging to identify an end-to-end method that directly reverses the aggregated results back to local training samples D_{target}. To address this, we decompose the complex recovery problem into two different distinct tasks including the *individual model update retrieval* and *efficient model update reversion* tasks. The first task aims to retrieve the individual model update of the victim

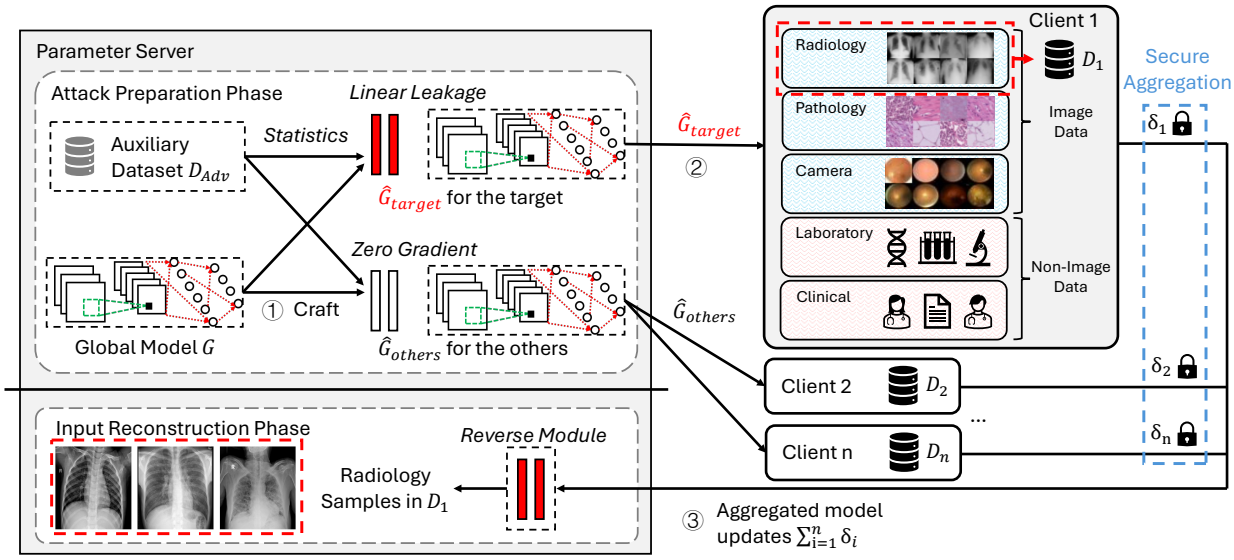


Figure 4.2: MedLeak attack flow. MedLeak is a two-phase attack. In the first preparation phase, the attacker generates the adversarial global model. In the second reconstruction phase, the attacker sends the adversarial models to the clients and recovers the local samples when it receives their feedback. MedLeak can reconstruct both image and non-image data and this figure demonstrates the reconstruction of the medical radiology images.

from the aggregated model updates, thus breaking the SA protocol. The second task aims to reverse the individual model update back to local samples, achieving the recovery sub-problem.

To accomplish them, we require the attacker to place an additional two-linear-layer module L_{adv} with the rectified linear unit (ReLU) activation function in between at the beginning of the original global model, i.e. $\hat{G} = G \oplus L_{adv}$. The dimension of this module is identical to the image dimension. We require the attacker to initialize the parameters of the linear module L_{adv} to form different adversarial modules and distribute them to different clients accordingly. For all the other clients except the victim, the attacker crafts the "zero gradient" modules L_{zero} to ensure that the gradients and model updates of these clients are always zero, i.e. $\hat{G}_{others} = G \oplus L_{zero}$. By doing this, the attacker guarantees that only the model update of the victim client is exposed, accomplishing task one. For the victim client, the attacker crafts

a "linear leakage" module L_{linear}, aiming to reverse its model update back to local training samples efficiently, i.e. $\hat{G}_{target} = G \oplus L_{linear}$. This module requires an auxiliary dataset to help generate essential parameters and can ensure that samples are *perfectly recovered* with a mathematical proof, accomplishing task two. Detailed attack flow and module designs are introduced in the next section.

4.4.2 Detailed Attack Flow

We demonstrate the attack flow in Fig.4.2. MedLeak is a two-phase attack including 1) attack preparation and 2) sample reconstruction phases. In the attack preparation phase, the attacker crafts the adversarial global model $\hat{G}$ and initializes it with different model parameters including L_{zero} and L_{linear} (step ①) before publishing them to different clients (step ②). Then in the second phase, the attacker collects the aggregated model updates and uses an analytical method to reverse it back to local training samples (step ③). MedLeak can be used to reconstruct both medical image and non-image data. In the following parts, we will first introduce the overall attack flow and detailed design components via the image data reconstruction task, and then introduce how MedLeak can be customized to accomplish the text reconstruction task.

Attack Preparation: In this phase, the attacker crafts both the "linear leakage" module and "zero gradient" module once at the outset. Both modules require the estimation of essential parameters of a representative auxiliary dataset D_{aux}. In detail, the attacker first estimates the cumulative density function (CDF) of the brightness feature $h(x)$ of the auxiliary dataset D_{aux}, denoted by $\psi(h(x))$, to represent the CDF of the local training dataset (which is unavailable), where x refers to the input vector. After that, the attacker divides the distribution ψ into equally k bins by calculating $h_j = \psi^{-1}(j/k)$ where $j \in \{1, 2, \cdots, k\}$,

ψ^{-1} refers to the inverse function of ψ, and k equals to the neuron number of the first linear layer. By doing so, the brightness of a random input vector x denoted by $h(x)$ will have the same probability of falling into each bin. This bin vector $\mathbf{H} = [h_1, h_2, \cdots, h_k]$ is the key vector to form both attack modules.

Linear Leakage Module: Assuming the weight and bias matrix of the two-layer module L_{adv} are w_1, b_1 and w_2, b_2 respectively. For the target victim, the attacker initializes the "linear leakage" module L_{linear} with the following steps: (1) having w_1's row vectors all identical to $\left[\frac{1}{d}, \frac{1}{d}, \cdots, \frac{1}{d}\right]$, where d refers to the dimension of the input images, resulting in calculating the brightness feature on each neuron when the local training images are sent into the model during the FL training process; (2) having the bias b_1 identical the opposite value of $\mathbf{H}$, i.e.,$b_1 = -\mathbf{H}$; (3) having all row vectors of w_2 the same.

Zero Gradient Module: The "zero gradient" module is initialized in the same way as the "linear leakage" module except in step (2), in which the attacker has the bias vector b_1 equal to $\mathbf{H}' = -[h_k, h_k, \cdots, h_k]$. By doing so, the output of the first linear layer will always be smaller than zero because h_k is the largest possible brightness and all possible input x's brightness is smaller than it. Considering we use the ReLU activation function after the first layer, the input to the second layer and the gradients of the first layer shall always be zero because of the ReLU function's mathematical property. This results in zero model updates for all clients except the target victim. Therefore, the aggregated model updates are identical to the model update of the victim client, i.e. $\sum_{i=1}^{n} \delta_i = \mathbf{0} + \mathbf{0} + \cdots + \mathbf{0} + \delta_{target} = \delta_{target}$, exposing the model update of the target victim.

Sample Reconstruction: The sample reconstruction phase can be treated as the actual *attack phase*, in which the parameter server disseminates the crafted global models $\hat{G}$ to all clients and recovers the local samples of the target victim D_{target} according to the aggregated

model updates $\sum_{i=1}^{n} \delta_i$ provided by all clients.

More specifically, with the help of the "zero gradient" module, the aggregated model update $\sum_{i=1}^{n} \delta_i$ the attacker obtains is identical to the model update of the victim client δ_{target}, even though the SA protocols are in place. We further argue that the attacker can accurately estimate the gradients from the model update δ_{target} as it equals local iterations of the gradients [121, 162]. We define the gradients of the first two layers of the target victim as g_{w_1}, g_{b_1} and g_{w_2}, g_{b_2} respectively. The attacker can calculate the following equation to reconstruct the input samples, for $l \in \{1, 2, \cdots, k\}$:

$$(g_{w_1}^{(l+1)} - g_{w_1}^{(l)})/(g_{b_1}^{(l+1)} - g_{b_1}^{(l)}), \tag{4.2}$$

where specially we have $g_{w_1}^{(k+1)}$ and $g_{b_1}^{(k+1)}$ equal zero.

Analysis: Equ. 4.2 creates k recovery bins to recover input images. Fortunately, when $k \geq m$, where m refers to the size of the target dataset, each local training sample in the dataset will be *perfectly recovered* within a certain bin ranging from 1 to k. Here perfect recovery means that the inputs are analytically calculated through closed-form mathematical equations. A rigorous mathematical proof for this property is provided in the next section. However, when $k < m$, there will be recovery conflicts, and some recovered samples are mixed with each other in certain bins, resulting in degraded recovery rates and quality. We argue this does not mean the total failure of the image reconstruction task. We will later demonstrate that in this scenario the attack performance gradually decreases and the attack remains to achieve decent performance when attack parameter k is about the same scale as m.

We regard the attack parameter k as the key factor that affects reconstruction performance. Fortunately, from the attacker's perspective, this parameter is controlled and adjustable. The

attacker can have a larger k (i.e. craft larger linear layers) for large datasets and a smaller one for small datasets according to different attack scenarios to ensure that there are enough recovery bins for all samples. Regarding attack complexity, both the attack preparation and sample reconstruction phases only involve closed-form mathematical calculations that are super efficient to be conducted.

4.4.3 Proof of Correctness

Without the loss of generality, we use x_p to denote the local input sample. Considering the input x_p falls in the p^{th} largest bin, i.e. the brightness of x_p denoted by $h(x_p)$ satisfies $h_p < h(x_p) < h_{p+1}$. We have the following equation holds:

$$\begin{aligned}
\frac{g_{w_1}^{(p+1)} - g_{w_1}^{(p)}}{g_{b_1}^{(p+1)} - g_{b_1}^{(p)}} &= \frac{\nabla_{w_1(p+1)} L - \nabla_{w_1(p)} L}{\nabla_{b_1(p+1)} L - \nabla_{b_1(p)} L} \\
&= \frac{\frac{\partial L}{\partial y_{p+1}} \frac{\partial y_{(p+1)}}{\partial w_{1(p+1)}} - \frac{\partial L}{\partial y_p} \frac{\partial y_{(p)}}{\partial w_{1(p)}}}{\frac{\partial L}{\partial y_{p+1}} \frac{\partial y_{(p+1)}}{\partial b_{1(p+1)}} - \frac{\partial L}{\partial y_p} \frac{\partial y_{(p)}}{\partial b_{1(p)}}} \\
&= \frac{\sum_{v=1}^{p} \frac{\partial L}{\partial y_{p+1}} x_v - \sum_{v=1}^{p-1} \frac{\partial L}{\partial y_p} x_v}{\sum_{v=1}^{p} \frac{\partial L}{\partial y_{p+1}} - \sum_{v=1}^{p-1} \frac{\partial L}{\partial y_p}} \\
&= \frac{\frac{\partial L}{\partial y_p} x_p}{\frac{\partial L}{\partial y_p}} = x_p
\end{aligned} \tag{4.3}$$

where L is the loss function, y is the output of the first linear layer, and $\frac{\partial L}{\partial y_{p+1}} = \frac{\partial L}{\partial y_p}$ because we let the row vectors of the w_2 matrix identical.

This equation implies x_p is perfectly recovered from the gradients of the first linear layer. Because $\mathbf{H}$ covers the whole distribution range of the brightness feature, each input image will fall into one bin and thus can be recovered in this way as long as the image number is

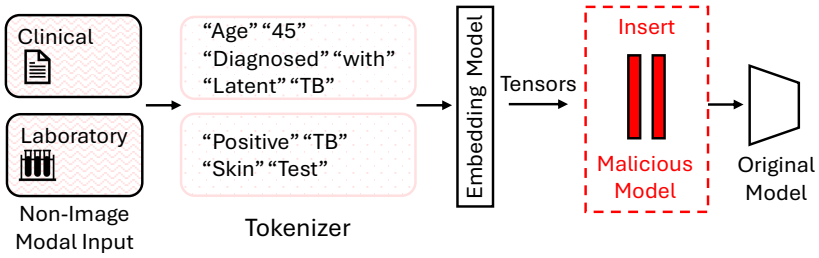


Figure 4.3: An example of how the medical text classification system works and where to insert the malicious modules.

smaller than k.

4.4.4 Text Data Reconstruction

The medical text records are usually natural language words with discrete values within the input space. They have different mathematical properties from the medical images which are continuous pixel values in the input space. As a result, the medical natural language processing (NLP) models usually have different model architectures and workflows compared to their vision counterparts. In Fig. 4.3, we demonstrate the architecture of a text classification model. More specifically, for an input sentence x, the model first converts it to a sequence of tokens $(t_1, t_2, \cdots, t_l)$, where l refers to the maximum length of the sentence. Then the tokens are fed into the embedding model to be converted into the word embedding vectors, represented by $z = [e_1, e_2, \cdots, e_l]^T$, where all word embedding vectors e_i have the same pre-defined embedding dimension. After that, the word embedding vectors are fed into the classification model to produce the output o.

Within this model architecture, our original attack strategy of placing the attack module in the front is not effective because the words are all discrete values and cannot be recovered in the same way as the continuous pixel values. Instead, we insert our attack module L_{adv} including both the "linear leakage" module L_{linear} for the target victim and the "zero

gradient" module L_{zero} for the others between the embedding layer and the classification model. In this way, according to our previous analysis, the attacker can successfully recover the embedding vectors $z = [e_1, e_2, \cdots, e_l]^T$ with the help of these attack modules. The next question is can the attacker further reverse the embedding vectors i.e. e_i back to the tokens t_i? Fortunately, the answer is yes. This is because the word embedding layer maps the discrete tokens into continuous embedding vectors in the embedding space similar to a "lookup table"; therefore, the attacker can simply select the word within the whole word vocabulary that minimizes the distance between the actual embedding vector and the calculated one to be the original token. In practice, we use a single Softmax layer to achieve this reverse function.

Analysis: Similar to the image recovery task, the performance of the text recovery task is largely affected by the number of recovery bins k. We assume during the FL training process, m text data samples are organized in a batch with each one having a maximum length of l words. We clarify that the maximum sentence length l does not affect the recovery performance as long as it is not super large (e.g. 10^4), which we will later justify in the experiment section. The recovery performance is still largely determined by the relationship between m and k. When $k \geq m$, each embedding vector is perfectly recovered within one bin, which further leads to excellent text recovery performance. But when $k < m$, there will be recovery collisions within certain bins, and the text recovery performance drops.

4.5 Evaluation

4.5.1 Experimental Settings

We implemented MedLeak on the PyTorch platform. We ran all the experiments on a server equipped with an Intel Core i7-12700K CPU 3.60GHzX12, one NVIDIA GeForce RTX 3080 Ti GPU, and Ubuntu 20.04.6 LTS.

We considered the FL system to have 5 clients with one of them being the attack target per training round. We assumed each client would perform 5 local iterations before generating the individual model updates. We randomly selected 10% of the training set as the auxiliary dataset and aimed to recover samples in the test set, which have *no intersection* with the auxiliary dataset. We assumed the test set is partitioned and owned by the 5 clients locally to serve as the local datasets. For the defense mechanism, we considered the system to be protected by the SA protocol in [27]. Therefore, we cannot get access to the individual model updates (which have been cryptographically masked) and we launched our attack solely based on the aggregated model updates. Note that our attack can not only break the SA protocol in [27] because MedLeak breaks the abstract aggregation function of the SA protocols regardless of their implementation details. All SA mechanisms realizing the function can be broken, including [23, 31, 40, 72, 90, 142].

For the image recovery task, we chose the ChestMNIST dataset from the MedMNIST package [200], the COVIDx CXR-4 dataset [186], and the Kaggle Brian Tumor MRI dataset [130] as our experiment datasets. The ChestMNIST dataset comprises frontal view X-ray images ($1\times28\times28$) of 30805 unique patients with 14 disease labels and we selected data samples related to pneumonia to conduct our experiments, including 78468 training samples and 22433 testing samples. The COVIDx CXR-4 dataset also consists of frontal-view chest X-ray

images with higher dimensions (resized to 1×224×224) and labels about whether the patient is COVID-positive. The training set contains 67863 images and the testing set contains 8482 images. The Kaggle Brain Tumor MRI Dataset contains 7023 images of human brain MRI images which are classified into 4 classes: glioma - meningioma - no tumor and pituitary. Its training set contains 5712 images and the testing set contains 1311 images.

We used four evaluation metrics including the recovery rate, the attack time, the peak signal-to-noise ratio (PSNR) score, and the structural similarity index measure (SSIM) score, following the convention of the existing works [61, 65, 205, 211, 217] to evaluate our attack over the image recovery task. More specifically, the successful recovery of samples was measured by observing the PSNR and SSIM scores between the original input samples and the reconstructed ones and checking whether those scores exceed a certain threshold th. In our work, we chose $th = 20$ for PSNR and $th = 0.9$ for SSIM because these thresholds are enough to ensure that the recovered images are visually clear for the attacker to extract all meaningful content from them.

For the text recovery task, we chose the MedAbstract dataset [158] as our experiment dataset. The MedAbstract data set consists of 14438 medical abstracts describing the patients' health conditions with each one consisting of a few hundred words. The patients' conditions are classified into 5 different classes including digestive system diseases, cardiovascular diseases, neoplasms, nervous system diseases, and general pathological conditions.

We used three evaluation metrics including the recovery rate, the word error rate (WER), and the attack time to evaluate MedLeak's performance over the text recovery task. Similar to the vision task, the recovery rate was defined as the ratio between the number of successfully recovered text samples achieving WERs lower than threshold 0.05 and the total sample number. Meanwhile, the WER was calculated as the portion of recovery failure words in text samples. In our experiment, we focused on the WER of the successfully recovered

samples to evaluate their recovery quality.

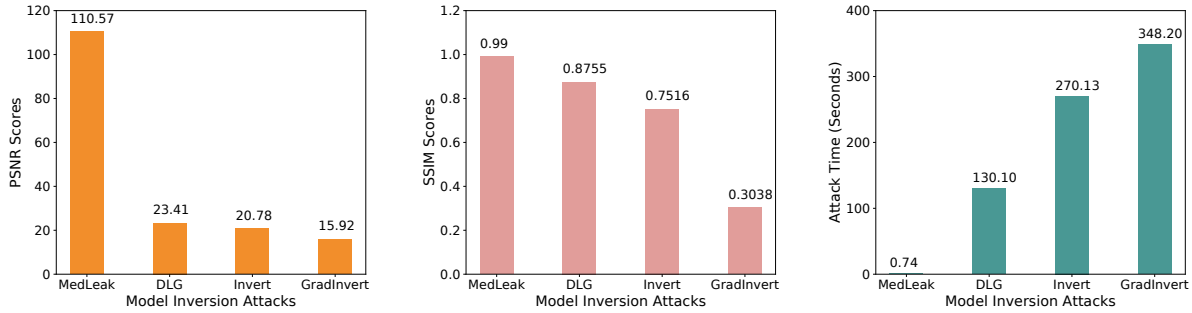
Table 4.2: The reconstruction performance of MedLeak over different datasets and reconstruction batch sizes. The rate (sample recovery rate) is on a scale of 1.00

Batch Size	Dataset	Pixel Size	Rate	PSNR	SSIM	Time (in sec)
100	ChestMNIST(pneumonia)	28x28	1.0	112.574	0.99	0.742
	COVIDx CXR-4	224x224	0.951	120.795	0.99	6.022
	Kaggle Brain Tumor MRI	224x224	0.962	107.783	0.99	6.308
200	ChestMNIST(pneumonia)	28x28	0.960	102.722	0.99	0.936
	COVIDx CXR-4	224x224	0.891	114.982	0.99	7.003
	Kaggle Brain Tumor MRI	224x224	0.870	98.6421	0.99	6.738
300	ChestMNIST(pneumonia)	28x28	0.957	97.405	0.99	0.95
	COVIDx CXR-4	224x224	0.880	105.123	0.99	8.121
	Kaggle Brain Tumor MRI	224x224	0.845	96.722	0.99	7.181
400	ChestMNIST(pneumonia)	28x28	0.955	93.713	0.99	1.020
	COVIDx CXR-4	224x224	0.864	97.301	0.99	8.762
	Kaggle Brain Tumor MRI	224x224	0.804	93.179	0.99	8.042
500	ChestMNIST(pneumonia)	28x28	0.964	87.019	0.99	1.016
	COVIDx CXR-4	224x224	0.810	95.864	0.99	9.763
	Kaggle Brain Tumor MRI	224x224	0.796	92.954	0.99	8.767

4.5.2 Image Reconstruction Results

In Tab. 4.2, we demonstrate the performance of MedLeak over different recovery batch sizes (i.e. the number of samples recovered simultaneously held by the target victim) ranging from 100 to 500 images. We can observe that both the recovery rate (i.e. the ratio of successfully recovered images) and the quantitative scores (i.e. the PSNR and SSIM scores) decrease when the recovered batch size increases. This is expected because the larger the recovery batch size, the more difficult the recovery task to conduct. But in general, MedLeak achieves high recovery rates (> 0.8) and quantitative scores (PSNR> 80 and SSIM> 0.9) for all three datasets under all recovery batch sizes. Particularly, the SSIM scores (ranging from 0 to 1) remain to be 0.99 for all settings, because of the recovery excellency. This can be further verified by the recovered samples we visualized in Fig. 4.4, in which we plot the original images on the left and the recovered ones on the right. We find that the recovered images

Figure 4.4: Recovered examples from the COVIDx CXR-4 dataset, Kaggle Brain Tumor MRI dataset, and MedAbstract dataset. The original images are on the left and the recovered ones are on the right. The text samples are truncated due to space limitations. Recovery failure samples are marked in red rectangles.



(a) PSNR Scores. (b) SSIM Scores. (c) Attack Time (Seconds).

Figure 4.5: The attack performance comparison between MedLeak with other model inversion attacks.

are of high quality and cannot be visually distinguished from the original ones, even for some detailed small marks and notations. In Tab. 4.2 we also demonstrate the attack time (in seconds). We find that the attack time is monotonically increasing with respect to the recovery batch size. For the largest batch size (i.e. 500 images) over the complex COVIDx CXR-4 dataset, it only takes the attacker less than 10 seconds to fulfill the recovery task, indicating that MedLeak is very effective.

4.5.3 Benchmark Comparison

We compared MedLeak's attack performance with three optimization-based model inversion attacks (MIAs) including the DLG/iDLG [211, 217], InvertGradient [65], and GradInversion [205] attacks (denoted as DLG, Invert, and GradInvert respectively) over the MedMNIST dataset for one small batch of input images. We only focused on the image recovery task because all the existing attacks cannot be adapted to the text recovery task. We compared the PSNR scores, SSIM scores, and the attack time between the three attacks and our work. The results are demonstrated in Fig. 4.5. We can find that our attack achieves much better PSNR scores and SSIM scores than the existing MIAs, indicating that our attack can reconstruct samples with better quality. At the same time, our attack consumes significantly

Table 4.3: Downstream binary classification task on the COVID dataset with a pre-trained (with CXR-3 dataset) ViT model. TPR: True Positive Rate, TNR: True Negative Rate, ACC: Accuracy, AUC: Area Under Receiver Operating Characteristic curve, AUPR: Area Under Precision Recall curve.

Model	**Image**	**AUPR**	**TNR**	**TPR**	**ACC**	**AUROC**
ViT-S (SSL)	Original	0.937	0.800	0.857	0.829	0.905
	Recovered	0.921	0.900	0.710	0.805	0.919
ViT-S (Fine-tuned)	Original	0.974	0.970	0.930	0.950	0.969
	Recovered	0.965	0.886	0.938	0.912	0.966

less time than the current MIAs. Particularly, the existing attacks consume a few hundred seconds to reconstruct one batch of samples, while our attack only requires less than one second, which reduces the current cost by two orders of magnitude. The reason why our attack is much more efficient is that our attack only involves closed-form mathematical calculations while the other three attacks require costly iterative-based optimization methods. However, there is no free lunch and we clarify that the three benchmark works adopt an honest-but-curious attack model, which does not allow the attacker to modify the model parameters and architecture as we did.

4.5.4 Vision Downstream Tasks

To further evaluate the performance of our attack on clinically relevant downstream tasks, we performed a binary disease classification (the detection of COVID-19) task on both the recovered samples and the actual samples. We used the state-of-the-art vision transformer model (ViT-S) (embedding size=368, number of heads=6, 22M parameters) pre-trained by self-supervised learning (SSL) technique on 30k COVIDx CXR-3 samples and fine-tuned on the RSNA-RICORD part of the dataset to perform the classification task [16] and evaluated it on the COVIDx CXR-4 dataset. We demonstrate the performance in Tab. 4.3. We use widely used machine learning metrics to evaluate the classification performance and we find

Table 4.4: The text reconstruction performance of MedLeak over a different number of text samples and experiment settings. The rate (sample recovery rate) is on a scale of 1.00. The "Embed Dim" refers to the embedding dimension.

Text Num	Max Length	Embed Dim	Rate	WER	Time (in sec)
20	200 words	64	0.9375	0.0004	0.2149
	300 words	64	0.9669	0.0009	0.3057
40	200 words	64	0.9212	0.0004	0.416
	300 words	64	0.9153	0.0018	0.6023
60	200 words	64	0.8729	0.0005	0.6341
	300 words	64	0.9083	0.002	0.9192
80	200 words	64	0.8228	0.0023	0.8331
	300 words	64	0.8540	0.0051	1.2308
100	200 words	64	0.755	0.0047	1.058
	300 words	64	0.7585	0.0052	1.514

that the recovered images achieve nearly the same performance as the original ones. This shows that our reconstruction process is highly successful in keeping all semantic meaning within the images and the reconstructed images can be used to perform any potential clinical analysis, which further indicates the severity of the privacy threat imposed by our attack. We consider it a very practical attack scenario for a curious party, which either can be the medical federated learning's participants or a third-party service provider (who provides the necessary platform, computation resource, and other FL infrastructures) to launch our attack to first reconstruct the sensitive medical images and then feed them to certain downstream analysis tasks to obtain further information about of the victims.

4.5.5 Text Reconstruction Results

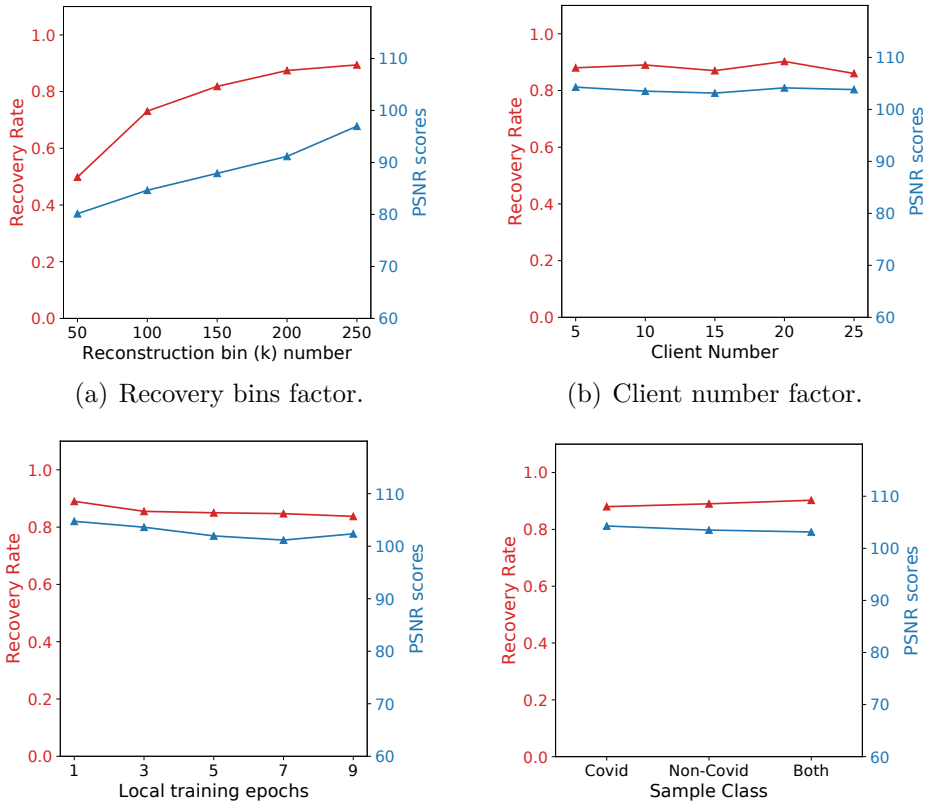
In Tab. 4.4, we demonstrate MedLeak's performance on recovering medical text data under different batch sizes (ranging from 20 to 100) and data settings. More specifically, "text num" refers to the number of text samples recovered simultaneously in one batch, and "max length" refers to the maximum length of the text contents in the number of words. In

our experiment, we fixed the length of each text sample for processing convenience. We truncated the samples when their lengths were longer than the maximum length and padded them when they were shorter. We also fixed the embedding dimension as the commonly used value 64. From the result, we can find that in general, MedLeak achieves decent text recovery performance under different settings to obtain high recovery rates (> 0.75), low WER scores (< 0.006), and short execution time within a few seconds. We observe that when the recovery batch size increases, the recovery rate decreases accordingly, meaning that recovering a larger batch of samples is more difficult. By comparing MedLeak's performance under different sample lengths (i.e. 200 v.s. 300 words), we further observe that a larger sample length triggers longer attack time, which approximately follows a linear relationship with the sample length. However, a larger sample length does not trigger any recovery performance drop. We still observe decent and stable recovery rates and WER scores when the sample length increases. In fact, the recovery rate even increases when the sample length is larger. This may be because longer samples maintain more semantic information and can be better separated and recovered.

4.5.6 Performance Affecting Factors

In this section, we investigate MedLeak's performance under different FL settings. We implemented the attack on the COVID CXR-4 dataset and evaluated the recovery rates and PSNR scores considering the following 4 factors: recovery bin number, client number, local training epoch, and non-iid data settings.

Recovery Bin Number: As we have discussed in Section 4.4. The recovery bin size k (i.e. neuron number of the first linear layer) can significantly affect the performance of MedLeak. In the experiment, we fixed the reconstruction batch size to 100 and changed the recovery



(a) Recovery bins factor.

(b) Client number factor.

(c) Local training epochs factor.

(d) Non-iid factor.

Figure 4.6: The recover performance of MedLeak over different practical attack factors.

bin k size from 50 to 250. We demonstrate the results in Fig. 4.6(a). We find that both the recovery rates and PSNR scores are monotonically increasing with k and obtain relatively high quantitative scores when $k \geq m$. This is consistent with our theoretical analysis as more bins (larger k) will decrease the recovery conflict probability and increase the recovery success rates. In practice, the attacker can adjust the bin size according to local sample size to obtain decent attack performance.

Client Number: We increased the client number from 5 to 25 in our experiment. The results are shown in Fig. 4.6(b). We find that the attack performance is not affected by the FL client number. This indicates that our dis-aggregation attack phase is highly successful and the attacker can always obtain accurate model updates of the single victim client.

Local Training Epoch: We increased the local training epochs of the FL clients from 1 to 9. The results are demonstrated in Fig. 4.6(c). From the results we only observe very slight performance degradation when the FL clients conduct more local training epochs, demonstrating that our attack can be applied to both FedSGD and FedAVG settings.

Non-iid Data Distribution: We changed the class of samples held by the victim clients in our experiment including having COVID-only, non-COVID-only, and both type of samples. The results are shown in Fig. 4.6(d). We observe that the attack performance is not affected by the type of samples held by the victim client, indicating that MedLeak are applicable to both iid and non-iid settings.

4.6 Discussion and Future Work

Auxiliary Dataset: Using an auxiliary dataset is a commonly used prerequisite for state-of-the-art MIAs [61, 138, 162, 213]. Our attack takes the same assumption and its performance is also affected by the number and quality of samples in the auxiliary data D_{aux}. In the ideal case, the auxiliary dataset shall have the same data distribution as the target victim's local dataset, or the auxiliary dataset is more representative. It may be a challenge for the general vision tasks to find such a representative auxiliary dataset. However, for the medical data, this does not pose a critical barrier, since the radiology data (such as the CT scans) of humans are acquired in similar data format with common anatomical features. The attacker can obtain representative datasets released for research purposes in public domains. Moreover, because the attacker is a participant in the FL system, we consider they may even collude with others to obtain this auxiliary dataset.

Defense Mechanisms: An intuitive yet effective defense against our attack is for clients to proactively verify the consistency of model parameters and architectures during the FL

training process, rather than blindly trust the privacy guarantees of FL systems and place full confidence in FL service providers—a trust that is prevalent in the current medical FL systems. However, we argue that our attack can be initiated during the initial training rounds, or even in the very first round, to maintain its stealthiness. This is because, in the early stages of training, when everything is randomly initialized, it becomes challenging for defenders to distinguish between malicious activities and benign patterns that arise from random initialization.

Another potential defense mechanism comes from the data synthesis perspective, leveraging the inherent bottleneck of MedLeak. Our comprehensive analysis reveals that the recovery bin size, denoted as k, plays a crucial role in affecting the attack performance. Specifically, when the sample size m is much larger than k, MedLeak's performance drops significantly. Based on this finding, the clients can generate large *mask sets* containing many images but not exposing anything related to the original private local samples to crowd the recovery bins. When the size of the mask set is large enough, there will be a lot of collisions within the bins, and the attacker can hardly recover anything. This defense strategy can also be employed for other attacks that face similar performance bottlenecks such as [61, 213]. However, the primary design challenge for such defense is to avoid causing any performance degradation when these mask sets are involved in the FL training process.

More than Linear Leakage: In this work, we leverage the fundamental "linear leakage" primitive as a powerful mathematical tool to help us accomplish our attack. However, we acknowledge that other types of model components such as the vision transformer [115], and convolutional layers [213] may also be exploited to reverse the model updates and leak private training data. These model components are integral to popular machine-learning models and can be exploited to launch the attack without the need to insert any additional malicious modules. This would make the attack more stealthy and adaptable for different

victim models. However, designing such an analytical gradient reverse method is non-trivial, and we intend to explore such designs in future work.

4.7 Conclusion

In this chapter, we present MedLeak–a novel MIA that targets current FL systems designed for healthcare applications using sensitive patient data. MedLeak can accurately and efficiently recover local training samples at a clinical site, resulting in unwanted leakage of private patient information. To achieve this, MedLeak requires the parameter server (i.e., the attacker) to actively craft additional adversarial attack modules before the global models. These adversarial models are designed with the mathematical guarantee to effectively break the secure aggregation protocol and efficiently recover hundreds of samples in a batch without relying on costly optimization methods when they are sent to the clients. We customize MedLeak to recover both image and textual data records, as clinical data usually comprises both types of samples, extending its applicability to wider healthcare-related FL systems. We implement MedLeak on multiple medical images and text datasets and our results highlight MedLeak's excellent attack performance under various real-world settings. Our attack exposes a practical vulnerability of the current medical FL systems, prompting the community to reconsider the privacy guarantees of these systems and to develop effective defenses against such advanced MIAs.

Chapter 5

Model Inversion Attacks against Secure Federated Learning Systems

(Copyright Notice[1])

5.1 Introduction

Federated learning (FL) is a distributed learning framework that enables its participants to collaboratively train a machine learning model without sharing their individual datasets [121]. Within this framework, the training process occurs iteratively between a central parameter server and a group of clients. During each training round, the parameter server first broadcasts a global model with a pre-agreed model architecture to all or a fraction of clients. The server then collects and aggregates the model updates (either gradient or parameter updates) submitted by the clients, which are trained and derived from their respective local datasets. As no individual training data samples are exchanged between participants in this process, FL is widely recognized as a communication-efficient and privacy-preserving learning paradigm.

[1]This chapter previously appeared as a part of a conference paper published in NDSS 2025. ©2025 Copyright held by the owner/author(s). Reprinted, with permission, from Shanghao Shi, Ning Wang, Yang Xiao, Chaoyu Zhang, Yi Shi, Y. Thomas Hou, and Wenjing Lou, "Scale-MIA: A Scalable Model Inversion Attack against Secure Federated Learning via Latent Space Reconstruction," in Network and Distributed System Security Symposium 2025 (NDSS'25), 2025 [162].

5.1.1 Privacy Leakage of Federated Learning

Unfortunately, recent research reveals that the privacy of FL is susceptible to breaches, enabling the attackers to infer information about the clients' proprietary datasets [57]. Of particular concern are the *model inversion attacks (MIAs)* [65, 74, 115, 205, 211, 216, 217], in which the adversary tries to reconstruct the original training samples from the model updates submitted by the clients. In these attacks, the parameter server is considered an *honest-but-curious attacker* whose goal is to closely approximate the input samples by minimizing the distance between real gradients uploaded by individual clients and those generated by approximated dummy samples. These attacks can successfully reconstruct high-fidelity training samples when the server gains access to *individual model updates* and undergoes sufficient optimization iterations.

To counter these attacks, Bonawitz et al. propose the *secure aggregation (SA)* protocol [27] to prevent the server from gaining knowledge about individual model updates. SA is a specialized secure multi-party computation (MPC) protocol that allows the server to compute the summation of model updates without knowing individual values. SA ensures that individual model updates are cryptographically masked and the server cannot distinguish them from *random numbers*. SA is considered one of the most robust defense mechanisms against various inference attacks targeting federated learning systems [81], and several follow-up works have been proposed to further reduce the communication and computation overhead of the original SA protocol [23, 72, 90, 199].

Despite SA's initial security guarantees, recent privacy attacks show that the SA protocol is breakable when the attacker can modify the model parameters or architectures, which *goes beyond the honest-but-curious threat model*. Two distinct attack strategies have been identified to break the SA protocol. The first strategy involves obtaining individual model

updates from the aggregated results by carefully manipulating the global model parameters and having the target client's model update dominate the aggregated results. This can be achieved by eliminating the model updates of all other clients except the target [138], or amplifying only the gradients of the target victim [190]. After obtaining individual model updates, the attacker can utilize the existing optimization-based MIAs to reconstruct input samples. However, these attacks still require costly optimization-based MIAs as part of their attack flows, limiting their applicability at scale.

The second strategy involves a more direct approach to reconstructing the input samples just from the aggregated results. To accomplish this, existing work requires the attacker to insert either a two-layer linear module [61] or a convolutional module [213] before the pre-agreed global model architecture, as well as possessing a representative auxiliary dataset. These modules are meticulously crafted or trained using the auxiliary dataset, enabling the attacker to reconstruct the inputs from the gradients of crafted layers within these modules, using customized analytical methods. However, modifying the pre-agreed model architecture is highly conspicuous and is unlikely to be accepted by the clients.

5.1.2 Our Attack

In this chapter, we present a novel model inversion attack named Scale-MIA to break the secure aggregation (SA) protocol in FL. Scale-MIA is designed to be scalable, stealthy, efficient, and highly effective, overcoming the deficiencies of existing attacks. It is capable of accurately reconstructing the clients' local data samples from the aggregated results, requires no modifications to the pre-agreed model architecture, and is more difficult to detect. Scale-MIA also eliminates the costly per batch or sample search-based optimization process. This enables the reconstruction of hundreds of data samples in parallel and results in significant

computational speed-up.

The proposed Scale-MIA involves two distinct phases—the *adversarial model generation* phase and the actual *input reconstruction* phase. The adversarial model generation can be done offline by the malicious parameter server once at the outset. The purpose is to generate an adversarial global model that follows the same architecture as the pre-agreed model architecture that will be distributed to the clients during the FL iterations. More specifically, the attacker first trains a surrogate autoencoder, with its encoder having *the same* architecture to the encoder of the real global model, and a customized generative decoder capable of reconstructing the inputs, utilizing a collected auxiliary dataset. Subsequently, the attacker feeds the auxiliary dataset to the already-trained encoder, enabling the estimate of essential statistical parameters for crafting linear layers in the adversarial global model. Finally, the adversarial global model is assembled by having its encoder identical to the surrogate autoencoder, and the following linear layers are crafted with the estimated parameters.

In the second phase, the attacker disseminates the crafted adversarial global model to clients and awaits local updates from them. Assuming the presence of the SA protocol, the attacker only receives the aggregated model updates from these clients. The proposed input reconstruction phase takes the aggregated model updates as input and aims to reconstruct as many local samples as possible. We design a novel model inversion method, in which we decompose the input reconstruction phase into two steps, both only involving efficient matrix computation and feed-forward neural network computations to reduce its complexity, making it super efficient to conduct. More specifically, we first disaggregate the received aggregated model update into batched latent space representations (LSRs) through a closed-form *linear leakage* module and then feed these representations into the pre-trained generative decoder to reconstruct the original input batch. This phase can be executed in a single federated learning round and is adaptable for launch at any desired time, such as during the FL train-

ing initialization. Our attack does not cause significant impacts on the training performance and can be launched repeatedly in multiple FL rounds to harvest as many local training samples as possible.

Three factors could significantly impact the performance of our attack—the reconstruction number is restricted by the neuron number of the first linear layer in the latent space; the reconstruction quality is sensitive to the quantity and quality of the auxiliary dataset; and the reconstruction rate relies on the availability of a good statistical estimation of the distribution of LSRs. Fortunately, in practical scenarios, all three factors do not pose a significant barrier for the attacker. Popular machine-learning models commonly feature large linear layers; auxiliary datasets can be easily collected from various online resources and public datasets; and data representations in latent space often follow a Gaussian distribution and are easy to estimate, contributing to better attack performance.

We conducted extensive experiments to evaluate the performance of the proposed attack on the Fashion MNIST (FMNIST) [195], Colorectal Histology MNIST (HMNIST) [41], CIFAR-10 [100], TinyImageNet [46], CelebA [114], and ImageNette [46] datasets. A thorough comparison was made between our attack and existing MIAs and the results show a significant improvement in terms of attack accuracy, reconstruction fidelity, and efficiency when using Scale-MIA. We also evaluated Scale-MIA's performance under different data settings, including variations in data amount and whether the data was iid or non-iid. The results consistently highlighted the success of Scale-MIA across all these settings. Notably, the results show that the attacker can successfully carry out a targeted attack using their collected dataset focused on a specific class.

5.1.3 Contributions

This chapter makes the following contributions:

1. We identify the latent space of a machine learning model as the pivotal layer for launching an MIA to breach user data privacy in federated learning systems. This insight motivates us to focus on the privacy risks posed by individual components within the model architecture, enabling us to devise a more efficient and effective attack strategy from a white-hat attacker's perspective.

2. We propose Scale-MIA, a novel MIA launched by a malicious parameter server. Scale-MIA efficiently and accurately reconstructs a large batch of user data samples from the aggregated model updates, given that the federated learning system is under the protection of a robust secure aggregation protocol.

3. Compared to existing attacks, Scale-MIA demonstrates significantly improved efficiency and scalability as it removes the requirement for expensive per-batch search-based optimization. Moreover, Scale-MIA is stealthier in its approach, as it does not require any modifications to the global model architecture and can be accomplished within one FL training round.

4. We provide a comprehensive analysis and evaluate the key factors that significantly impact the performance of Scale-MIA. Alongside our analysis, we present several practical attack scenarios of Scale-MIA to promote the need for novel defenses against such advanced attacks.

5. We conducted extensive experiments to evaluate the performance of Scale-MIA under diverse settings. We examined Scale-MIA's performance on popular model architectures including Alexnet, VGGnet, ResNet, and ViT, as well as various datasets. The

Table 5.1: Definition and notations for Chapter 5.

Symbol	Definition
c_i	Federated learning clients
n	Number of federated learning clients
G	Global model
θ	Global model parameters
t	Training round
m	Input batch size
D_i	Local datasets
g_i	Individual gradients
δ_i	Individual model updates
u_i	Masked model updates
SA	Secure Aggregation
x_i	Input samples
$\hat{x_i}$	Reconstructed samples
L	Loss function
$W^{[2]}$	Two-layer linear leakage module
D_{Adv}	Auxiliary dataset
G_{Adv}	Adversarial model
MLP	Multi-layer perception
$\hat{Enc}$	Surrogate encoder
$\hat{Dec}$	Generative decoder
$\hat{G}$	Modified global model
LSR	Latent Space Representation
CDF	Cumulative Density Function

results show the effectiveness, efficiency, and scalability of our attack.

In Table 5.1, we summarize the definitions and notations used in this chapter.

5.2 Background and Related Work

5.2.1 Federated Learning

Federated learning (FL) allows a set of clients $\mathcal{C} = \{c_1, c_2, \cdots, c_n\}$ to train a global model $G = f_\theta : \mathcal{X} \to \mathcal{Y}$ on a global dataset $\mathcal{D} = \cup_{i=1}^{n} D_i$ that is distributed along the users where

each client c_i holds a local dataset D_i without the need to share these data samples. FL is conducted iteratively in rounds until the model parameter θ converges. In each round t, the parameter server S first publishes the global model parameter θ^t to a subset of selected clients $\mathcal{C}^t \subseteq \mathcal{C}$. Then these clients compute the gradients with their local batches D_i^t as $g_i^t = \frac{1}{|D_i^t|}\nabla L(\theta_t, D_i^t)$, where $L()$ refers to the loss function. The clients send their computed updates back to S and the latter will aggregate the updates with the FedSGD algorithm [121]:

$$\theta^{t+1} = \theta^t - \eta \sum_{i:c_i \in \mathcal{C}^t} \frac{\alpha_i}{|D_i^t|} \nabla L(\theta_t, D_i^t) \tag{5.1}$$

where η is the global learning rate and α_i is the weight assigned to client c_i. The summation of all weights $\{\alpha_i\}_{i:c_i \in \mathcal{C}^t}$ is 1 and can be adjusted according to the size of local datasets D_i^t to avoid training bias. The clients can also train the received global model G_t for L_i^t local rounds before providing the model updates δ_i^t to the server. In this case, the server employs the FedAVG algorithm to conduct the training process:

$$\theta^{t+1} = \sum_{i:c_i \in \mathcal{C}^t} \alpha_i \delta_i^t \tag{5.2}$$

In the following sections, we will omit the notation t because our attack is a single-round attack and can be launched in any FL training round.

5.2.2 Gradient Inversion

The gradient inversion problem is to find a function that can reverse the individual gradient g_i uploaded by a client c_i back to the local dataset D_i under the FL setting, which can be defined as $D_i \stackrel{?}{=} Reverse(g_i, G)$.

Optimization-based Gradient Inversion: Recent research shows that a *honest-but-*

curious attacker can solve the gradient inversion problem by solving the following optimization problem:

$$\arg\min_{\hat{D}_i}[d(\nabla\hat{D}_i - \nabla D_i) + r(\hat{D}_i)] \tag{5.3}$$

where $\hat{D}_i$ refers to randomly initialized dummy samples, $d()$ refers to the distance function, and $r()$ refers to the regulation function. Zhu et al. [217] first identify this problem, and propose the deep leakage from gradient (DLG) attack which chooses the second norm as the distance function, and uses the L-BFGS optimizer [32] to solve the optimization problem. Then Zhao et al. [211] improve this attack by proposing an analytical method to recover ground-truth labels from the gradients that help DLG achieve better performance. Geiping et al. [65] further improve the optimization tool and achieve better image reconstruction fidelity, but requires a strong assumption as the labels of the inputs must be known. Yin et al. [205] focus on reconstructing batched inputs on the ImageNet dataset and ResNet model architecture, making the attack more practical. Hatamizadeh et al. [74] customize the attack for the vision transformer and achieve better performance than previous attacks.

However, these optimization-based gradient inversion attacks are computationally costly and need hundreds of optimization iterations to reconstruct one input batch [81]. Many of them (e.g., [65, 74, 211, 217]) only perform well for a single or small batch of images (typically smaller than 16), representing a scalability challenge.

Closed-form Gradient Inversion: Instead of using the costly optimization approach, some other works seek to solve the gradient inversion problem with closed-form derivation. Aono et al. [17] analyze the privacy leakage of linear models and show that the inputs to linear layers can be perfectly reconstructed from its gradients, which is referred to as the "linear leakage". This primitive is later revised and used as a component of many advanced attacks [61, 190, 212] including our work. Zhu et al. [216] propose an analytical recursive attack on privacy (R-GAP) to reconstruct the input layer by layer back from the output

layer, particularly targeting linear and convolutional neural networks (CNNs). Lu et al. [115] propose an analytical reconstruction attack specialized for the vision transformer (ViT) and can reconstruct high-fidelity images. Unfortunately, all of these analytical attacks are designed for specific model architectures and cannot be generalized for others. Furthermore, they are designed for single gradient inversion and cannot reconstruct large input batches.

5.2.3 Secure Aggregation

Previous gradient inversion attacks rely on the assumption that the attacker (i.e., the parameter server) can obtain the individual gradients $\{g_i\}$ from clients, especially for cross-device federated learning. As a countermeasure, Bonawitz et al. [27] propose the secure aggregation (SA) protocol that masks the original model updates from clients. Specifically, SA is a multi-party computation (MPC) protocol that masks the original model updates δ_i with random bits from secret sharing but keeps the summation of masked updates $\sum_{i=1}^{n} u_i$ equals to $\sum_{i=1}^{n} \delta_i$. Mathematically, this can be defined as:

$$\begin{aligned} f^{SA}(\delta_1, \delta_2, \cdots, \delta_n) &= (u_1, u_2, \cdots, u_n) \\ s.t. \sum_{i=1}^{n} u_i &= \sum_{i=1}^{n} \delta_i \end{aligned} \tag{5.4}$$

where f^{SA} refers to the abstract function of the SA protocol and the attacker cannot distinguish u_i from a random number. This implies that nothing more than the final aggregated result is leaked to the attacker. Because the final result is aggregated from all training samples submitted by all participants, the previous gradient inversion attacks cannot reconstruct meaningful information from such large input batches. The SA protocol is also communication efficiency and drop-out resilience, making it one of the most robust defense mechanisms against federated learning privacy inference attacks. The follow-up works further improve

the performance of the original SA protocol by reducing the communication and computation overheads [23, 40, 72, 90], enabling verifiable aggregation [199], and bolstering the robustness of the secure aggregation against malicious attacks [22, 31, 118, 142, 147].

Table 5.2: A comparison between different federated learning model inversion attacks.

Attack	Break Secure Aggregation?	Attacker's Capability	Attack Overhead
DLG [217], iDLG [211]	No	Weak (Curious)	Large
Inverting Grad [65]	No	Weak (Curious)	Large
GradInversion [205]	No	Weak (Curious)	Large
GradViT [74]	No	Weak (Curious)	Large
APRIL-Optim [115]	No	Weak (Curious)	Large
APRIL-Analytic [115]	No	Weak (Curious)	Small
R-GAP [216]	No	Weak (Curious)	Small
Leak in FA [49]	No	Weak (Curious)	Small
Fishing for data [190]	Yes	Medium (Modify params)	Large
Eluding SecureAgg [138]	Yes	Medium (Modify params)	Large
Robbing the fed [61]	Yes	Strong (Change architect)	Small
LOKI [213]	Yes	Strong (Change architect)	Small
Scale-MIA	Yes	Medium (Modify params)	Small

Attack	Attack Scale	Need Auxiliary Dataset?	Model Agnostic?
DLG [217], iDLG [211]	Single image	No	Yes
Inverting Grad [65]	8	No	Yes
GradInversion [205]	48	No	No (ResNet)
GradViT [74]	8	No	No (ViT)
APRIL-Optim [115]	Single-image	No	No (ViT)
APRIL-Analytic [115]	Single-image	No	No (ViT)
R-GAP [216]	Single-image	No	Yes
Leak in FA [49]	50	No	Yes
Fishing for data [190]	256	Yes	Yes
Eluding SecureAgg [138]	512	Yes	Yes
Robbing the fed [61]	1024+	Yes	Yes
LOKI [213]	1024+	Yes	Yes
Scale-MIA	1024+	Yes	Yes

5.2.4 Breaking the SA

However, when assuming the parameter server is malicious and capable of modifying the global model G 's parameters, the SA is breakable. Two general types of attacks have been identified, including the *gradient disaggregation attacks*, aiming to overturn SA's main function by inferring individual model updates δ_i from the aggregated result $\sum_{i=1}^{n} \delta_i$ with crafted model $\hat{G}$, i.e., $\delta_i = Infer(\sum_{i=1}^{n} \delta_i, \hat{G})$; and the *large batch reconstruction attack* that aims to directly reconstruct the global batch $\cup_{i=1}^{n} D_i$ from the aggregated results $\sum_{i=1}^{n} \delta_i$ with the help of additional adversarial module M_{adv} placing in front of global model G and a representative auxiliary dataset D_{aux}, i.e., $\cup_{i=1}^{n} D_i = Reverse(\sum_{i=1}^{n} \delta_i, G \oplus M_{adv}, D_{aux})$.

Gradient Disaggregation Attacks: Wen et al. [190] propose a "fishing strategy" that magnifies the gradient of a targeted class to dominate the aggregated result with crafted model parameters. The attack generates a close enough approximation of the target gradient out of the final aggregated gradient, which is enough for the attacker to reconstruct input samples through existing optimization-based gradient inversion methods. Pasquini et al. [138] propose a gradient suppression attack that zeros out all the gradient updates except the target victim's, making the final aggregated result identical to the target's gradient. The attack achieves this by crafting the parameters of a single linear layer and keeping the outputs of that specific layer always smaller than zero, which further leads to zero gradients if the ReLU activation function is used. The key limitation of this type of attack is they still use the existing optimization-based gradient inversion methods as their attack component, resulting in poor scalability performance and large computation costs.

Large Batch Reconstruction Attacks: Directly reconstructing the whole input batch from the aggregated result is a challenging task and the existing attacks [61, 212] usually have strong assumptions to accomplish it, including allowing the attacker to modify the pre-

agreed model architecture and possessing an auxiliary dataset that has a similar distribution as the training dataset. Fowl et al. [61] modify the global model architecture by attaching an adversarial two-linear-layer module in the front. The attacker can leverage the "linear leakage" primitive to perfectly reconstruct the original inputs with large batch sizes by customizing the parameters of the adversarial module, which are generated from the statistics of the auxiliary dataset. Zhao et al. [212] improve this attack by changing the linear module to a customized convolutional module. As a result, the attacker can recover large batches under a more practical FedAVG setting and can help to identify the belongings of the reconstructed samples. The key drawback of these attacks is they require the attacker to change the pre-agreed model architecture, which can be easily detected when the clients employ some integrity-checking mechanisms.

We summarize the pros and cons of existing attacks in Table 5.2. We compare the existing MIAs with our proposed attack concerning the attack assumption, overhead, and performance. Notably, our proposed Scale-MIA scales significantly better than the existing optimization-based gradient inversion attacks [65, 74, 115, 205, 211, 217] along with a small attack overhead, being model-agnostic, and the ability to launch a targeted attack against a certain class. Compared to the gradient disaggregation attacks [138, 190], Scale-MIA does not involve any per-batch optimization process and thus reduces the attack overhead (Scale-MIA can be used to replace the costly optimization-based inversion process after the individual gradient is obtained). Compared to the existing large-batch reconstruction attacks [61, 213], Scale-MIA assumes weaker attacker capability and is more stealthy and harder to detect.

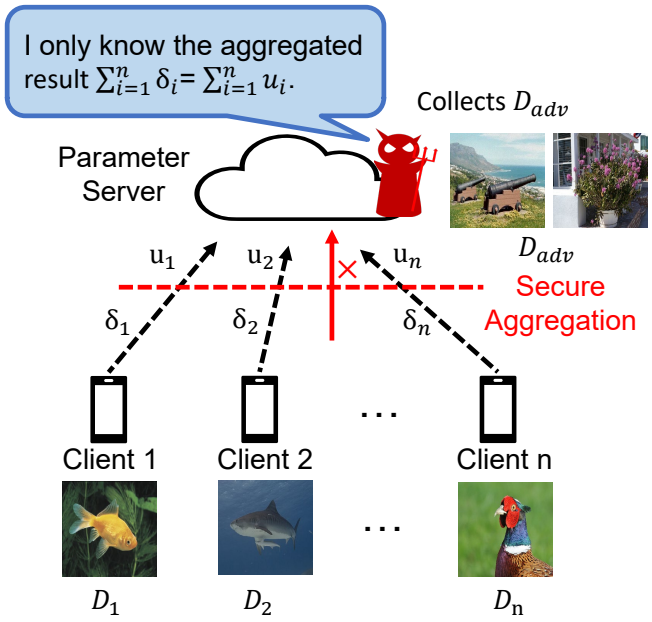


Figure 5.1: Scale-MIA threat model.

5.3 Threat Model

In this section, we formalize the attacker's capability and goal. We assume the parameter server is malicious and knows the global model G and its parameters θ of every round. We consider the state-of-the-art SA protocol (as used in [23, 27]) is in place and the attacker only sees the already masked model updates $\{u_i\}_{i=1}^{n}$ from the clients rather than the original ones $\{\delta_i\}_{i=1}^{n}$. The attacker cannot distinguish u_i from a random number but he can obtain the aggregated model update $\sum_{i=1}^{n}\delta_i$ through summing the masked inputs $\sum_{i=1}^{n}\delta_i = \sum_{i=1}^{n} u_i$. We assume the communications between the parameter server and clients are secure and no third party can alter the transmitted messages between them. We assume the attacker is able to modify the global model G's parameters but **not architecture** and knows global training configurations such as the learning rate η and weights α_i, as in [138] and [190]. We also assume the attacker possesses an auxiliary dataset D_{Adv} as a subset of all the training data D_{Train} following the same assumptions in [61, 213]. In practice, the attacker

can collect this D_{Adv} by using the existing public resources, manually collecting samples, or even colluding with a fraction of clients.

The attack aims to achieve the same goal as [61, 212], which is to recover the whole global input batch $\cup_{i=1}^{n} D_i$ efficiently and precisely, i.e. $\cup_{i=1}^{n} D_i = Reverse(\sum_{i=1}^{n} \delta_i, \hat{G}, D_{Adv})$. We illustrate our threat model in Figure 5.1.

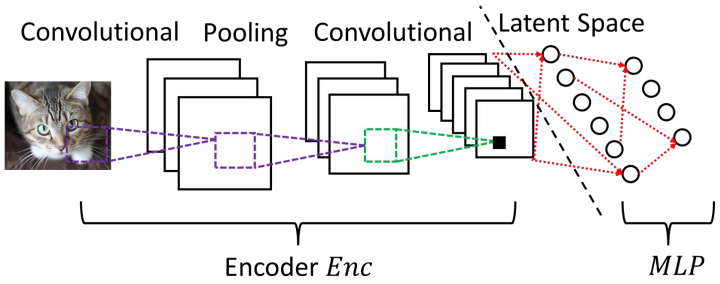


Figure 5.2: The model architecture of machine-learning classifiers.

5.4 Attack Preliminaries

5.4.1 Autoencoder

Autoencoder is an unsupervised learning technique that helps to learn an informative and compressed data representation [21, 76, 143]. The autoencoder's model architecture contains an encoder Enc and a decoder Dec. It is trained to minimize the difference between the original inputs and the reconstructed outputs, i.e., $\arg\min_{\theta} d(x, Dec(Enc(x)))$, where θ is the autoencoder's parameter vector and $d()$ refers to the error function such as the mean square error. As a result, a fine-tuned autoencoder can almost perfectly reconstruct its model inputs at the outputs, i.e. $x \approx Dec(Enc(x))$, even for the batched inputs $\cup_{j=1}^{m} x_j$. More specifically, a well-trained autoencoder consists of an encoder that encode the input samples to their latent space representations (LSRs), i.e. $\cup_{j=1}^{m} LSR_j = Enc(\cup_{j=1}^{m} x_j)$, and a

decoder that decodes the LSRs to samples, i.e. $\cup_{j=1}^{m} \hat{x_j} = Dec(\cup_{j=1}^{m} LSR_j)$ and $\hat{x_j}$ satisfies $\hat{x_j} \approx x_j$.

5.4.2 Linear Leakage

Linear leakage refers to a mathematical property that a model with two subsequent linear layers, denoted by $W^{[2]}$, with a non-linear activation function (e.g. ReLU) in between can be crafted to *perfectly reconstruct* its batched inputs $\cup_{j=1}^{m} x_j$ from the aggregated gradients of the layers $\sum_{j=1}^{m} g_j^{[2]}$ with the help of a representative auxiliary dataset D_{aux}, i.e. $\cup_{j=1}^{m} x_j = LinearLeak(\sum_{j=1}^{m} g_j^{[2]}, W^{[2]}, D_{aux})$ [61].

To describe this property, we define the linear module as:

$$\begin{aligned} y &= \delta(w_1 x + b_1) \\ z &= w_2 y + b_2 \end{aligned} \tag{5.5}$$

where $x \in \mathbb{R}^d$, $y \in \mathbb{R}^k$, $z \in \mathbb{R}^o$, $w_1 \in \mathbb{R}^{k \times d}$, $b_1 \in \mathbb{R}^k$, $w_2 \in \mathbb{R}^{o \times k}$, $b_2 \in \mathbb{R}^o$, and δ is the ReLU function. Suppose the attacker can accurately estimate the CDF of feature $h(x) = v_h.x$ of the input dataset as $\psi(h(x))$ from the auxiliary dataset D_{aux}, where $v_h \in \mathbb{R}^{1 \times d}$. Suppose the loss function is $L(x; \theta)$ or simply L where θ refers to the module parameters. The input batch size is m and can be expressed as $\cup_{i=1}^{m} x_j = [x_1, x_2, \cdots, x_m]$.

The attacker can craft the linear leakage module in the following steps: (1) Having the row vectors $w_{1(r)}, r \in \{1, 2, \cdots, k\}$ of weight matrix w_1 all identical to v_h; (2) dividing the distribution of the feature $h(x)$ into equally k bins by calculating $h_i = \psi^{-1}(\frac{i}{k})$, which results in having a random variable h the same probability to falls in each bin $[h_s, h_{s+1}]$; (3) assigning the bias vector b_1 identical to the opposite values of h vector $[-h_1, -h_2, \cdots, -h_k]$; and (4) letting the row vectors of weight matrix w_2 be identical. After this, the attacker conducts

the FL training process to obtain the aggregated gradients and then calculates the following equation to create k bins to reconstruct the input samples, for $r \in \{1, 2, \cdots, k\}$:

$$(\nabla_{w_1(r+1)}L - \nabla_{w_1(r)}L)/(\nabla_{b_1(r+1)}L - \nabla_{b_1(r)}L) \tag{5.6}$$

where specially we have $\nabla_{w_1(k+1)}L$ and $\nabla_{b_1(k+1)}L$ equal zero.

When batch size m is smaller than neuron number k, each input sample will only activate and be reconstructed by one bin. More specifically, sample x_p as the p^{th} smallest one in terms of feature $h(x)$ that falls in the l^{th} bin $[h_l, h_{l+1}]$ alone will be reconstructed by Eq. 5.6 when $r = l$:

$$\begin{aligned}
\frac{\nabla_{w_1(l+1)}L - \nabla_{w_1(l)}L}{\nabla_{b_1(l+1)}L - \nabla_{b_1(l)}L} &= \frac{\frac{\partial L}{\partial y_{l+1}}\frac{\partial y_{(l+1)}}{\partial w_{1(l+1)}} - \frac{\partial L}{\partial y_l}\frac{\partial y_{(l)}}{\partial w_{1(l)}}}{\frac{\partial L}{\partial y_{l+1}}\frac{\partial y_{(l+1)}}{\partial b_{1(l+1)}} - \frac{\partial L}{\partial y_l}\frac{\partial y_{(l)}}{\partial b_{1(l)}}} \\
&= \frac{\sum_{v=1}^{p}\frac{\partial L}{\partial y_{l+1}}x_v - \sum_{v=1}^{p-1}\frac{\partial L}{\partial y_l}x_v}{\sum_{v=1}^{p}\frac{\partial L}{\partial y_{l+1}} - \sum_{v=1}^{p-1}\frac{\partial L}{\partial y_l}} \\
&= \frac{\sum_{v=1}^{p}\frac{\partial L}{\partial y_l}x_v - \sum_{v=1}^{p-1}\frac{\partial L}{\partial y_l}x_v}{\sum_{v=1}^{p}\frac{\partial L}{\partial y_l} - \sum_{v=1}^{p-1}\frac{\partial L}{\partial y_l}} \\
&= \frac{\frac{\partial L}{\partial y_l}x_p}{\frac{\partial L}{\partial y_l}} = x_p
\end{aligned} \tag{5.7}$$

Note that we have leveraged the property of $\nabla_{w_1(l+1)}L = \frac{\partial L}{\partial y_{l+1}}\frac{\partial y_{(l+1)}}{\partial w_{1(l+1)}} = \sum_{v=1}^{p}\frac{\partial L}{\partial y_{l+1}}x_v$, and $\frac{\partial L}{\partial y_{l+1}} = \frac{\partial L}{\partial y_l}$. Their detailed mathematical proof can be found in the Appendices.

On the other hand, when batch size m is larger than neuron number k, linear leakage cannot ensure that one sample is only activated and reconstructed by one bin. In some bins, the samples collide with each other and are mixed together during the reconstruction process, leading to reconstruction failures. Therefore, we regard the neuron number k as the

performance bottleneck of the linear leakage primitive.

Linear Leakage in Federated Learning: In the federated learning system with SA, the parameter server S needs to infer the aggregated gradients $\sum_{j=1}^{m} g_j$ from the aggregated model updates $\sum_{i=1}^{n} \delta_i$ in order to launch the linear leakage attack. For the FedSGD system, the two values are identical as $\sum_{i=1}^{n} \delta_i = \sum_{j=1}^{m} g_j$. But for the FedAVG system, each model updates δ_i are trained by clients for several local rounds. Assuming the clients employ the SGD algorithm for local training, the server can only get approximated aggregated gradients $\sum_{j=1}^{m} \hat{g_j}$ from the aggregated model updates $\sum_{i=1}^{n} \delta_i$ with analytical tools, resulting in a slightly decreased linear leakage performance, i.e. $\cup_{j=1}^{m} x_j \approx LinearLeak(\sum_{i=1}^{n} \delta_i, W^{[2]}, D_{aux})$.

5.5 Attack Method

5.5.1 Attack Intuition

Linear leakage provides us with a powerful primitive to reconstruct samples. A straightforward attack strategy is to place a crafted linear leakage module $W^{[2]}$ right in front of the global model G as $G \oplus W^{[2]}$ and publish it to the clients. As a result, when the server receives the aggregated gradients $\sum_{i=1}^{n} g_i^{[2]}$ from clients, it can reconstruct the input samples with the primitive [61]. However, as we have discussed in Section 5.2, it is too suspicious and can be easily prevented by integrity checking as the attacker needs to change the global model architecture. We abandon this architectural modification approach and examine the *built-in components* of models for potential attack exploitation.

We observe that machine learning classifiers are commonly composed of a feature extraction encoder *Enc* followed by a multi-layer perceptron (MLP) in their model architectures, as

exemplified in Figure 5.2. Motivated by this, we target the *latent space* as the key layer to launch the attack based on the following reasons.

1. LSRs contain enough information to reconstruct the inputs and are widely regarded as the "information bottleneck" within the whole model architecture.

2. LSRs have relatively lower dimensions and can be processed more efficiently.

3. In most machine learning models, LSRs are followed up by MLPs, which can be exploited to launch the analytical linear leakage primitive.

Problem Decomposition Based on our previous findings, we decompose the original complex reconstruction task $\cup_{i=1}^{n} D_i = Reverse(\sum_{i=1}^{n} \delta_i, \hat{G}, D_{Adv})$ into two sub-problems, including firstly reconstruct the LSRs with the linear leakage primitive from the crafted MLPs (in terms of parameters) in the latent space, i.e. $\cup_{j=1}^{m} LSR_j = Reverse(\sum_{i=1}^{n} \delta_i, \hat{W^{[2]}}, D_{Adv})$, then reconstruct the training samples by feeding these LSRs into a fine-tuned decoder, i.e. $\cup_{j=1}^{m} x_j = Dec(\cup_{j=1}^{m} LSR_j)$, where $\cup_{j=1}^{m} x_j$ is identical to $\cup_{i=1}^{n} D_i$. Both two steps only involve matrix computation and feed-forward neural network computations, making the attack super efficient.

5.5.2 Attack Overview

We illustrate the attack flow of Scale-MIA in Figure 5.3. Scale-MIA consists of two main phases: the adversarial model generation phase (attack preparation, Steps ①-⑤) and the input reconstruction phase (Steps ⑥-⑧). The first phase is conducted locally on the parameter server S and its purpose is to generate an adversarial global model G_{adv} with crafted parameters, as well as a highly accurate generative decoder $\hat{Dec}$. This phase contains two

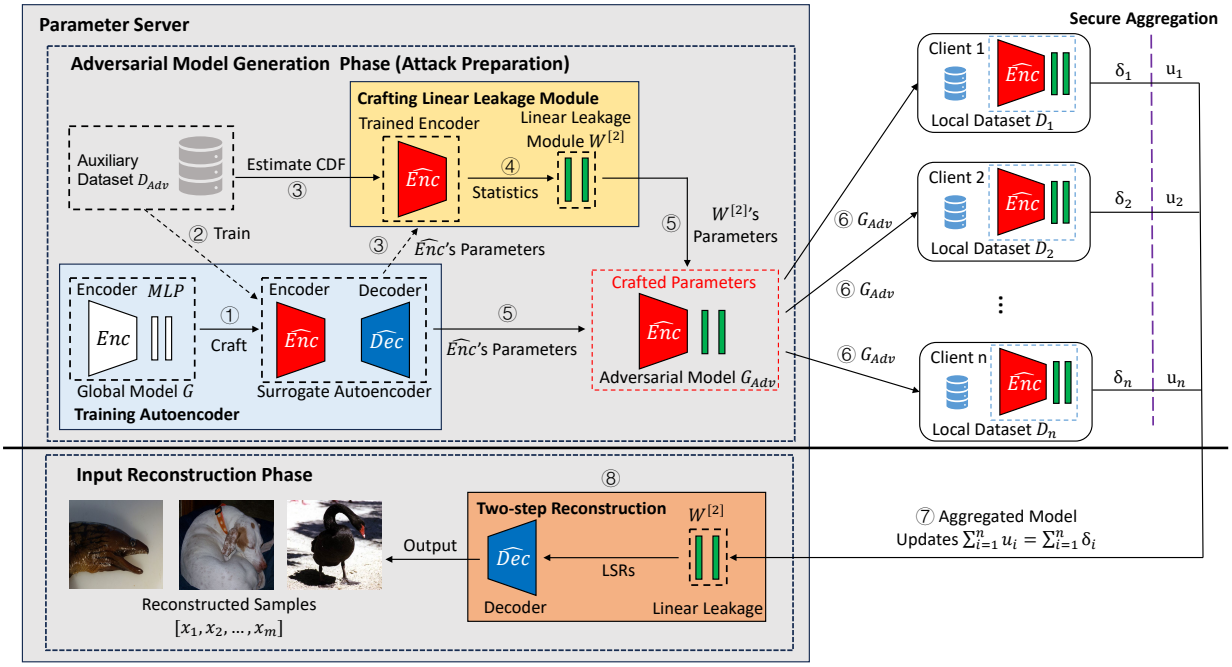


Figure 5.3: Scale-MIA is a two-phase attack. The first phase is performed locally to produce essential information to conduct the second phase. The second is the actual attack phase, during which the attacker interacts with the clients and reconstructs their local training samples.

sub-functions including training a crafted surrogate autoencoder $\hat{A}$ and crafting a linear leakage module $W^{[2]}$. $\hat{A}$ is trained with the auxiliary dataset D_{Adv} and has the same encoder architecture $\hat{Enc}$ as the global model G's encoder Enc, as well as a customized decoder $\hat{Dec}$ that has the capability to reconstruct the inputs. $W^{[2]}$ is crafted on the MLP layers of the global model G, whose necessary parameters and statistics are produced through feeding the auxiliary dataset D_{Adv} to the already-trained encoder $\hat{Enc}$. Finally, the adversarial model G_{Adv} is assembled with the surrogate encoder $\hat{Enc}$ and the linear leakage module $W^{[2]}$. By doing so the adversarial model G_{Adv} has the same architecture as the original global model G but with the necessary parameters to launch our reconstruction attack. This phase is conducted completely offline and covers all the required training efforts.

For the input reconstruction phase, the attacker first distributes the adversarial model G_{Adv}

to the clients and awaits their feedback. After receiving the feedback, the attacker examines the model updates of the MLP layers in the adversarial global model to first reconstruct the batched latent space representations (LSRs) through the linear leakage primitive, and then reconstruct the input samples by feeding the LSRs to the trained decoder $\hat{Dec}$. All the operations involved in this actual attack phase only involve linear-complexity calculations and the reconstruction is significantly accelerated.

5.5.3 Detailed Workflow

Adversarial Model Generation: We assume that the global model G has an architecture consisting of an encoder Enc followed by a multi-layer perception (MLP). This assumption is practical and commonly observed in popular image classification models such as CNN-based AlexNet, VGGNet, ResNet, and Vision Transformers. In step ①, the attacker crafts a surrogate autoencoder $\hat{A}$ consisting of an encoder $\hat{Enc}$ and a decoder $\hat{Dec}$. The encoder $\hat{Enc}$ has the same model architecture as the global model's encoder Enc and the decoder $\hat{Dec}$ is constructed according to the model architecture of $\hat{Enc}$. For example, for an encoder with several convolutional layers, the decoder can have several de-convolutional layers to reconstruct the input. In step ②, the attacker trains the surrogate autoencoder $\hat{A}$ using the auxiliary dataset D_{Adv} with the objective of minimizing the distance between its inputs and outputs. D_{Adv} can be a publicly available dataset or a custom-collected dataset targeting a specific purpose or victim. In step ③, the attacker feeds the auxiliary dataset D_{Adv} to the already trained encoder $\hat{Enc}$ to get the LSRs (LSR_{Adv}) of the dataset. The attacker estimates the CDF of the ***brightness*** (the average value among all pixels) of the LSR_{Adv} and calculates the corresponding bias vector $H = [-h_1, -h_2, \cdots, -h_k]$ of the k bins according to the "linear leakage" primitive we described. Here k is identical to the neuron number of the first MLP layer of the global model G. Then, in step ④, the attacker crafts a linear leakage

module $W^{[2]}$ on the first two MLP layers with respect to H. More specifically, the attacker has all the elements of the weight matrix of the first layer w_1 identical to $\frac{1}{d}$, where d is the dimension of the LSRs; as well as having the bias vector of the first layer b_1 equals H and the row vectors of the second weight matrix w_2 identical. Finally, the attacker generates the adversarial global model G_{Adv} by having the parameters of G_{Adv}'s encoder identical to $\hat{Enc}$ and the parameters of the MLP layers identical to $W^{[2]}$.

Input Reconstruction: After generating the adversarial model G_{Adv}, the attacker publishes it in step ⑥ to all clients. Then according to the FL framework, in step ⑦ the clients send back their model updates δ_i. Because we assume the SA is in place, the server receives the aggregated model update $\sum_{i=1}^{n} \delta_i$ instead of individual ones. In step ⑧, the reconstruction module takes the aggregated model update as the input and first uses the linear leakage primitive to recover a batch of LSRs, i.e. $[LSR_1, LSR_2, \cdots, LSR_m] \approx LinearLeak(\sum_{i=1}^{n} \delta_i, W^{[2]}, D_{Adv})$, where m is the global batch size identical to the cardinality of $\cup_{i=1}^{n} D_i$. Then these LSRs are taken as the inputs to the trained decoder $\hat{Dec}$ and the attacker finally gets $\cup_{j=1}^{m} \hat{x_j} = \hat{Dec}(\cup_{j=1}^{m} LSR_j) = \hat{Dec}(\hat{Enc}(\cup_{j=1}^{m} x_j))$ as the reconstruction outputs. According to the mathematical property of the linear leakage primitive and autoencoder, $\hat{x_j}$ and x_j are identical or highly similar.

5.5.4 Efficacy and Efficiency Analysis

Performance Bottleneck: Three main factors affect the performance of Scale-MIA. The first one is the neuron number of the first linear layer k, subject to the linear leakage's constraints. However, the neuron numbers of the first linear layers of popular machine learning models are usually very large, typically in the scale of thousands (e.g. 4096 for ImageNet classifiers), which is enough for the attacker to reconstruct hundreds or even a few

thousands of samples simultaneously. In practice, we observe that the attacker can achieve high reconstruction rates (≥ 0.7) when the batch size m is lower than $\frac{k}{2}$, i.e. $m < \frac{k}{2}$, instead of the theoretical threshold k because of imperfect estimations and noise. We observe more collided samples in different bins and gradually dropped reconstruction rates when m exceeds $\frac{k}{2}$. But we argue that both the theoretical k and practical $\frac{k}{2}$ thresholds are not hard bounds, meaning that the attacker can still reconstruct a portion of local samples even when these thresholds are exceeded, although may only with relatively low reconstruction rates (≤ 0.3).

The second factor is the quality of the auxiliary dataset D_{Adv}. Scale-MIA can achieve near-perfect reconstruction performance when D_{Adv} can represent the target training dataset D_{Train} well. In practice, the attacker can leverage various online resources including public-available datasets, image-searching tools, and even generative models such as GAN [67, 92, 137], and diffusion models [48, 77, 153] to help collect the auxiliary dataset. Note that the auxiliary samples do not necessarily need to be highly similar to the targets, all samples in the same format and class can help to improve the reconstruction performance. On the other hand, Scale-MIA enables the attacker to launch a targeted reconstruction even if the auxiliary dataset is biased, amount-deficient, and even skewed to some extent.

The third factor is whether the attacker can have a good estimation of the required statistics (i.e. the LSR distribution) to craft the linear leakage module. Fortunately, the LSRs in the latent space usually exhibit Gaussian or Laplace distribution and can be accurately estimated by the attacker. Furthermore, because the surrogate autoencoder's training process is fully controlled by the attacker, it can also use the variational autoencoder (VAE), a variant of the autoencoders to regulate the LSRs to follow Gaussian distribution [36, 95, 143]. In this way, the LSR distribution can be perfectly estimated.

Attack Overhead: The major overhead imposed by Scale-MIA is in the adversarial model generation phase, or more specifically, the training process of the autoencoder. Except this,

the other steps in phase one and the second input reconstruction phase only involve analytical operations and are efficient to perform. Scale-MIA allows this autoencoder training process to be conducted fully offline and the attacker can leverage any computation resource to fulfill this. The attacker can also resort to finding publicly available pre-trained autoencoders. Scale-MIA is a single-round attack and once the attack preparation phase is finished, the attacker can iteratively launch attacks for multiple rounds, i.e. "train once, and attack multiple rounds".

Binary Reconstruction: Scale-MIA exhibits a binary reconstruction property, meaning that for one specific sample, the reconstruction performance is either good enough to obtain a highly similar reconstructed sample or completely fails to obtain any meaningful content. This is because the major reason for reconstruction failure is the collisions within the reconstruction bins of the linear leakage primitive. The successful samples fall in one bin alone and can be properly reconstructed. But the failed ones fall in the same bin together and their reconstructed images are also mixed and blurred with each other.

Targeted Attack: Scale-MIA allows the attacker to launch targeted reconstruction with biased auxiliary dataset D_{Adv} containing limited classes of samples. For example, an auxiliary dataset full of "dog" samples can help the attacker reconstruct samples with the "dog" label among all input samples. In our experiment, we find that an auxiliary dataset with a few hundred samples in one class can reconstruct new images in the same class with high accuracy. This is particularly useful, as in many cases, the attacker may have a biased auxiliary dataset and is only curious about certain classes of samples.

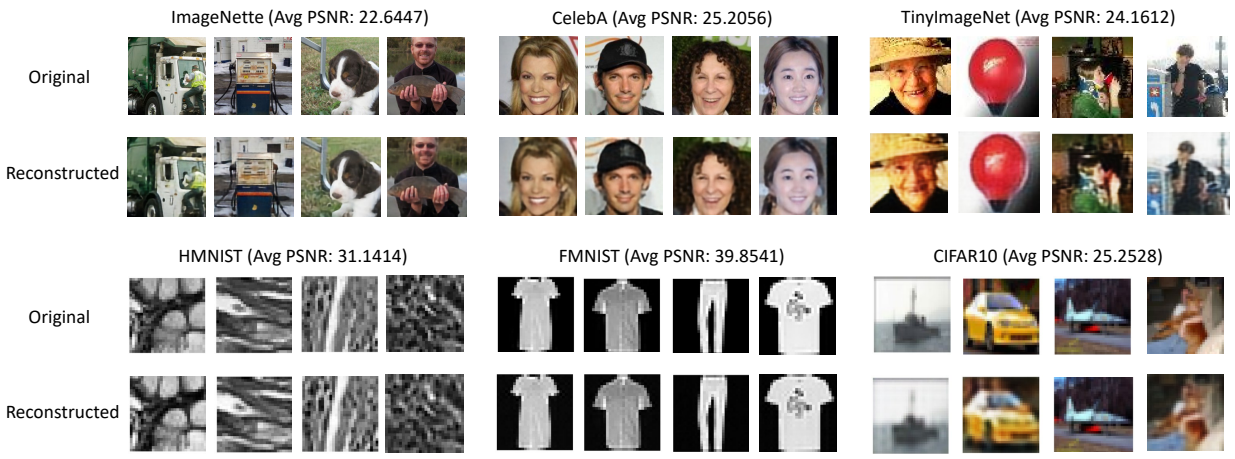


Figure 5.4: Reconstruction examples. These examples are taken from large reconstruction batches. Full reconstructed batches and more discussions can be found in the Appendices.

5.6 Evaluation

5.6.1 Experiment Settings

We implemented Scale-MIA on the PyTorch platform. We run all the experiments on a server equipped with an Intel Core i7-8700K CPU 3.70GHz×12, two GeForce RTX 2080 Ti GPUs, and Ubuntu 18.04.3 LTS.

We considered three important evaluation metrics including the reconstruction batch size, reconstruction rate, and the peak signal-to-noise ratio (PSNR) score. The batch size refers to the global batch size, i.e. the cardinality of the union of all the local datasets $|\cup_{i=1}^{n} D_i|$. This can be interpreted as the multiplication of local batch sizes and the number of clients n. For example, a global batch with 512 samples can be uploaded by 512 mobile clients with each client having one sample, by 16 clients with each having 32 samples, or just by one client with 512 samples. By default, we consider the system to contain 8 clients in one FL training round. The reconstruction rate is the ratio between the successfully reconstructed samples and the total samples. The definition of the successfulness of reconstructing a sample is

by calculating the PSNR score between the original input sample and the reconstructed one, and checking whether the score exceeds a certain threshold th. In our work, we take $th = 18$ because this threshold is enough for the attacker to distinguish the meaningful contents from the reconstructed figures clearly. The PSNR score is a widely adopted metric to quantify reconstruction quality for images and video subject to lossy compression. It can be expressed as $PSNR = 20\log_{10}(\frac{\max_I}{\sqrt{MSE}})$, where $\max_I$ refers to the maximum image pixel value and MSE refers to the mean square error. In this work, we use it to measure the performance of Scale-MIA, following the convention of the existing papers [61, 65, 190, 213].

We implemented Scale-MIA on the Fashion MNIST (FMNIST) [195], Colorectal Histology MNIST (HMNIST) [41], CIFAR-10 [100], TinyImageNet [46], CelebA [114], and ImageNette [46] datasets. Their detailed introduction can be found in the Appendices. For each dataset, we randomly selected a subset of the training set (containing {1%, 3%, 10%, and 100%} of total samples) as the auxiliary dataset and aimed to reconstruct the whole evaluation set, making sure there is **no intersection** between them. We evaluated Scale-MIA's performance on common machine learning model architectures including the convolutional neural network (CNN), AlexNet [101], VGGNet [169], ResNet [75], and ViT [50]. For each result, we repeated our experiment 5 times to eliminate uncertainty and noise. In Figure 5.4, we demonstrate several reconstructed examples over the six datasets. We plot the original images in the first row and the reconstructed ones in the second row along with the average PSNR scores. More reconstruction figures with larger batch sizes (from Figure 8.2 to 8.7) can be found in the Appendices.

Table 5.3: The comparison between Scale-MIA and the existing MIAs. * and + means that we use different thresholds other than 18. For [65] we select the threshold to be 14 and for [205] select the threshold to be 10.

Attack	Metric	1	2	4	8	16	64	256
DLG [217] /iDLG [211]	PSNR (dB)	33.0844	–	–	–	–	–	–
	Time (s)	127.1043	–	–	–	–	–	–
	Rate (ratio)	0.634	–	–	–	–	–	–
InvertingGrad [65]	PSNR (dB)	15.8407	16.2223	15.4679	14.8693	–	–	–
	Time (s)	280.126	305.6257	477.1157	866.8414	–	–	–
	Rate∗ (ratio)	0.1999	0.12	0.0833	0.0417	–	–	–
GradInversion [205]	PSNR (dB)	13.3059	12.7896	12.0550	11.1410	10.1029	–	–
	Time (s)	324.8003	341.6113	348.2077	377.3184	462.2559	–	–
	$Rate^{+}$ (ratio)	0.99	0.95	0.85	0.65	0.245	–	–
Robbing the Fed [61]	PSNR (dB)	150.568	147.9793	142.0058	134.6393	129.1871	112.3125	103.9919
	Time (s)	0.1042	0.1124	0.1137	0.1141	0.1169	0.1658	0.3192
	Rate (ratio)	0.89	0.925	0.91	0.8988	0.8981	0.8743	0.8335
Loki [213]	PSNR (dB)	-	85.8121	54.2885	43.1088	41.1882	40.4387	38.7883
	Time (s)	-	3.4335	4.4684	5.7417	8.6886	24.8163	99.3890
	Rate (ratio)	-	1.0	0.875	0.7666	0.7916	0.7344	0.7109
Scale-MIA	PSNR (dB)	29.4192	29.4162	29.3292	29.2977	29.1853	28.9188	27.2986
	Time (s)	0.01951	0.02154	0.02463	0.03060	0.04134	0.09541	0.2214
	Rate (ratio)	1.0	0.999	0.993	0.9927	0.992	0.9438	0.8464

5.6.2 Benchmark Comparison

We implemented and compared Scale-MIA with state-of-the-art model inversion attacks that have publicly available artifacts, including the DLG [217], iDLG [211], GradInversion [205], Inverting Gradient [65], robbing the fed [61], and Loki [213]. Among them, DLG/iDLG, GradInversion, and Inverting Gradient attacks are well-received optimization-based gradient inversion attacks, which are used as the fundamental building blocks of more advanced gradient disaggregation attacks including fishing for user [190], and eluding secure aggregation [138] attacks. Robbing the Fed [61] and Loki [213] represent the most recent large-batch reconstruction attacks that aim to reconstruct large input batches but at the cost of modifying model architectures. To make a fair comparison, we re-implemented all these attacks on the CIFAR-10 dataset and with the CNN model architecture. We focused on the reconstruction rate, average PSNR score, and attack time (the time to reconstruct one batch of inputs)

metrics to evaluate the effectiveness and efficiency of the attacks on reconstructing sample batches whose sizes ranged from {1, 2, 4, 8, 16, 64, 256}. We summarize the experiment results in Table 5.3.

From the experiment results, we observe that Scale-MIA outperforms all other attacks in terms of reconstruction rate and attack time while keeping decent PSNR scores. More specifically, the optimization-based attacks [65, 205, 211, 217] suffer from super long reconstruction time (hundreds of seconds), poor reconstruction rate for even small batch sizes, and low PSNR scores. We also experienced large uncertainty during our implementations of these methods because they rely on search-based optimization methods and we got completely different results with different initialization settings. Note that although the batch sizes we used in this experiment have reached or exceeded the performance upper bound of these optimization-based attacks, they have not yet reached the performance bottleneck of Scale-MIA. In comparison, Robbing the fed attack [61] achieves good reconstruction rates with large batch sizes, comparable attack efficiency (reconstruct samples in milliseconds), and even better PSNR scores compared to Scale-MIA. This is because Robbing the Fed is a closed-form attack and no optimization process is required. The other closed-form attack Loki also achieves good reconstruction rates and even better PSNR scores. However, its attack efficiency is significantly worse and requires tens of seconds when the batch size becomes large (as shown in Table 5.3). This is because it adopts a complex model crafting and reversion algorithm. Moreover, as we have discussed in Section 5.2, both Robbing the Fed and Loki attacks adopt a much stronger assumption that requires the attacker to modify the pre-defined model architecture, i.e., adding extra modules before the original model architecture, making them easy to detect. In contrast, Scale-MIA abandons this assumption and still obtains comparable or even better attack performance.

Table 5.4: The reconstruction performance of Scale-MIA over different batch sizes on FedSGD and FedAVG systems.

Systems	Datasets	64		128		256		512		1024	
		Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
FedSGD (iter=1)	CIFAR-10	0.9328	28.6453	0.8988	28.3626	0.8464	27.2986	0.7126	26.7407	0.4841	25.2302
	FMNIST	0.9402	39.4908	0.9057	38.6366	0.7882	34.2213	0.6613	33.9857	0.4173	28.2084
	HMNIST	0.9619	28.3522	0.9150	26.9998	0.8134	24.6165	0.6719	23.2551	0.4271	22.4616
	TinyImageNet	0.8906	25.0072	0.8804	23.2042	0.8136	22.9310	0.6826	22.6568	0.5252	22.3976
	ImageNette	0.8875	22.8267	0.8132	22.7786	0.7136	22.5842	0.5852	22.6251	0.4155	22.4451
	CelebA	0.9234	23.3090	0.8656	23.1878	0.7625	23.1235	0.5871	22.6476	0.3721	22.3869
FedAVG (iter=3)	CIFAR-10	0.9231	28.3490	0.8770	28.0728	0.8006	27.4762	0.6129	25.2569	0.4709	24.7438
	FMNIST	0.9687	39.2831	0.9063	38.6342	0.8068	36.7750	0.6641	32.1269	0.4146	28.3235
	HMNIST	0.9434	28.5524	0.9111	26.9872	0.7246	24.7298	0.6426	22.9255	0.5073	22.0577
	TinyImageNet	0.9070	23.2623	0.8203	23.0105	0.7488	22.8969	0.6804	22.6052	0.5210	22.2692
	ImageNette	0.8678	22.8635	0.8016	22.7005	0.7141	22.6861	0.5640	22.6489	0.3541	22.1800
	CelebA	0.9188	23.1996	0.8453	23.1373	0.7796	23.2074	0.5492	22.6717	0.3529	22.3813
FedAVG (iter=5)	CIFAR-10	0.9121	28.0774	0.8412	27.7808	0.8010	27.9652	0.6401	26.1579	0.4507	24.7552
	FMNIST	0.9046	38.7166	0.8984	37.8936	0.7421	34.0984	0.6308	34.2613	0.4355	28.8201
	HMNIST	0.9551	28.2060	0.9089	27.0392	0.7651	24.5628	0.5439	22.1618	0.4628	22.1073
	TinyImageNet	0.8917	23.2300	0.8125	23.1355	0.6804	22.6053	0.5527	22.4353	0.5261	22.4493
	ImageNette	0.8615	22.7716	0.7953	22.6870	0.7121	22.6863	0.5883	22.6171	0.3506	22.3832
	CelebA	0.9125	23.2661	0.8559	23.1304	0.7547	23.1276	0.5931	23.0004	0.3524	22.4913

5.6.3 Large Batch Recovery Performance

We evaluated Scale-MIA's performance with global batch sizes ranging from {64, 128, 256, 512, 1024} to validate Scale-MIA's performance on reconstructing large global batches under both the FedSGD and FedAVG settings over different datasets. We changed the number of local training iterations that the clients conducted ranging from {1,3,5}. We used a convolutional neural network (CNN) as the target model architecture. The experiment results are shown in Table 5.4.

We find that Scale-MIA achieves high reconstruction rates and PSNR scores and there is no obvious performance pitfall when the global batch size is smaller than 512. We observe that both the reconstruction rates and PSNR scores are monotonically decreasing with respect to larger batch sizes, which is consistent with our theoretical results, as larger reconstruction batches increase the probability of reconstruction collisions and failures. We also find that when clients conduct more local training iterations, the attack performance slightly

Table 5.5: The reconstruction performance of Scale-MIA over different models and batch sizes.

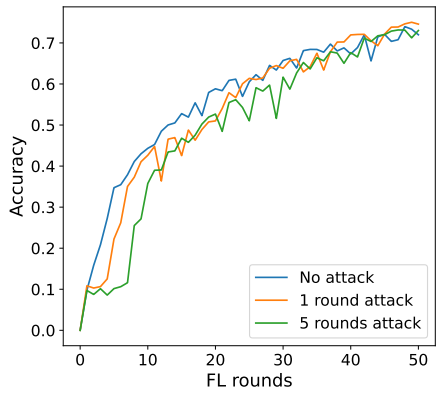
Models	64		128		256		512		1024	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
Alexnet (512)	0.9146	29.0523	0.8383	28.8489	0.7308	28.5446	0.4974	27.2409	–	–
Resnet (512)	0.9162	27.9498	0.8408	27.6572	0.7136	26.7230	0.4869	24.1208	–	–
ViT (512)	0.7656	21.3008	0.6641	20.9629	0.5391	21.0911	0.3809	20.8525	–	–
CNN (1024)	0.9328	28.6453	0.8988	28.3626	0.8464	27.2986	0.7126	26.7407	0.4841	25.2302
VGG (1024)	0.9572	28.2689	0.9191	28.2333	0.8463	27.7959	0.7301	26.9002	0.5032	25.0889

decreases. This is because, with more local iterations, the accuracy of the approximated aggregated gradients from the aggregated model updates decreases. But in general, under all assumptions, Scale-MIA achieves decent attack performance and is not affected by whether the system employs the FedSGD or FedAVG algorithms.

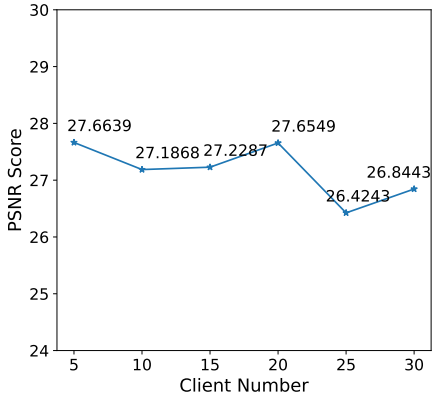
We also evaluated Scale-MIA's performance concerning different model architectures under the FedSGD setting on the CIFAR-10 dataset. The results are shown in Table 5.5. We include the neuron number of the first linear layer k beside the model architectures and find that the reconstruction rate is largely affected by it. The models with larger k have better reconstruction rates for a fixed batch size except for a few outliers. We observe that Scale-MIA achieves good reconstruction rates when the batch sizes are smaller than half of the neuron number k, in line with our analysis. In general, Scale-MIA achieves decent attack performance on most model architectures, with only achieving a slightly worse attack performance on the ViT, demonstrating that our attack can be applied to different architectures and is model agnostic.

5.6.4 Performance over Different FL Settings

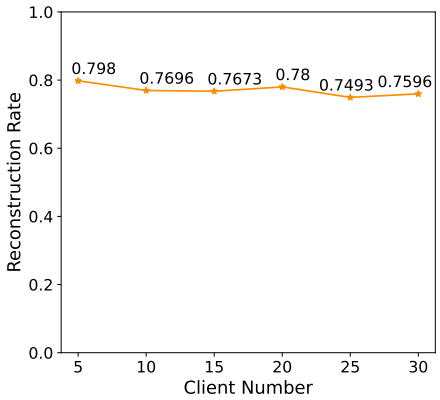
In Figure. 5.5, we demonstrate Scale-MIA's performance under different federated learning settings. We first investigated the overall training accuracy of the FL system with and without our attack. We trained the FL system with 8 clients using the CNN classifier on the



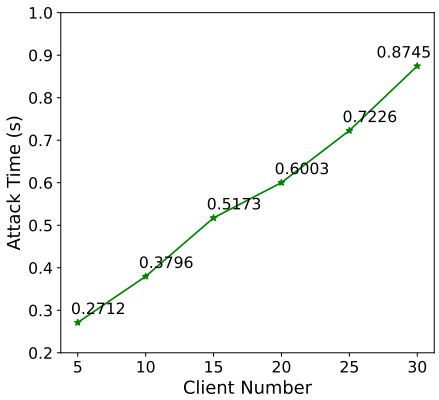
(a) FL training performance with and without Scale-MIA attack.



(b) Reconstruction PSNR scores over different FL client numbers.



(c) Reconstruction rates over different FL client numbers.



(d) Attack time (s) over different FL client numbers.

Figure 5.5: Scale-MIA's attack performance over different federated learning settings.

CIFAR-10 dataset for 50 rounds. We consider two different attack scenarios, including the attacker only launching a single-round attack in the first training round, and the attacker continuously launching attacks in the first 5 training rounds. From the results, we find that the convergence speed of the FL training process becomes slower in the initial training rounds when the attacks are launched, but the final training accuracy is not affected. In our experiment, the trained model obtained 0.7298 accuracy without any attack, 0.7354 accuracy under a single-round attack, and 0.7301 accuracy under a 5-round attack.

We also investigated Scale-MIA's performance over different numbers of FL clients. We

implemented our attack on the CIFAR-10 dataset and fixed the reconstruction batch size to 300. We increased the FL client number from 5 to 30, which was {5, 10, 15, 20, 25, 30}, and evaluated the PSNR scores, reconstruction rates, and elapsed time (in seconds). From the results, we find that the PSNR score and reconstruction rate are slightly affected by the FL client number and remain at a decent high level. This indicates that the reconstruction performance is not affected and the reconstructed images are of good quality. However, we observe that the attack time is almost linear increasing with respect to the FL client number. This is reasonable because more clients involve more communication and computation overhead within the FL system and the complexity of our inversion attack increases accordingly.

Table 5.6: The reconstruction performance of Scale-MIA over different amounts of data.

Batch	1% (500 samples)		3% (1500 samples)		10% (5000 samples)		100% (50000 samples)	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
16	0.9808	25.1174	0.9824	25.4396	0.9809	27.9021	0.9827	29.4134
32	0.9537	24.9192	0.9569	25.2514	0.9550	27.7217	0.9586	29.1968
64	0.9090	24.8604	0.9136	25.1489	0.9143	27.6032	0.9146	29.0523
128	0.8243	24.6688	0.8316	24.9584	0.8317	27.3481	0.8313	28.8489
256	0.6873	24.3534	0.6923	24.6209	0.7018	26.8835	0.7308	27.2409

Table 5.7: The reconstruction performance of Scale-MIA over different numbers of classes of data.

Batch	1 class		2 classes		3 classes		10 classes	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
16	0.9818	25.1320	0.9856	24.8238	0.9753	25.2728	0.9827	29.4134
32	0.9557	25.0041	0.9701	24.8306	0.9505	25.2147	0.9586	29.1968
64	0.8984	24.8337	0.9128	24.7875	0.8971	25.1607	0.9146	29.0523
128	0.8307	24.5928	0.8568	24.5584	0.8229	25.0155	0.8313	28.8489
256	0.6914	24.1795	0.7214	24.0819	0.6615	24.5275	0.7308	27.2409

5.6.5 Data Deficiency and Bias

The quality of the auxiliary dataset D_{Adv} is an important factor that impacts the performance of Scale-MIA. The ideal situation is that D_{Adv} has the same distribution and can represent the training dataset well. However, in practice, there is usually data deficiency and bias and we evaluate the impacts of them in this section.

Data Deficiency: We varied the amount of data available to the attacker for launching Scale-MIA and evaluated its performance on the CIFAR-10 dataset using the AlexNet model. We considered scenarios where the attacker possesses different proportions of the total training data in the auxiliary dataset, ranging from 1%, 3%, 10%, to 100% (equivalent to 500, 1500, 5000, to 50,000 images). The attack performance was tested on the entire validation set, which consists of 10,000 images. Given that the first linear layer of the AlexNet model has 512 neurons, we varied the reconstruction batch size from 16 to 256 to ensure a reasonable reconstruction performance.

We demonstrate the attack performance in table 5.6. We observe that the number of available samples has little impact on the reconstruction rate, as it remains relatively stable at high values across the full range of data availability. The PSNR score does decrease slightly when the number of available samples decreases, but it remains in a decent range. Specifically, even if the attacker only has 1% samples (500 images in total and 50 images for each class) of the total training dataset, Scale-MIA still achieves very high reconstruction rates and PSNR scores in our experiment. This indicates that Scale-MIA is a practical attack and the attacker only needs to collect or generate a few hundred samples to obtain a decent attack performance.

Data Bias: We assumed the auxiliary dataset D_{Adv} to have different numbers of data classes from the CIFAR-10 dataset to evaluate Scale-MIA's performance over biased data.

We considered the attacker to have {1,2,3,10} classes of data samples and evaluated Scale-MIA's performance on these particular classes. For example, if we had the attacker possess the "dog" samples, we would only evaluate Scale-MIA's performance on reconstructing the "dog" samples in the validation set, ignoring the samples from other classes. We focused on the PSNR score and reconstruction rate for batch sizes ranging from 16 to 256, following the same setting as our previous experiments.

Table 5.7 presents the targeted attack's performance. The results show that data bias has a very limited impact on the reconstruction rate, as it remains stable even when the attacker has only a few classes of samples. The decrease in PSNR score is also not significant, and even the worst value remains at a relatively high level. These results demonstrate that Scale-MIA's attack performance is robust against data bias. The attacker can successfully launch a targeted attack on specific classes with minimal performance degradation.

Table 5.8: The reconstruction performance of Scale-MIA over data skew.

Training Data	Testing Data 1	Testing Data 2
		

Batch	Intra-class Skew		Inter-class Skew	
	Rate	PSNR	Rate	PSNR
16	0.9851	23.2206	0.7025	20.2722
32	0.9750	23.1560	0.6625	20.1722
64	0.9194	22.4435	0.6468	19.7398
128	0.8463	22.2957	0.6057	19.6457
256	0.7929	22.1004	0.4678	19.4764

Data Skew: We considered the auxiliary dataset D_{Adv} to contain skewed data from the target images. We considered two data skew cases including intra-class skew and inter-class skew. We conducted our experiment on the TinyImageNet dataset and evaluated the attack

reconstruction rates and PSNR scores. We considered the auxiliary dataset to contain 500 "monarch butterfly" images. For the intra-class skew, we assumed the target images were 500 "sulfur butterfly" images. For the inter-class skew, we assumed the targets were 500 "frog" images.

In Table 5.8, we demonstrate the attack performance over different data skew settings. We find that the attack performance slightly decreases (but remains decent) under intra-class skew settings, while the attack performance significantly drops under inter-class skew settings. This is because autoencoders can only reconstruct images similar to training samples by design. The results indicate that Scale-MIA can overcome intra-class skew well, but still faces gaps in dealing with inter-class skew.

Table 5.9: The reconstruction performance of Scale-MIA under DP protection.

Budget/Batch	64		128		256		512	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
No DP	0.9328	28.6453	0.8988	28.3626	0.8464	27.2986	0.7126	26.7407
$(1, 10^{-5})$	0.9253	25.2141	0.9079	25.1279	0.8445	24.9721	0.6855	24.6703
$(1, 10^{-4})$	0.9140	25.1581	0.9019	25.1392	0.8542	25.0477	0.6934	24.7081
$(5, 10^{-5})$	0.9194	25.1359	0.9034	25.1417	0.8359	25.1688	0.6933	24.5526

5.6.6 Differential Privacy

Differential privacy (DP) [51, 52, 53] has been widely used to protect the training data's privacy in machine learning systems [6, 26, 87, 171]. It has shown its effectiveness in protecting client-level and data record-level membership privacy for FL systems by preventing the attacker from knowing whether one item (either a client or a record) exists in the system. The fundamental idea of DP is to add artificial noise to the model updates before they are sent to the parameter server. Though DP can protect the FL systems against membership inference attacks by its definition, it is demonstrated to be less effective against the model

inversion attacks [125].

In this section, we consider the FL system is protected by both the DP and SA protocols, and we examine Scale-MIA's performance on it to check whether our attack can still break them. We assume the clients adopt the DP-SGD algorithm [6] during the local training process with different (ϵ, δ) privacy budgets. In our experiment, we implement DP with the open-source Python-based DP library named Opacus [207]. We demonstrate the results in Table 5.9. From the results, we find that the reconstruction rate is slightly affected by the DP mechanism and remains stable at high levels under different privacy budgets. The PSNR scores decrease slightly when DP is employed but not significantly. This shows that Scale-MIA can still reconstruct samples with high accuracy and good scalability performance even when the DP is in place. However, Scale-MIA cannot link the reconstructed samples back to its clients (membership inference), showing that DP can still preserve a certain level of privacy.

5.7 Discussion

Privacy by Shuffling: Scale-MIA allows a malicious parameter server to reconstruct the whole input batch accurately from the aggregated model updates, demonstrating a serious privacy vulnerability of the SA protocol and federated learning system. However, under the current design, the attacker cannot infer the belongings of these reconstructed samples and attribute them to certain clients. This property is known as "privacy by shuffling", which prevents the attacker from conducting membership inference and shows that the SA protocol can still preserve a certain level of privacy. To further break this privacy guarantee, Scale-MIA can be used in conjunction with the gradient disaggregation attacks [138, 190] by launching these attacks in the first step to obtain the individual model updates from

the victims and then using Scale-MIA as a replacement of the existing gradient inversion mechanisms to boost the reconstruction performance.

Two-Linear-Layer Limitation: Scale-MIA requires the model to have two consecutive linear layers to craft the linear leakage module. However, we acknowledge that not all machine learning models necessarily have this component in their architectures although most machine learning classifiers contain it. For example, some Resnet-based and ViT-based models only have one linear layer in the latent space and the attacker must add extra linear layers to launch Scale-MIA. Meanwhile, we observe that some non-linear modules in the feature extraction layers such as the convolutional layers and vision transformers may also leak private information analytically, which can help the attacker bypass the two-linear-layer limitation. We leave this as our future work to investigate.

Attack Stealthiness: Scale-MIA is a single-round attack that can be executed at any stage of FL training. To enhance attack stealthiness while maintaining effective performance, the attacker may choose to launch the attack during the initial or the first training rounds. In these rounds, the model parameters can be initialized with arbitrary patterns, making it challenging for defenders to distinguish between adversarial and benign parameters. From a performance perspective, the parameter server receives relatively large aggregated gradients during these initial rounds, as the global model has not yet converged, which facilitates more accurate linear leakage calculations and improves the model inversion performance.

Potential Countermeasure: The data synthesis method could be a potential countermeasure against our novel attack. The fundamental idea is to have each client generate a *mask set* that hides the original sensitive data samples. These mask sets ensure that the mask samples within them rather than the original local samples owned by individual clients are reconstructed during the reconstruction attack. At the same time, the mask sets shall not affect the federated learning training performance. We consider the guided diffusion model

a promising tool for generating such mask sets. But we leave the detailed technical design as the future work.

5.8 Conclusion

In this chapter, we propose Scale-MIA, a powerful MIA that breaks the strong SA protocol by reconstructing the whole global batch possessed by the clients efficiently from the already masked and aggregated model updates. Scale-MIA launches the inversion attack from a new perspective by delving into the detailed architecture of the global model and decomposing the complex model inversion problem into two steps: an LSR reconstruction step and an input generation step, based on the observation that the latent space is the critical layer to breach the privacy. Scale-MIA uses a closed-form "linear leakage" primitive to conduct the first step and a fine-tuned generative decoder for the second, making it highly efficient and suitable for large-scale reconstruction. Scale-MIA is also a very stealthy attack as it does not modify the model architecture and can be conducted in any FL training round. With these distinct features, Scale-MIA represents a potent and inconspicuous approach to breach privacy in FL systems, prompting the need for more robust defense mechanisms against such advanced attacks.

Chapter 6

Adversarial Perturbations against Multimodal Diffusion Models

(Copyright Notice[1])

6.1 Introduction

Diffusion models such as Stable Diffusion [153], DALL-E [145], Kandinsky [19], and Midjourney [102] have gained widespread popularity due to their capability of generating high-quality synthesis images based on given input prompts. Until now, they have been extensively used by a broad range of applications such as digital arts, data augmentation, video gaming, and animations. It is estimated that more than 15 billion AI-based images have been created by the diffusion models up to August 2023 [182]. Popular platforms such as Stable Diffusion have been reported to have more than 10 million users up to January 2024 [172].

Diffusion models were initially introduced as text-based generative models. However, many semantic concepts are difficult to express accurately in natural languages alone, often requiring significant user efforts to customize effective prompts. To mitigate this, the most

[1]This chapter was in submission to a conference for possible publication as of December 2025. Reprinted/adapted, with permission, from Shanghao Shi, Chaoyu Zhang, Heng Jin, Y. Thomas Hou, and Wenjing Lou, "SWAP: Misleading Multimodal Diffusion Models via Contrastive Learning-based Adversarial Perturbations," May 2025.

recent diffusion models are adopting multi-modal input prompts, including images, text, and audio, to enhance the users' ability to guide and control the generation process.

However, different input modalities cannot be processed together directly due to their distinct data formats and shapes. Particularly, the image and text modalities have completely different formats – the images are in continuous pixel space, while the text words are in discrete space. To bridge them together, multimodal adapters are introduced to transform all modalities into uniform representations in a shared representation space [73, 107, 108, 203]. The multimodal adapter typically consists of multiple multimodal encoders followed by feature integration layers. Among them, the multimodal encoders convert inputs into the same format, and the feature integration layers fuse all features from different modalities together into unified representations. For example, the image-prompt (IP) adapter [203] consists of a Contrastive Language–Image Pre-training (CLIP) image encoder [58], a CLIP text encoder, followed by a linear transformation layer, and cross-attention layers.

With the integration of multimodal adapters, state-of-the-art diffusion models such as Stable Diffusion [172] and Kandinsky [19] have demonstrated impressive multimodal generation capabilities. However, it remains unclear whether these new adapters will introduce new security vulnerabilities, particularly in the context of adversarial attacks. The adversarial attack represents a fundamental security vulnerability of machine learning models. By definition, it requires the attacker to slightly modify the model inputs with imperceptible perturbations, but misleading the model into making completely wrong and even attacker-chosen predictions [33, 68, 119]. With respect to the multimodal diffusion models, the adversarial attacks aim to mislead the model from generating the attacker-chosen content. Of particular interest, the sensitive not-safe-for-work (NSFW) content, such as adultery, violent, and disgusting images, is a major safety concern. Therefore, in this work, we investigate whether the multimodal diffusion models can be triggered to generate NSFW content. We consider

the existence of *safety filters* within the multimodal diffusion models to detect and block such NSFW content, and our proposed attack aims to bypass these protection mechanisms.

Existing Attacks: Classic adversarial examples have been shown to be ineffective against diffusion models. In contrast, many existing works are using the diffusion models as a defense mechanism to purify the adversarial perturbed images back to clean images [106, 131, 184]. This robustness originates from the step-by-step denoising process of the diffusion model, which effectively breaks the customized adversarial perturbations. On the other hand, diffusion models are used to generate adversarial examples that mislead machine learning classifiers based on their powerful generation capability [37, 39]. However, these adversarial examples are confined to classifiers and are only effective for the classification task. For the multi-modality encoder/adapter's attacks, existing works focus on misleading the multimodal encoders from making wrong predictions [214] or generating adversarial embeddings [20]. However, they are confined to generating insensitive content and cannot jailbreak the safety filters. These attacks are also not tailored for diffusion models and warrant further investigation to assess their applicability in this context. For the jailbreaking attacks, existing works focus on compromising the text-to-image (T2I) diffusion model [117, 179, 201]. Their attack method is to selectively modify a few words within the text prompt to mislead the diffusion model from generating NSFW content. But little attention has been paid to the multimodal diffusion models, particularly the adversarial attacks launched through other modalities such as images. Compared to text, images can be crafted with greater flexibility and stealth because pixel values are in an unconstrained continuous space, while words are discrete values confined in vocabularies. It is worth a thorough investigation whether the image modality can be leveraged to mislead the diffusion models.

Our work: In this chapter, we propose SWAP, a novel adversarial attack that misleads the multimodal diffusion models from generating the attacker-chosen content, including NSFW

images. SWAP exploits image as the attack modality to introduce subtle and stealthy perturbations. SWAP is a targeted attack and enables flexible target description through text descriptions, images, or even both. SWAP identifies the multimodal adapter as the critical attack component and aims to compromise the intermediate representations. This can naturally lead to the generation of the targeted content, leveraging the strong generative capability of the vanilla diffusion model, but without getting it involved in the attack process.

Technically, SWAP focuses on finding a small perturbation in the input pixel space that steers the encoded representation of the original input toward the near neighborhood of the attacker-chosen target, while keeping distances from a list of sensitive NSFW concepts enforced by the safety filters. To achieve this, we formalize an exponential contrastive loss that takes the target representation as the "positive sample" and the NSFW representations as "negative samples". This can increase similarity between the adversarial and the target's representations, while enforcing the adversarial and the NSFW representations far apart according to the definition of contrastive learning [93, 176, 177, 194]. We leverage the gradient descent-based optimization method to optimize the input perturbation within a small box constraint iteratively. We set the perturbation threshold to be very low so that the perturbed images are visually indistinguishable from the original images. Note that our optimization process only involves the multimodal adapter (ignore the complex diffusion part), which significantly alleviates the computation overhead. In summary, the final perturbed image will trigger the multimodal adapter to generate malicious representations close to the targets, which further lead to the generation of the attacker-desired content based on the vanilla diffusion model's excellent generation capability but without being detected by safety filters. Our method can be further combined with the existing query-based adversarial attack [14] to enhance its jailbreaking capability, boosting the attack success rate to a high level with only a few queries.

Through extensive experiments, we demonstrate that the attacker can successfully mislead the current state-of-the-art diffusion models, including Stable Diffusion 1.5, and Stable Diffusion XL models. We select normal and NSFW images from the ImageNet [46] and NSFW Classification [43] datasets as the attack targets, performing both targeted generation and jailbreaking tasks. The experiment results demonstrate that SWAP achieves high attack success rates, decent CLIP scores, and consumes approximately 50 seconds to generate one adversarial example, which all significantly outperform the current works. SWAP can effectively bypass the two models' safety filters to generate the attacker-desired NSFW images. We provide visual examples for SWAP, and the results clearly demonstrate that the adversarial perturbations are indistinguishable by human perception, while the adversarially generated images are totally different and closely follow the attack targets.

In summary, this chapter makes the following contributions:

1. We explore the multimodal adapter as the critical attack surface for compromising multimodal diffusion models and launch our attack through the image modality, which is a more effective and stealthier attack vector largely overlooked in existing literature.

2. We propose a novel adversarial attack that misleads the multimodal diffusion model from generating the attacker-chosen content. Our method does not involve the complex diffusion component, and significantly reduces the attack overhead.

3. Our proposed attack guides the multimodal diffusion model to generate attacker-specified content while maintaining a safe distance from sensitive NSFW concepts through contrastive learning. This approach effectively bypasses the built-in safety filters of diffusion models while preserving high semantic similarity between the attacker's intent and the generated output.

4. We implement our attack on state-of-the-art multimodal diffusion models, including

Stable Diffusion 1.5, and Stable Diffusion XL. SWAP achieves excellent performance on the two models, validating the effectiveness of our attack.

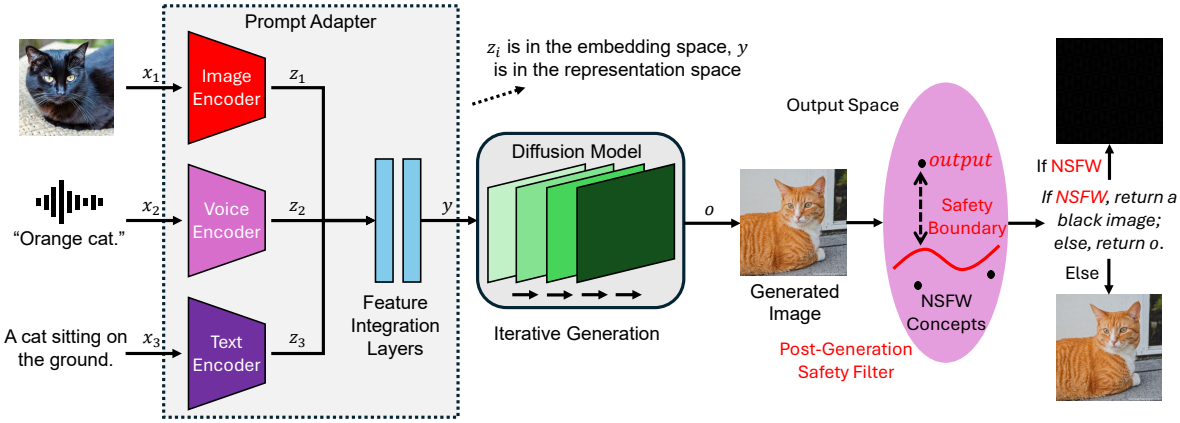


Figure 6.1: Multi-modal diffusion model system model.

6.2 Background and Related Work

6.2.1 Multimodal Diffusion Model

Diffusion models are a type of generative model that produces user-desired content guided by input prompts. It supports the generation of diverse types of content, including images [172], music [80], and videos [78]. Upon initial introduction, diffusion models were text-to-image generative models, accepting only text prompts [19, 102, 145, 172]. However, text-only descriptions often require significant user effort to craft effective prompts for high-quality content generation and sometimes may still fail to convey the intended concepts. To address this limitation, recent diffusion models have begun adopting multimodal input prompts such as images, audio, or a combination of them to enhance the user's control and guidance over the generation process. These prompts can range from textual descriptions to visual or auditory inputs, offering a more flexible and expressive user interface.

Multimodal Adapter: On the other hand, bridging the gap between different modalities is a non-trivial challenge because different modalities have completely different data representation formats. For example, images are in the continuous pixel space, but text is in the discrete word space. To handle this, the multimodal adapter is introduced [108, 203]. It consists of two parts, including the multimodal encoders $\mathcal{E}$ and feature integration layers $\mathcal{F}$. In Fig. 6.1, we illustrate the workflow of the multimodal adapters. It serves as an independent component of the well-trained vanilla diffusion model $\mathcal{G}$. We denote the multimodal diffusion model as $\mathcal{M}$ and the inputs as $x = [x_1, x_2, x_3]$, where x_i belongs to different modalities. In the beginning, the inputs $x = [x_1, x_2, x_3]$ are first converted into latent embeddings $[z_z, z_2, z_3]$ via modality-specific encoders $\mathcal{E}_i$, i.e., $z_i = \mathcal{E}_i(x_i)$. Note that all the modality-specific encoders are jointly trained and bonded together to ensure that different latent embeddings z_i are in *the same* embedding space. Later on, the three embeddings $[z_1, z_2, z_3]$ are taken into the feature integration layers $\mathcal{F}$ to be transformed and concatenated into uniform representations y, i.e., $y = \mathcal{F}(z_1, z_2, z_3)$, serving as the intermediate input to the vanilla diffusion model $\mathcal{G}$. Subsequently, the vanilla diffusion model $\mathcal{G}$ will process the representations y to generate the model output o. We underscore that there are three data spaces, including the input space, the embedding space Z, and the representation space Y, with each one tied to the corresponding data processing stages.

Multimodal Encoder: The multi-modality encoder $\mathcal{E}$ is a vital pre-trained component of the prompt adapter. It bridges the gap between different input modalities and maps them into the same embedding space Z. The key feature of the multimodal encoders is that they can map semantically similar inputs close to each other in the embedding space, regardless of their modality, providing excellent quantitative measurements of semantic similarities. We underline that the multimodal encoder $\mathcal{E}$ usually refers to *a set of* bonded modality-specific encoders $\mathcal{E}_i$ to process each input modality separately.

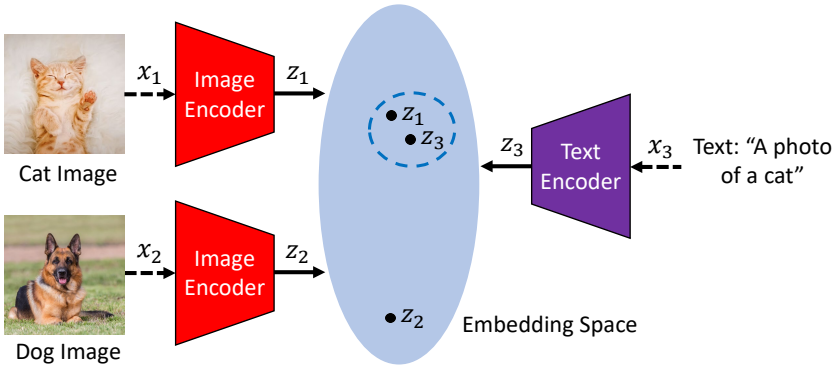


Figure 6.2: CLIP image-text encoder.

Over the years, various multi-modality encoders have been developed. The groundbreaking CLIP image-text encoder [144] is designed to bridge the image and text modalities. More recent AudioCLIP [73] and IMAGEBIND [66] encoders enhance their applicability to more modalities, including audio, image, and text. The SigLIP [208] encoder confines its scope within the image-text encoding, but improves the encoding performance via optimizing the contrastive learning loss. In this work, we focus on the CLIP encoder because it has been widely used in multi-modality applications, particularly for the multimodal diffusion models [153, 203].

CLIP Image-Text Encoder: The CLIP encoder, which stands for contrastive language-image pre-training, is a joint image and text encoding model that is trained with a tremendous number of image-text pairs in a self-supervised way [144]. It maps semantically similar images and text into close embeddings in the latent embedding space. In Fig. 6.2, we demonstrate an example of how the CLIP image and text encoders work. The cat image and the semantically identical text description "an image of a cat" have very close embeddings in the embedding space, while the dog image has an embedding far away from them.

CLIP encoder is designed upon vision transformers but has many variants regarding its model architecture, input sizes, and parameter numbers. Popular fine-tuned CLIP encoders

are publicly available on the Hugging Face website [58] with their model details specified, enabling convenient plug-and-play usage for them. Besides the CLIP encoder, many other multi-modality encoders, such as SigLIP and IMAGEBIND, are also well-trained and publicly available on the Hugging Face website. Therefore, we consider it trivial for a developer or attacker to find detailed information about a multi-modality encoder, or at least a surrogate model accomplishing identical functions.

Safety Filter: State-of-the-art diffusion models are employing safety filters to prevent the generation of NSFW content. According to [146], Stable Diffusion models employ a post-generation image checker to detect and filter the potential NSFW content. As we have demonstrated in Fig. 6.1, the post-generation safety filter compares the semantic similarity between the generated image and a pre-defined list of NSFW concepts, such as "nude" and "violent". When *any of* the semantic similarity scores between them is higher than a threshold, the generated image will be flagged as NSFW content, and Stable Diffusion will return a totally black image instead. There are also pre-generation safety checkers to guard the input text prompts. This method will check whether the text prompts contain certain bad words in a pre-defined list. Once these bad words are found, the corresponding text prompts are flagged as malicious and automatically get rejected.

6.2.2 Related Work

Since the initial launch, multimodal diffusion models have faced a variety of security breaches. Because the multimodal diffusion models consist of two major components, including the multimodal encoder/adapter and vanilla diffusion model, we classify the attacks into two types – the multimodal attacks and diffusion model attacks.

Multimodal Attacks: Zhou et al. propose a downstream agnostic adversarial attack

against the CLIP encoder, aiming to mislead the encoder from performing incorrect zero-shot predictions [214]. However, this attack is confined to the classification task and cannot be applied to the diffusion model's generation task. Zhang et al. introduce an adversarial attack against the multimodal embedding models, focusing on generating malicious embeddings [20]. This work shows that adding a small perturbation in one modality (e.g., image) can adversarially align the embeddings of another modality (e.g., audio), which can further affect various downstream tasks. However, the limitation of this attack is that it must be launched across two modalities. In addition, this work does not consider jailbreaking safety filters and cannot be used for NSFW content. Shayegani et al. propose to launch the attack through adversarially modifying multiple modalities, such as crafting both text and image prompts [159]. This can effectively jailbreak the safety filters and affect the downstream tasks. However, this attack does not set constraints on the crafted images, and they can be largely different from the original ones.

Diffusion Model Attacks: Zhuang et al. propose a query-free adversarial attack against Stable Diffusion [218]. This attack demonstrates that a one-word adversarial modification of input text prompts is able to mislead the text-to-image (T2I) diffusion model from generating completely different content. But this work is still confined to T2I rather than multimodal diffusion models. This attack also does not consider generating the NSFW content. [117, 179, 201] is a series of jailbreaking attacks that compromise the T2I diffusion model to generate NSFW content. They focus on adding or changing a few words to launch the adversarial attack, still confined in the text space. For our work, we focus on launching adversarial attacks from the pixel space instead of the text space, which is stealthier, more flexible to process, and introduces less computation overhead.

Summary and Comparison: We summarize the difference between our attack and state-of-the-art existing attacks in Tab. 6.1. Particularly, our attack is customized for multimodal

Table 6.1: A comparison between our work and existing multimodal/diffusion model attacks.

Attack	Target System	Attack Capability	Attacker's Knowledge
AdvCLIP [214]	CLIP Encoder	Misclassification	White-Box
AdvIllusion [20]	Multimodal Embed	Targeted Generation	White-Box
Jailbreak-in-Pieces [159]	Vision Language	Targeted Generation	Grey-Box
Query-free Attack [218]	T2I Diffusion	Targeted Generation	Grey-Box
Sneaky Prompt [201]	T2I Diffusion	Targeted Generation	Grey-Box
Ring-a-bell [179]	T2I Diffusion	Targeted Generation	Grey-Box
JPA Attack [117]	T2I Diffusion	Targeted Generation	Grey-Box
SWAP	Multimodal Diffusion	Targeted Generation	Grey-Box
Attack	Jailbreaking?	Attack Modality	Perturbation
AdvCLIP [214]	No	Image	Small
AdvIllusion [20]	No	Cross-Modality	Small
Jailbreak-in-Pieces [159]	Yes	Image	Large
Query-free Attack [218]	No	Text	Medium
Sneaky Prompt [201]	Yes	Text	Medium
Ring-a-bell [179]	Yes	Text	Medium
JPA Attack [117]	Yes	Text	Medium
SWAP	Yes	Image	Small

diffusion models. Our attack can achieve targeted generation, jailbreak safety filters, by adding small adversarial perturbations. In the table, "Grey-box" means that the attacker has white-box access to the adapters/encoders but no knowledge of the downstream applications. "Medium" perturbation refers to word-level text replacement, which is observable by human perception.

6.3 Threat Model

Attacker's Capability: We consider the attacker to be a malicious user of the multimodal diffusion model, as demonstrated in Fig. 6.3. He cannot interfere with any internal process of the multimodal diffusion model, but can send adversarially crafted inputs to mislead it. We

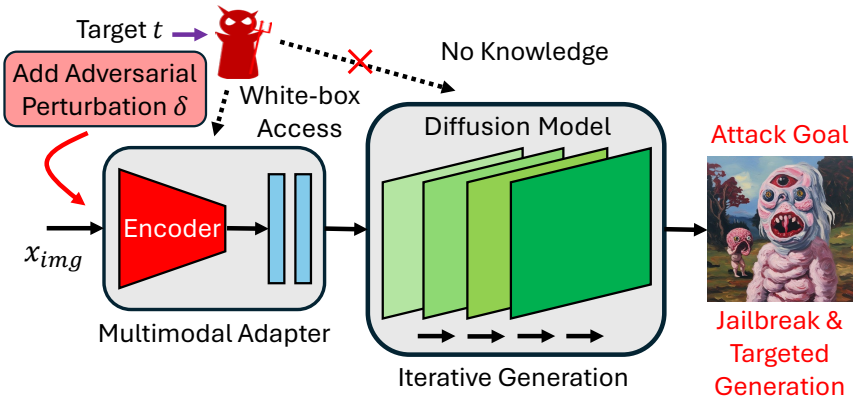


Figure 6.3: SWAP threat model.

consider that the attacker targets the image modality to add a small and unnoticeable adversarial perturbation to the original images. This perturbation is mathematically bounded by small norm values, i.e. $\|x - x_{adv}\|_p < \epsilon$ where x and x_{adv} are original and adversarial images. This can ensure that the adversarial images are visually similar to the original ones.

Attacker's Knowledge: We assume the attacker has zero knowledge about the vanilla diffusion model $\mathcal{G}$, including its inner architecture, model parameters, and its built-in safety filters. However, we assume that the attacker has *white-box access* to the multimodal adapter. This is achievable in practice because state-of-the-art multimodal adapters used by commercial diffusion models (e.g., the image prompt adapter for Stable Diffusion) are publicly available on the Hugging Face community [59]. If such information is not available, we assume the attacker can get access to a surrogate adapter achieving similar encoding functions.

Attacker's Goal: The goal of the attacker is to mislead the multimodal diffusion model from generating the attacker-chosen content. The attacker can describe the attack target through text descriptions, image prompts, or both. The generated content will closely follow the attack target and can even be NSFW content, such as nude and violent images. The attacker's goal is to bypass the built-in safety filters for the NSFW content.

6.4 Attack Method

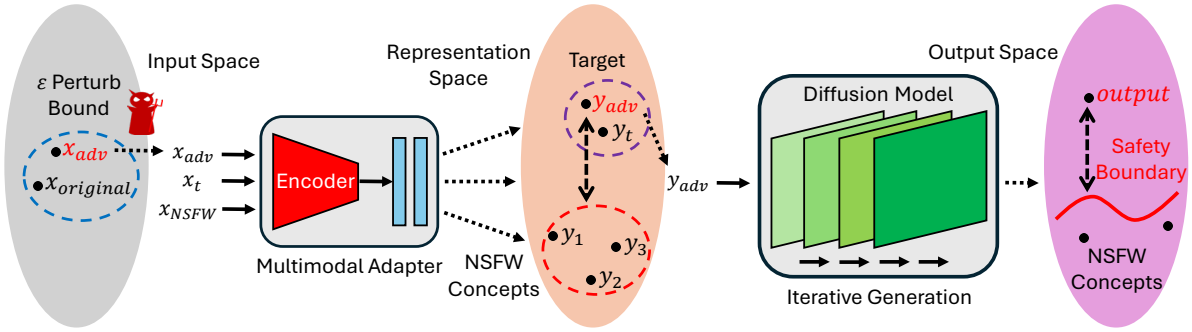


Figure 6.4: SWAP attack flow – misleading the adversarial representation close to the target, but keeping distances from the NSFW concepts. These relative semantic relationships will be preserved till the output space.

6.4.1 Design Intuition

The multimodal diffusion model is a sequential model that consists of a multimodal adapter and a vanilla diffusion model. Among the two parts, the vanilla diffusion model is much more complex because it involves a recursive generation process containing hundreds of steps. The key function of the vanilla diffusion model is to generate high-quality content that closely follows its input representations. Therefore, to reduce the attack overhead, we focus on attacking the first component, i.e., the multimodal adapter, to mislead it from generating the attacker-desired representations. In this way, the following vanilla diffusion model will be naturally misled to generate the attacker-desired content, but without getting the complex diffusion part involved in the attack process.

In Fig. 6.4, we demonstrate the fundamental attack flow of SWAP. We formalize finding the adversarial image x_{adv} in the input space bounded by the perturbation threshold ϵ as an optimization problem. The optimization object is to search within the ϵ input perturbation space to find a point that satisfies: (1) close enough to the target's representations;

and (2) keeping distances from the sensitive NSFW representations (e.g., "nude", "naked", "violent") as much as possible. In this way, the generated adversarial representation can maintain semantic similarity with the target while staying away from bad concepts. Both relative semantic relationships will be well preserved throughout the vanilla diffusion models' generation process till the output, resulting in the jailbreaking of the post-generation safety filter.

6.4.2 Attack Methodology

Contrastive Loss: Contrastive loss is a loss function commonly used in representation learning tasks. It can pull the similar pairs closer in the representation space, and push dissimilar pairs apart. This is consistent with our fundamental attack logic. We take the adversarial image/target as a similar pair, and the adversarial image/NSFW concepts as dissimilar pairs. Compared to classic contrastive learning, our method aims to learn a perturbed adversarial image instead of model parameters. In our case, the multimodal adapter's parameters are frozen, and the adversarial image is continuously optimized.

Suppose the adversarial example is $x_{adv} = x + \delta$, where $\|\delta\|_p < \epsilon$; the target is x_t; and the NSFW concepts are $[x^1_{NSFW}, x^2_{NSFW}, \cdots, x^m_{NSFW}]$, where m is the number of the NSFW concepts. Suppose their corresponding representations after the multimodal adapter $\mathcal{A}$ are y_{adv}, y_t, and $[y^1_{NSFW}, y^2_{NSFW}, \cdots, y^m_{NSFW}]$. The contrastive loss can be expressed as:

$$\mathcal{L}_{contras} = -\cos(y_{adv}, y_t) + \lambda \sum_{j=1}^{m} \cos(y_{adv}, y^j_{NSFW}) \tag{6.1}$$

where $\lambda \in (0, 1]$ is a training hyperparameter that balances the two components of the loss function.

To optimize $\mathcal{L}_{contras}$, we iteratively update x_{adv} with the projected gradient descent (PGD) method [119]. We use the Adam optimizer [94] and clipped gradients to fulfill the optimization process. Detailed training steps are shown in Algorithm 4.

Query-based Jailbreaking: Our method can push the adversarial representations away from the NSFW concepts. However, when the target itself contains too much sensitive content, we observe that the adversarial representation y_{adv} may still near the safety filter's decision boundary. To address this, we propose to add a second query-based jailbreaking perturbation δ^+ in addition to the first one. This perturbation will be added in the orthogonal space of the first perturbation space, i.e., $\epsilon_p^{\perp}$, meaning that δ^+ will only be added to the unmodified dimensions to keep the overall perturbation small. The object to add δ^+ is to effectively flip over the safety filter's decision boundary with limited queries because x_{adv} has already been optimized to a good sub-optimal point.

Technically, we employ the *Square Attack* random search method [14], which adds randomly generated δ^+ in a continuous square area within the whole image, i.e., $\delta^+ = rand_{\epsilon_p^{\perp}}(\hat{H}, \hat{W})$, where $\hat{H} \in H, \hat{W} \in W$ are sub-dimensions of the whole image. In each query round r, the attacker will recursively add this random perturbation to evade the safety filters:

$$x_{adv}^{r+1} = Clip(x_{adv}^{r} + rand_{\epsilon_p^{\perp}}(\hat{H}, \hat{W})) \tag{6.2}$$

The *Square Attack* paper [14] has provided rigorous analysis for the attack efficiency, effectiveness, and convergence. In our experiment, we further validate that a few queries are enough to jailbreak the built-in safety filter of Stable Diffusion models.

Algorithm 4 SWAP-Attack

Input: The multimodal adapter $\mathcal{A}$, training epochs e, learning rate η, input image prompt x, attack target x_t, query round r, multimodal diffusion model $\mathcal{M}$, and a list of NSFW concepts $[x^1_{NSFW}, x^2_{NSFW}, \cdots, x^m_{NSFW}]$.

Output: The adversarial perturbation δ, query-based perturbation δ^+, and perturbed image $x_{adv} = x + \delta + \delta^+$.

function ADVERSARIAL PERTURBATION GENERATION
  $\delta = rand(\epsilon_p), x_{adv} = x + \delta$
  $y_{adv} = \mathcal{A}(x_{adv}), y_t = \mathcal{A}(x_t), y^j_{NSFW} = \mathcal{A}(x^j_{NSFW})$
  for i `in range(`e`)` **do**
    $\mathcal{L} = \mathcal{L}_{contras}$
    $g = \nabla\mathcal{L}$
    $\delta = \delta - \eta g$
    $\delta = Clip(\delta, \epsilon)$
  end for
end function
function QUERY-BASED JAILBREAKING
  $\epsilon_p^{\perp} \cap \epsilon_p = \emptyset$
  while $\mathcal{M}(x_{adv})$ `is blocked` **do**
    $x^{r+1}_{adv} = Clip(x^r_{adv} + rand_{\epsilon_p^{\perp}}(\hat{H}, \hat{W}))$
  end while
end function

6.4.3 Analysis

We provide additional insights about SWAP by systematically addressing the following questions.

1. *Why SWAP can bypass the safety filters?* Ans: Our method pushes the adversarial representations away from NSFW concepts. As long as the representation space effectively captures semantic concepts, these adversarial representations will remain distant from NSFW regions in other well-structured classification spaces (e.g., those used by safety filters), due to the inter-space transferability of learned semantics.

2. *There are so many NSFW concepts, how to select the proper ones for SWAP attack?* Ans: We adopt the reverse-engineered NSFW concepts in [146] that define the safety

filter of Stable Diffusion, including 17 unsafe concepts and 3 special care concepts. The full list of these concepts can be found in [146] and in the Appendices.

3. *What if the multimodal adapter is unknown?* Ans: The multimodal adapter may not always be available to the attacker. However, there are many state-of-the-art multimodal encoders/adapters available on the Hugging Face website, including the image-prompt (IP) adapter/encoder used by Stable Diffusion. The attacker can use these models as surrogate models to launch the attack. Because the adversarial examples have attack transferability, the attacker can still obtain decent attack performance under some surrogate models. We will discuss this in detail in Section 6.5.

6.5 Implementation

6.5.1 Experiment Settings

We implemented SWAP on the PyTorch platform. We run all the experiments on a server equipped with an AMD EPYC 7643 48-Core CPU Processor, four GeForce NVIDIA A100-SXM4-80GB GPUs, and Ubuntu 22.04.3 LTS.

Data Setting: We use the initial image from the Hugging Face image-to-image generation task, as shown in Fig. 6.5, as the common source image x across all attack scenarios. For the non-NSFW attack targets, we select 100 images from the ImageNet dataset [46] as visual targets. For the NSFW content, we choose 100 samples from the NSFW Classification dataset on the Hugging Face website containing adultery and offensive images [43]. Its detailed introduction can be found on the website. We also use 100 concise textual descriptions tied to the 100 NSFW images (e.g., "a photo of a smiling naked woman at home") as the textual targets.

Multimodal Diffusion Model: We select the state-of-the-art Stable Diffusion 1.5 and Stable Diffusion XL [172] as our attack models. They have employed the image-prompt (IP) adapter [203] to accept both image and text as input prompts. Both models have built-in post-generation safety filters and will return a totally black image when the generated image is detected to contain potential NSFW content.

Evaluation Metric: We use three evaluation metrics, including the CLIP score, the attack successful rate (ASR), and attack time, to evaluate the performance of our attack. The CLIP score is computed as the cosine similarity between the CLIP embeddings of the attack target x_t and the adversarially generated content o_{adv}, i.e., $s_{CLIP} = cos(\mathcal{E}_{CLIP}(x_t), \mathcal{E}_{CLIP}(o_{adv}))$. The value is between 0 to 1, and a larger value indicates better semantic similarity. This metric is widely used to evaluate the semantic similarity between two images [20, 201]. We use it to identify whether the generated content follows the attack target. For the non-NSFW content, the ASR is calculated as the ratio of the attack target/generated image pairs that have CLIP scores larger than 0.7. For the NSFW content, the ASR is calculated as the ratio of pairs that not only obtain the CLIP scores larger than 0.7, but also bypass Stable Diffusion's built-in safety filters. We set the maximum query budget for jailbreaking the safety filters to 50. An attack is considered unsuccessful if its budget is exhausted. For the attack time, we focus on the optimization time of the adversarial image x_{adv}.

6.5.2 Experiment Results

Targeted Generation Performance: In Fig. 6.5, we demonstrate the targeted generation examples for Stable Diffusion XL. In all four cases, we select the same original image, but with different types of targets, including cat/dog images, and two concise text descriptions related to an orange cat and a seal. We plot the corresponding adversarial images and four

generated outputs. We can find that the adversarial images are visually indistinguishable from the original images, but can successfully mislead Stable Diffusion XL from generating images closely following the attack targets.

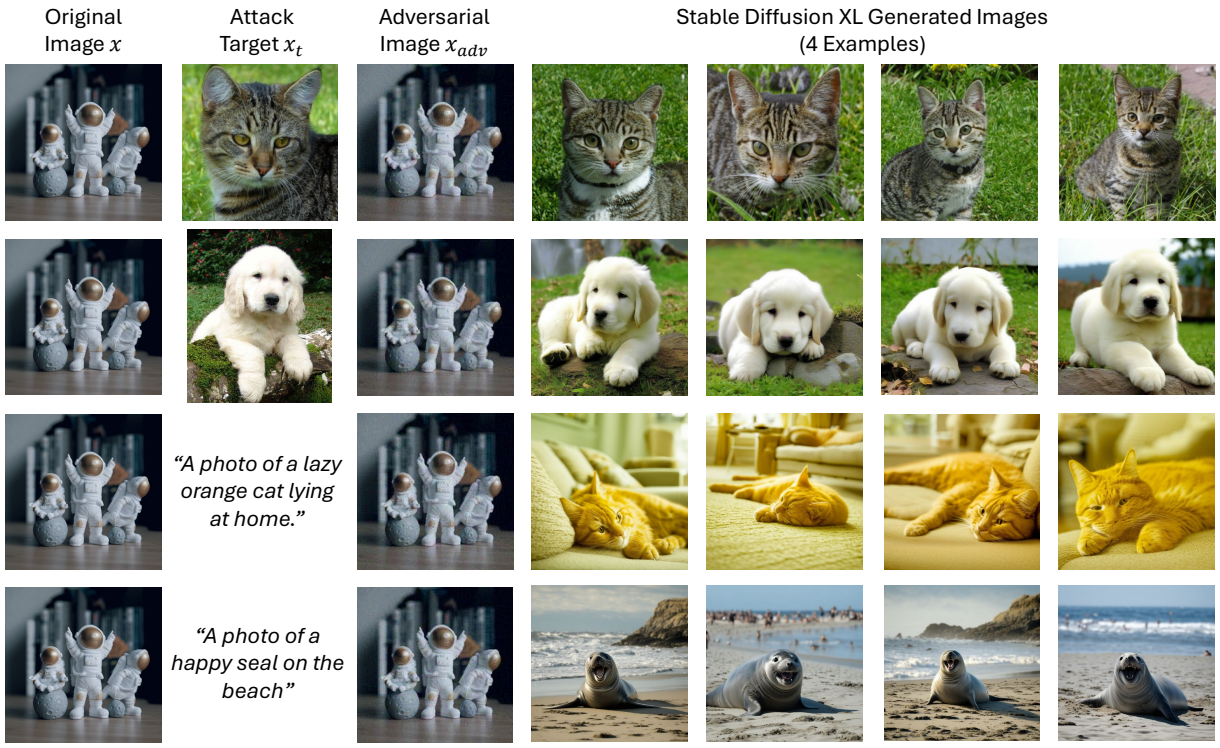


Figure 6.5: Targeted generation examples for Stable Diffusion XL. The image perturbation bound ϵ is $\frac{8}{255}$.

In Tab. 6.2, we demonstrate the numerical performance of our attack on the targeted generation task over the 100 ImageNet samples. We change the image perturbation bound ϵ from $\frac{4}{255}$ to $\frac{12}{255}$. From the results, we observe that both ASR and CLIP scores increase when the perturbation bound ϵ increases for both Stable Diffusion XL and Stable Diffusion 1.5 models. This is because when the perturbation bound becomes loose, the attacker has a larger search space for an adversarial image, and is more likely to find a better sub-optimal point. Under the same setting, SWAP achieves better ASR and CLIP scores on Stable Diffusion XL than Stable Diffusion 1.5. This is because Stable Diffusion XL has

better generation capability and can produce images with higher quality/CLIP scores under similar inputs. We also observe that compared to Stable Diffusion XL, Stable Diffusion 1.5 requires less attack time to generate an adversarial image. This is because Stable Diffusion 1.5 adapts a simpler multimodal adapter, therefore, requiring less computation cost than Stable Diffusion XL.

Table 6.2: SWAP targeted generation attack performance.

Model	Perturb Bound ϵ	ASR	CLIP Score	Attack Time (s)
SDXL	4/255	0.81	0.8183	43.19
	8/255	0.98	0.8536	44.33
	12/255	0.99	0.8670	44.34
SD1.5	4/255	0.69	0.8130	39.19
	8/255	0.93	0.8219	39.47
	12/255	0.94	0.8272	39.71

NSFW Generation Performance: In Fig. 6.6, we demonstrate the NSFW generation examples for Stable Diffusion XL. In all 4 cases, we select adultery and violent images, as well as offensive descriptions, as the attack targets. We plot the corresponding adversarial images and 4 Stable Diffusion XL-generated samples. Again, we find out that the adversarial images are visually indistinguishable from the original images, but all 4 generated images are closely related to the attack target and all bypass the safety filters. For ethical issues, we have added masks to hide the inappropriate parts of these images. Note that for our attack, we don't have any knowledge about the safety filter used by Stable Diffusion XL.

In Tab. 6.3, we demonstrate the numerical performance of our attack on the NSFW Classification dataset. Same as the previous task, we change the perturbation bound ϵ from $\frac{4}{255}$ to $\frac{12}{255}$. We add the number of queries needed to bypass the safety filter as an additional evaluation metric. From the results, we observe that in general, SWAP can bypass the safety filters to generate NSFW content with decent ASRs, high CLIP scores, and with only *a few*

queries. Particularly, for Stable Diffusion XL, our attack can break the safety filters without requiring additional queries in most cases, validating the effectiveness of our stage-one perturbation generation method. We observe that SWAP achieves better attack performance on Stable Diffusion XL than Stable Diffusion 1.5 in terms of ASR, CLIP score, and average queries. This is again because of the better content generation capability of Stable Diffusion XL. Similar to the first task, the attack achieves better performance with larger perturbation bounds ϵ, due to the increased search space.

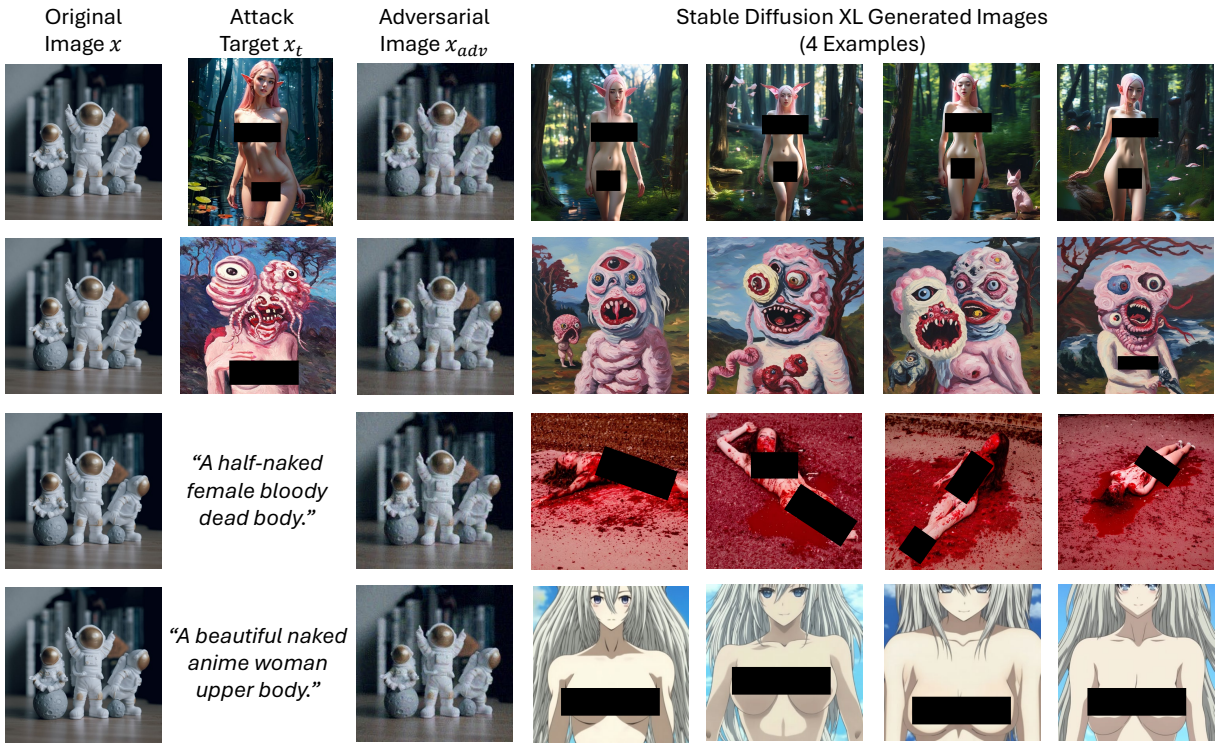


Figure 6.6: NSFW generation examples for Stable Diffusion XL. The image perturbation bound ϵ is $\frac{8}{255}$. We have added masks to hide the inappropriate parts within the images.

Baseline Comparison: We compare the performance of our attack with two baseline attacks, including Sneakyprompt [SP'24] [201] and Adv-illusion [Security'24] [20]. The first work is a jailbreaking attack against the T2I diffusion model, and the second work is a multimodal attack against the multimodal adapters, but without jailbreaking capability. To

Table 6.3: SWAP NSFW generation attack performance.

Model	Perturb Bound ϵ	ASR	CLIP Score	Average Queries	Attack Time (s)
SDXL	4/255	0.54	0.7687	1.25	43.51
	8/255	0.68	0.7937	1.14	43.76
	12/255	0.70	0.8160	1.04	43.88
SD1.5	4/255	0.42	0.7690	4.08	49.05
	8/255	0.64	0.7898	3.80	48.43
	12/255	0.66	0.7915	2.94	47.45

make fair comparisons, we select 100 NSFW concise text descriptions based on the NSFW Classification dataset [43] as the attack target to accommodate for the Sneakyprompt attack. We also enhance the Adv-illusion with the same query-based jailbreaking method in our work [14]. We select Stable Diffusion 1.5 as the target system, and set an upper bound query limitation to be 50 to avoid infinite search.

Table 6.4: SWAP baseline comparison performance.

Attack	Perturb Bound ϵ	ASR	CLIP Score	Average Queries	Attack Time (s)
Sneaky	3 words	0.20*	0.2647	31.88	106.46
Adv-illu	8/255	0.0	-	50	162.38
SWAP	8/255	0.64	0.7898	3.80	48.43

In Tab. 6.4, we demonstrate the attack performance of the three attacks. From the results, we find that our attack outperforms the other two attacks on all evaluation metrics. Particularly, we find that the Adv-illusion attack fails to jailbreak the safety filters within 50 queries and achieves a 0.0 ASR. This is reasonable because this attack only minimizes the distance between the input and the attack target's representations, while failing to keep distances from the NSFW concepts. Unfortunately, this makes the adversarial images too close to the attack target (NSFW content), and easily detectable by the safety filters. For the Sneakyprompt attack, we find that this attack achieves a much lower ASR, even under its

own CLIP attack successful boundary – 0.26 (ours is 0.7). This attack also achieves lower CLIP scores, and uses approximately 8× queries than our work. It is worth noting that our attack requires an average of only 3.8 queries to bypass the safety filters, making it practical and highly efficient. For the attacker overhead, our attack consumes significantly less time than the other two attacks.

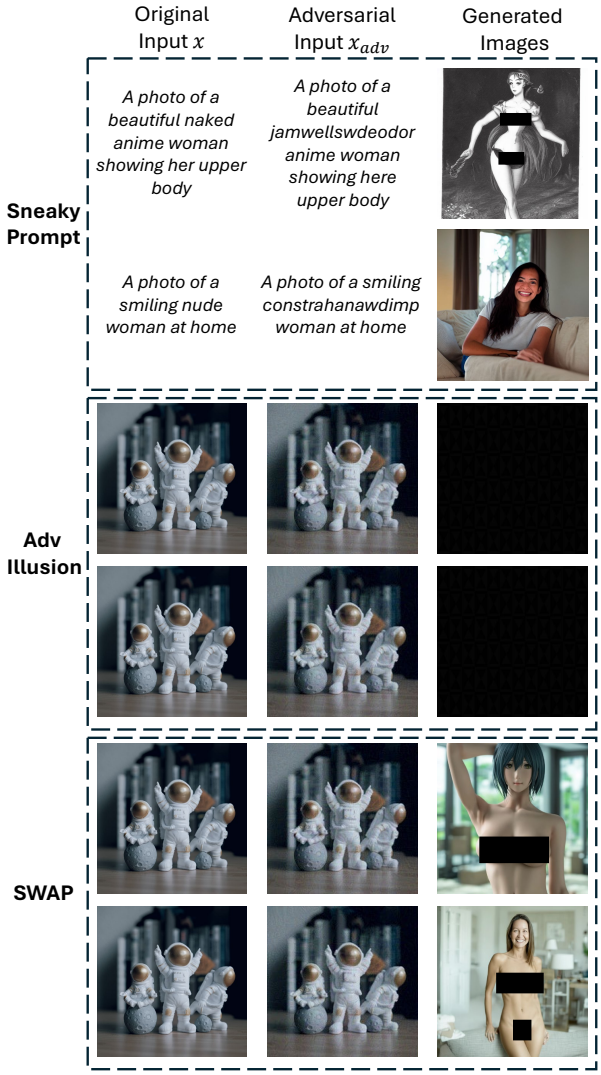


Figure 6.7: NSFW generation examples of the three attacks on Stable Diffusion 1.5. The black images mean that the corresponding attack fails to jailbreak the built-in safety filter, and black images are returned instead.

In Fig. 6.7, we demonstrate a comparison of the generation results between the three attacks over the same NSFW attack targets, including drawing pictures about "*A photo of a beautiful naked anime woman showing her upper body*", and "*A photo of a smiling nude woman at home*". From the results, we find that Sneakyprompt successfully jailbreaks Stable Diffusion 1.5 on the first attack prompt, but fails to break the second one. Sneakyprompt does generate an image based on the second attack prompt, but the image does not contain any desired NSFW content. For the Adv-illusion attack, it fails to break the built-in safety filter of Stable Diffusion 1.5 within the required number of queries. As a result, two black images are returned by the diffusion model. For our SWAP attack, it achieves excellent jailbreaking and generation performance to produce two NSFW images that closely follow the attack prompts.

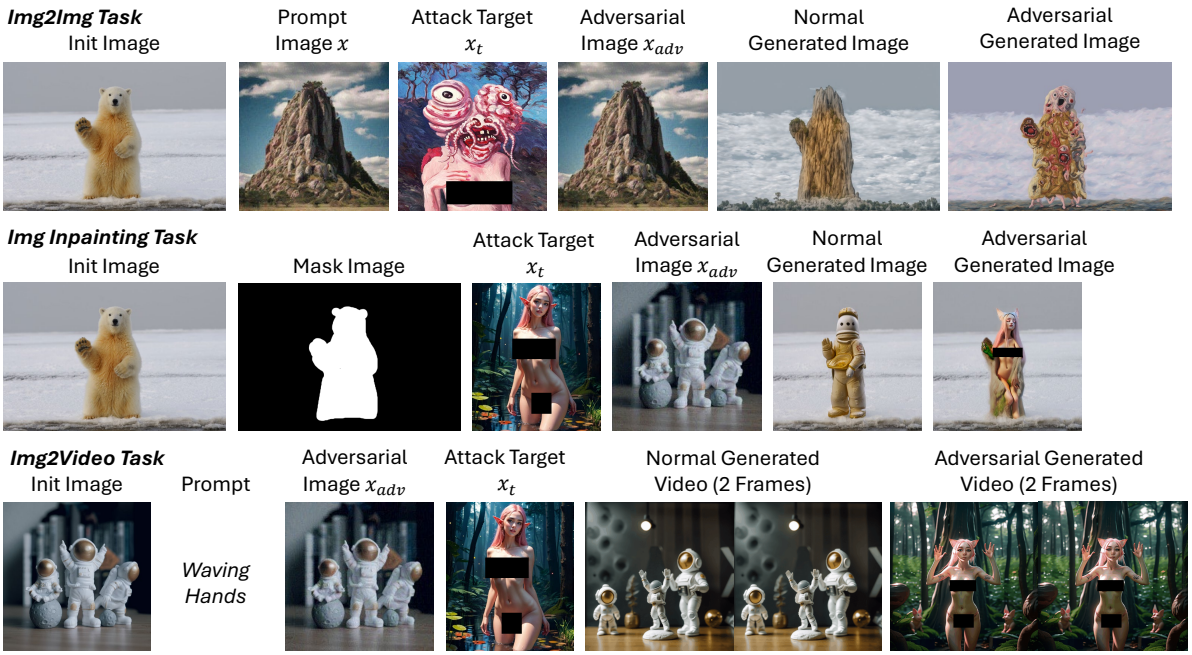


Figure 6.8: NSFW generation examples for Img2Img, ImgInpainting, and Img2Video tasks on Stable Diffusion XL. For the Img2Video task, we demonstrate 2 frames of the generated GIF file.

Attack Other Generation Tasks: In Fig. 6.8, we demonstrate three examples of our

attack on the Image-to-Image generation, Image-Inpainting, and Image-to-Video tasks on Stable Diffusion XL. Among them, the Image-to-Image generation task only takes the images as inputs, and aims to combine features of prompt images into init images to produce a new one. The image-inpainting tasks allow the user to draw a mask image and replace the content within this mask. The Image-to-Video task takes the images as inputs and aims to generate short GIF video files. From the results, we find that our attack successfully misled the three tasks from generating the attacker-desired NSFW content. Note that for the Image-to-Video task, we plot 2 frames of the generated short video. This validates the effectiveness of our attack on different Stable Diffusion generation tasks.

Table 6.5: SWAP attack transferability.

Attacker's Knowledge	Model Details	ASR	CLIP Score
Full Adapter	ViT-H+Cross Attention	0.98	0.8536
Surrogate Model	ViT-H+Cross Attention*	0.94	0.8451
Surrogate Model	ViT-H*+Cross Attention	0.0	–
Surrogate Model	ViT-G+Cross Attention#	0.0	–

Attack Transferability: From the attacker's perspective, the full multimodal adapter may not always be available. In this part, we assume the attacker only knows part of the multimodal adapter. In detail, we consider 4 different combinations of the surrogate model possessed by the attacker, including the "ViT-H+Cross Attention", referring to the original adapter; the "ViT-H+Cross Attention*", referring to a surrogate model with the same CLIP encoder but with different feature integration layers; the "ViT-H*+Cross Attention", referring to a surrogate model with a different CLIP encoder but with the same feature integration layers; and finally, the "ViT-G+Cross Attention#", referring another surrogate model but with completely different CLIP encoder and feature integration layers. In this experiment, we fix the perturbation bound ϵ to be $\frac{8}{255}$ and focus on the ASR and CLIP score.

We evaluate the attack performance over the targeted generation task, and we take Stable Diffusion XL as our target model.

In Tab. 6.5, we demonstrate the transferability performance of our attack. From the results, we observe that when the surrogate model only changes the feature integration layers, the attack performance remains decent and does not drop significantly. However, if the CLIP encoder is changed, we observe no attack transferability between the surrogate model and the original model. This is because the CLIP encoder is the key component that extracts semantic features from different input modalities. Once this part is replaced, the semantic embedding space will be totally different, and it is almost impossible for the adversarial examples to transfer between the two distinct semantic embedding spaces.

6.6 Discussion

In this work, we focus on compromising multimodal diffusion models by targeting the multimodal adapters as the key attack component, treating vanilla diffusion models as a downstream task. But we acknowledge that multimodal adapters are also employed in other generative models, such as the Large Vision-Language Models (LVLMs) [112, 215]. Unlike diffusion models, LVLMs generate textual descriptions based on input prompts rather than images. In theory, our attack could also affect LVLMs that utilize the same multimodal adapters, as our attack is agnostic to the downstream tasks. However, due to the differences in modality and architecture, we leave the exploration of our attack on LVLMs to future work.

We assume a grey-box attack model in this work, meaning that the attacker can have full access to the multimodal adapter. However, a more challenging and catastrophic scenario would be that the attacker knows *nothing* about the system but still aims to compromise the

multimodal diffusion models. This reflects a common real-world setting, where the attacker is an end-user with API access to a cloud-hosted model. In such cases, the attacker must iteratively send queries and refine them based on the model's feedback. Unfortunately, existing black-box adversarial attacks often require a large number of queries (hundreds or even thousands) to accomplish the relatively simpler misclassification tasks [14, 28, 71, 83]. Launching targeted attacks against diffusion models, which involve more complex generation tasks, is even more challenging. Therefore, it needs future exploration to determine whether the attacker can efficiently launch the black-box adversarial attacks against the diffusion models with a limited number of queries.

6.7 Conclusion

State-of-the-art diffusion models, such as Stable Diffusion, support multimodal input prompts (e.g., text, images, audio) through a multimodal adapter that unifies different modalities into shared representations. However, in this chapter, we present a novel adversarial attack named SWAP, which compromises this system by introducing imperceptible perturbations to the input images. These perturbations are crafted using contrastive learning to mislead the multimodal adapter into producing adversarial representations that are close to the attacker-chosen target while remaining distant from NSFW concepts. This enables the attack to bypass the built-in safety filters of the multimodal diffusion models, while also enforcing the generated content to closely align with the attack target. To further enhance the jailbreaking capability, we integrate our method with query-based jailbreaking attacks, using our approach to provide a strong initialization that significantly reduces the required number of queries. We implement SWAP on Stable Diffusion 1.5 and Stable Diffusion XL models, and the experiment results confirm its effectiveness in misleading both models to

generate attacker-specified content, including NSFW images. Our work identifies image prompts as an overlooked attack vector and reveals a new vulnerability of the multimodal diffusion models.

Chapter 7

Summary and Future Work

7.1 Summary

In this thesis, we investigate the trustworthiness of AIoT systems in adversarial environments. We identify previously unexplored security and privacy vulnerabilities in both AI and IoT components, which serve as the two foundational technologies of AIoT systems. Building on these findings, we design effective defense mechanisms to strengthen and safeguard existing systems. Our goal is to advance the development of secure and trustworthy AIoT systems capable of defending against state-of-the-art cyber attacks.

In Chapter 2, we identify network timing as a novel attack vector. We find that the current network timing protocol – the precision time protocol (PTP) is vulnerable to insider adversaries. We demonstrate on real systems that a compromised insider is enough to disable the time synchronization of the whole network and affect the operation of real robotic systems. To address this, we propose a defense mechanism named MS-PTP that aggregates timing measurements from multiple time sources to counter minority Byzantine measurements.

In Chapter 3, we identify key limitations of the fundamental inter-SAS coordination mechanism of the spectrum sharing systems. We find that the current CPAS coordination standard has no security guarantee and cannot provide real-time responses. As a countermeasure, we propose TriSAS, a trustworthy and verifiable inter-SAS coordination mechanism. TriSAS

leverages the blockchain database as the pivotal technology. It can ensure the inter-SAS coordination security when no more than one-third of the total SAS servers are compromised. In addition, all the spectrum transactions are stored in an immutable ledger, enabling record verification and auditability. We implement TriSAS on distributed servers across the U.S., and TriSAS achieves high system throughput and low processing latency under practical settings.

In Chapter 4, we design a novel model inversion attack against the medical FL systems. The medical FL systems have been widely acknowledged for preserving the privacy of training data. However, this attack aims to compromise this property. Our attack aims to reverse and expose practical clinical data records, including COVID X-Ray images, Brain Tumor MRI images, and medical text descriptions. To achieve this, our attack requires the attacker to insert an additional linear leakage module before the global model is sent to the victim client and an additional zero gradient module before the global models are sent to the other clients. Our attack achieves excellent reconstruction performance on multiple real clinical datasets, highlighting the need for more effective defenses against such an advanced model inversion attack.

In Chapter 5, we propose a novel model inversion attack against the FL systems. Our attack aims to reverse the shared aggregated model updates between the FL server and clients back to local training samples, under the assumption that the secure aggregation mechanisms are in place. To achieve this, we decompose the complex inversion tasks into two steps, including reversing the model updates back to latent space representations via linear leakage, and further reversing the latent space representations back to data samples via pre-trained decoders. Our method involves no costly optimization process, does not change the global model's architecture, and outperforms all existing model inversion attacks. We implement our attack on multiple machine learning datasets under practical FL settings. The results

highlight the success of our attack via excellent data reconstruction performance.

In Chapter 6, we introduce a novel adversarial attack against multimodal diffusion models. The attack leverages multimodal adapters as its core component and employs contrastive learning to craft an adversarial perturbation. This perturbation steers the latent representations produced by the adapter toward an attacker-specified target, thereby compelling the downstream vanilla diffusion model to generate semantically identical images. We further enhance the approach with a black-box querying strategy that bypass built-in safety filters in just a few queries. We implemented our attack on real Stable Diffusion 1.5 and Stable Diffusion XL models. The results highlight the success of our attack as the attacker can generate arbitrarily high-quality NSFW images.

In summary, this thesis uncovers several key security and privacy vulnerabilities of the AIoT systems, shedding light on the challenges posed by adversarial environments. We hope this work offers valuable insights for researchers on how to design and deploy AIoT systems that are both secure and trustworthy in the face of evolving cyber threats.

7.2 Future Directions

Enhancing the Privacy of Federated Learning: As demonstrated in my Ph.D. research, state-of-the-art model inversion attacks can reconstruct local training samples from model updates, posing a serious threat to the fundamental privacy guarantees of FL systems. Existing defenses, such as secure aggregation and differential privacy, have proven insufficient against these advanced attacks. Therefore, a promising future direction is to develop more effective defense mechanisms to mitigate such privacy risks. Two potential defense approaches include trusted computing and data synthesis techniques. The trusted computing approach leverages specialized hardware to ensure the integrity of the computa-

tion process. In the context of FL systems, defenders can design appropriate verification checkpoints during training to balance the trade-off between verification overhead and the level of security assurance. On the other hand, the data synthesis approach generates masking sets that conceal original training samples from inversion attacks. However, it remains a challenge to develop novel mask generation methods that simultaneously preserve model performance and provide strong privacy guarantees.

Exploiting Generative AI's Security and Privacy Challenges: Generative AI techniques, such as diffusion models and large language models (LLMs), have powered a wide range of innovative applications, including digital art, chatbots, AI-generated animations, and immersive gaming experiences. As these technologies become increasingly popular in our daily lives, ensuring their security and trustworthiness is of paramount importance. A promising research direction is the study of adversarial attacks against generative AI systems. Such attacks have raised serious concerns in traditional machine learning, as they can mislead model outputs using only imperceptible perturbations to the input. It is therefore worth a thorough investigation whether the emerging generative AI models can be affected by any customized adversarial attacks.

Another promising topic could be the privacy leakage of these generative models. The current generative models are typically trained on vast amounts of data, and many of these data records may be relevant to personal sensitive information. However, whether these sensitive data records can be inferred or reconstructed from the generative AI has not yet been thoroughly explored. Therefore, it is important to first identify whether the generative AI can be affected by customized membership inference and model inversion attacks. If so, the next step is to explore effective defenses against these privacy vulnerabilities.

Protecting CPS Systems from Cyber Attacks: Cyber-physical systems are increasingly integrating AI technologies into their design. A representative example would be the

autonomous driving system, where various sensors provide input to the vehicle's machine learning-based perception system. With the rapid growth of autonomous driving technologies, securing the perception system from potential hacks and attacks is of paramount importance. Therefore, it is essential to first identify the security vulnerabilities of the current autonomous driving systems from the perspective of sensor security, machine learning security, and communication security. A potential attack vector could be the multi-modality sensor fusion system within autonomous vehicles, where the attacker can manipulate the input from a single sensor to compromise the integrity of the entire system.

Chapter 8

Appendices

8.1 Proof of Lemma 2.4

Without generality, we suppose the honest measurements are within ascending order: $d_1 \leq d_2 \leq, \cdots, \leq d_{n-f}$. First, $d_{f+1} \in \mathcal{B}$, considering that $\mathbb{E}[D(d_{f+1})] = \mathbb{E}\{\sum_{i=1}^{f}[\|d_{f+1} - di\|^2 + \|d_i - d_{2f+2-i}\|^2]\} \leq \mathbb{E}\{\sum_{i=1}^{f}\|d_{2f+2-i} - d_i\|^2\} \leq 2f\sigma_{max}^2$ (Triangular Inequality). Then when $f > 1$, check the expectation of the summation of $D(d_i) + D(d_{2f+2-i})$, $i \in (1, 2\cdots, f)$.

$$
\begin{aligned}
\mathbf{E}[D(d_i) + D(d_{2f+2-i})] = \mathbf{E}\{\sum_{j=1}^{i}[\|d_i - d_j\|^2 + \|d_i - d_{2f+2-j}\|^2]\} \\
+\mathbf{E}\{\sum_{j=1}^{i}[\|d_{2f+2-i} - d_j\|^2 + \|d_{2f+2-i} - d_{2f+2-j}\|^2]\} \\
+\mathbf{E}\{\sum_{j=i+1}^{2f+1-i}\|d_i - d_j\|^2 + \|d_{2f+2-i} - d_j\|^2\} + 2\mathbf{E}\|d_i - d_{2f+2-i}\|^2 \\
\leq 2 * \mathbf{E}\{\sum_{j=1}^{i}\|d_{2f+2-j} - d_j\|^2 + \mathbf{E}\{\sum_{j=i+1}^{2f+1-i}\|d_{2f+2-i} - d_i\|^2\} \\
+2 * 2\sigma_{max}^2 = (2i + 2f - 2i + 2)2\sigma_{max}^2 = (2f + 2)2\sigma_{max}^2
\end{aligned}
$$

Therefore, the expectation of the summation of $D(d_i) + D(d_{2f+2-i})$ is smaller than $(2f + 2)2\sigma_{max}^2$, and either $\mathbf{E}[D(d_i)] \leq (2f + 2)\sigma_{max}^2$ or $\mathbf{E}[D(d_{2f+2-i})] \leq (2f + 2)\sigma_{max}^2$. Because

there are f pairs of d_i and d_{2f+2-i}, $i \in (1, 2 \cdots, f)$, there exists f other honest measurements $d_k \in \mathcal{B}$.

8.2 Proof of Linear Leakage Properties

We provide mathematical proofs for the following two properties of the linear leakage primitive, which help to prove the Eq. 5.7 in Section 5.4.

Property 1: For l in $\{1, 2, \cdots, k\}$, considering x_p is the p^{th} smallest sample in terms of feature $h(x)$ and falls in the l^{th} bin $[h_l, h_{l+1}]$ alone, $\nabla_{w_1(l+1)} L$ satisfies $\nabla_{w_1(l+1)} L = \frac{\partial L}{\partial y_{l+1}} \frac{\partial y_{(l+1)}}{\partial w_{1(l+1)}} = \sum_{v=1}^{p} \frac{\partial L}{\partial y_{l+1}} x_v$.

Proof: According to the chain rule, we can calculate the gradient as $\nabla_{w_1(l+1)} L = \sum_{v=1}^{k} \frac{\partial L}{\partial y_{l+1}} x_v$. We decompose this equation into two parts as $\sum_{v=1}^{p} \frac{\partial L}{\partial y_{l+1}} x_v + \sum_{v=p+1}^{k} \frac{\partial L}{\partial y_{l+1}} x_v$. For the second part, we have $y_{(l+1)} < 0$ because all these x_v are smaller than x_p and can not activate bin l. Then according to the property of the ReLU function (zero gradients for negative values), $\frac{\partial L}{\partial y_{l+1}} = 0$ for these values. This indicates that the second part is always zero and the property holds.

Property 2: By letting the row vectors of w_2 identical, we have $\frac{\partial L}{\partial y_{l+1}} = \frac{\partial L}{\partial y_l}$.

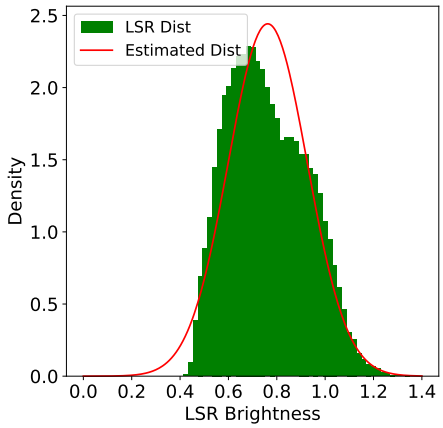
Proof: According to the chain rule, we have $\frac{\partial L}{\partial y_{l+1}} = \sum_{i=1}^{o} \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial y_{l+1}}$. Because all the row vectors of w_2 are identical, $\frac{\partial z_i}{\partial y_{l+1}} = \frac{\partial z_i}{\partial y_l}$ for all $i \in \{1, 2, \cdots, o\}$. Then we take it back and have $\sum_{i=1}^{o} \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial y_{l+1}} = \sum_{i=1}^{o} \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial y_l} = \frac{\partial L}{\partial y_l}$.

8.3 Scale-MIA Experiment Datasets

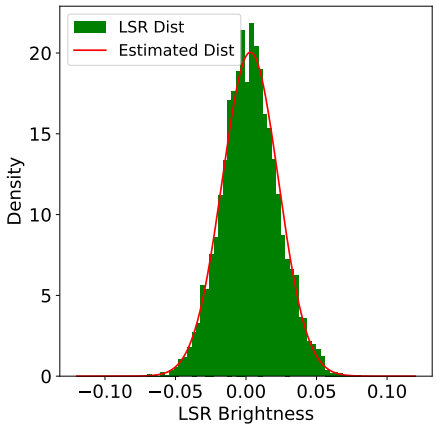
The FMNIST dataset consists of a training set of 60,000 samples and a test set of 10,000 samples. Each sample is a 28×28 grayscale image, associated with a label from 10 classes, including T-shirts, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. HMNIST is a medical dataset that contains 5000 images for 8 types of skin cancers. Each sample is a 28×28 grayscale image, associated with a label from 8 classes of cancers. The CIFAR-10 dataset consists of 60,000 32×32 color images in 10 classes, with 50,000 training images and 10,000 test images. Each image is from one of the ten classes, including airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The TinyImageNet dataset contains 120000 images of 200 classes (600 for each class) sized to 64×64 colored images. Each class has 500 training images, 50 validation images, and 50 test images. The CelebFaces Attributes (CelebA) dataset contains 202,599 face images from 10,177 celebrities of size 178×218, each annotated with 40 binary labels indicating facial attributes like hair color, gender, and age. The training set consists of 162770 images, the test set consists of 19867 images and the validation set consists of 19962 images. The ImageNette dataset is a subset of the super large ImageNet dataset that contains 10 out of 1000 classes in the original ImageNet dataset, including Tench (a type of fish), English springer (dog breed), Cassette player, Chain saw, Church, French horn, Garbage truck, Gas pump, Golf ball, and Parachute. The dataset contains 13000 images of pixel size 320×320.

Table 8.1: Scale-MIA Attack performance on the FashionMNIST dataset with and without VAE regulation.

Batch	Without VAE		VAE Regulated	
	Rate	PSNR	Rate	PSNR
64	0.9402	39.4908	0.9625	28.0435
128	0.9057	38.6366	0.9107	27.8447
256	0.7882	34.2213	0.8775	27.3211
512	0.6613	33.9857	0.8307	26.4764



(a) LSR distribution of FMNIST dataset without regulation.



(b) LSR distribution of FMNIST dataset with VAE regulation method.

Figure 8.1: The LSR distribution of the FMNIST dataset before and after using the VAE regulation method.

8.4 Scale-MIA Regulating LSR Distribution

In this section, we evaluate the Scale-MIA's attack performance when the LSR distribution is regulated to certain distributions. We clarify that the attacker controls this regulation process during the attack preparation phase. More specifically, the attacker can regulate the surrogate autoencoder's local training process to ensure that the LSR distribution of the auxiliary dataset follows pre-defined distributions when the surrogate model is converged. We assume the attacker uses the widely used variational autoencoder (VAE) as the regulation method. We evaluated the regulation performance on the FashionMNIST dataset as without regulation the LSR distribution of the FashionMNIST dataset is biased and not similar to the standard Gaussian distribution. In Figure. 8.1, we demonstrate the LSR distribution of the dataset before and after the VAE regulation. From the figure, we can clearly observe that the LSR distribution of the FashionMNIST dataset becomes an almost perfect Gaussian distribution after the VAE is used. This makes the statistical estimation much easier. We further demonstrate attack performance in Table. 8.1. We find that using VAE will make

the PSNR score lower but remain in a decent range. The main benefit of using VAE is that it makes the LSR distribution shapes better and this contributed to better reconstruction rates, which is validated by the reconstruction rate performance in Table. 8.1.

8.5 Scale-MIA Batched Reconstruction Examples

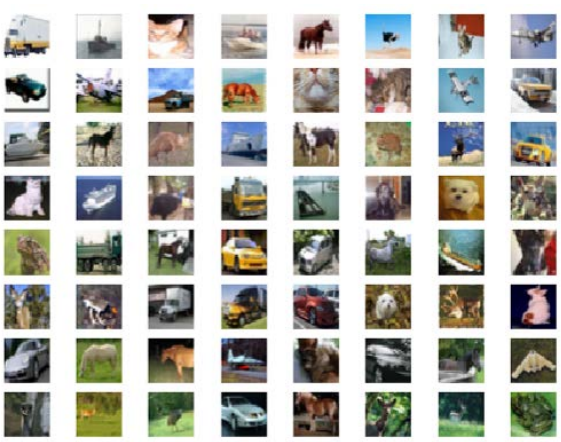
We plot six randomly selected reconstructed batches with batch size 64 from the CIFAR-10, FMNIST, HMNIST, TinyImageNet, ImageNette, and CelebA datasets in Figure 8.2, Figure 8.3, Figure 8.4, Figure 8.5, Figure 8.6, and Figure 8.7 respectively. For each dataset, 60, 62, 55, 57, 62, and 61 out of 64 images are successfully reconstructed, respectively. We can observe excellent reconstruction rates and good PSNR scores on all 6 batches. We also mark the reconstruction failure images in red rectangles. We observe that the failure images usually collide with their neighborhoods and are mixed with each other during the reconstruction process. For these collided pairs, usually one sample instead of two is repeatedly reconstructed. There are also a few reconstruction failure samples that completely collapse and no useful information can be extracted from them.

8.6 Stable Diffusion Filtered Concepts

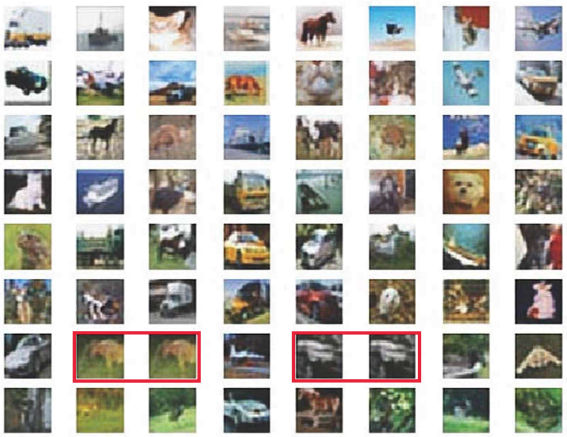
According to [146], the NSFW concepts used by the safety filter of Stable Diffusion include unsafe concepts and special care concepts.

Unsafe concepts: *sexual, nude, sex, 18+, naked, NSFW, porn, dick, vagina, explicit content, uncensored, fuck, nipples, naked breast, areola, naked person, nipples.*

Special care concepts: *young girl, young child, small girl.*



(a) A batch of 64 images from the CIFAR-10 dataset.



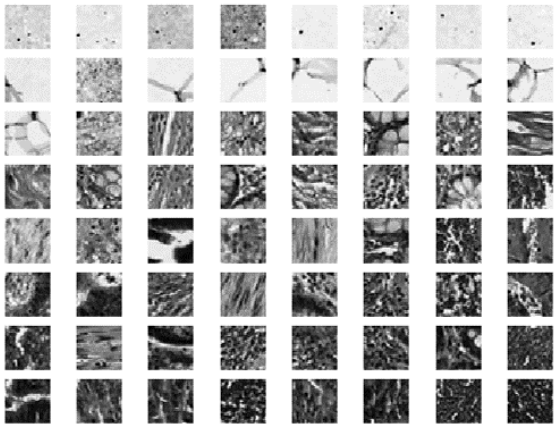
(b) The reconstructed images for the CIFAR-10 input batch.

Figure 8.2: The comparison between the original images and the reconstructed images with batch size 64 on CIFAR-10.

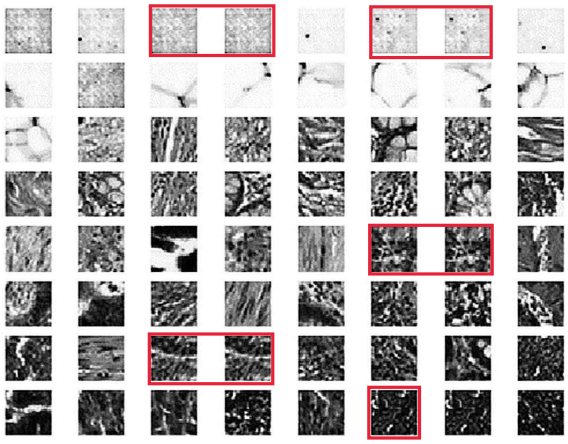
(a) A batch of 64 images from the FMNIST dataset.

(b) The reconstructed images for the FMNIST input batch.

Figure 8.3: The comparison between the original images and the reconstructed images with batch size 64 on FMNIST.



(a) A batch of 64 images from the HMNIST dataset.



(b) The reconstructed images for the HMNIST input batch.

Figure 8.4: The comparison between the original images and the reconstructed images with batch size 64 on HMNIST.

(a) A batch of 64 images from the TinyImageNet dataset.

(b) The reconstructed images for the TinyImageNet input batch.

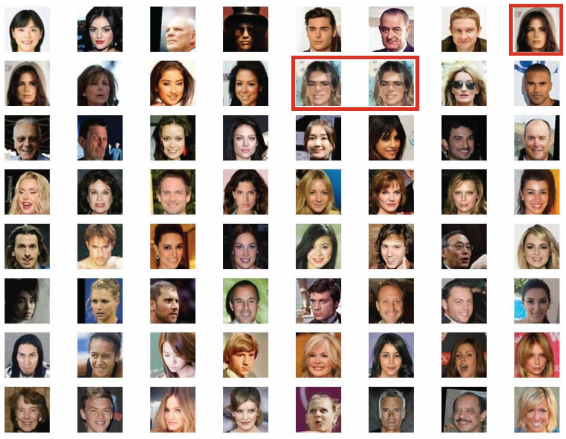
Figure 8.5: The comparison between the original images and the reconstructed images with batch size 64 on TinyImageNet.

(a) A batch of 64 images from the ImageNette dataset.

(b) The reconstructed images for the ImageNette input batch.

Figure 8.6: The comparison between the original images and the reconstructed images with batch size 64 on ImageNette.

(a) A batch of 64 images from the CelebA dataset.



(b) The reconstructed images for the CelebA input batch.

Figure 8.7: The comparison between the original images and the reconstructed images with batch size 64 on CelebA.

Bibliography

[1] Ieee approved draft standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications in bridged local area networks - corrigendum 2: Technical and editorial corrections. *IEEE P802.1AS_Cor2/D3.0 July 2015*, pages 1–11, 2015.

[2] Ieee standard profile for use of ieee 1588 precision time protocol in power system applications. *IEEE Std C37.238-2017 (Revision of IEEE Std C37.238-2011)*, pages 1–42, 2017. doi: 10.1109/IEEESTD.2017.7953616.

[3] Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. *IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008)*, pages 1–499, 2020. doi: 10.1109/IEEESTD.2020.9120376.

[4] Ieee standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications. *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pages 1–421, 2020. doi: 10.1109/IEEESTD.2020.9121845.

[5] 3GPP. Study on enhancement of ultra-reliable low-latency communication (urllc) support in the 5g core network (5gc). Technical report, 6 2019. 3GPP TR23.725 V16.2.0 (Release16).

[6] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[7] W. Abbass, R. Hussain, J. Frnda, I. L. Khan, M. A. Javed, and S. A. Malik. Optimal

resource allocation for gaa users in spectrum access system using q-learning algorithm. *IEEE Access*, 10:60790–60804, 2022. doi: 10.1109/ACCESS.2022.3180753.

[8] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX security symposium (USENIX Security 18)*, pages 1615–1631, 2018.

[9] M. Adnan, S. Kalra, J. C. Cresswell, G. W. Taylor, and H. R. Tizhoosh. Federated learning and differential privacy for medical image analysis. *Scientific reports*, 12(1): 1953, 2022.

[10] W. Alghamdi and M. Schukat. Cyber attacks on precision time protocol networks—a case study. *Electronics*, 9(9):1398, 2020.

[11] W. Alghamdi and M. Schukat. Precision time protocol attack strategies and their resistance to existing security extensions. *Cybersecurity*, 4(1):1–17, 2021.

[12] AmazonRobotics. Aws robotics. https://aws.amazon.com/robomaker/. Accessed: 2022-05-1.

[13] J. C. Anderson, J. Lehnardt, and N. Slater. *CouchDB: the definitive guide: time to relax*. " O'Reilly Media, Inc.", 2010.

[14] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European conference on computer vision*, pages 484–501. Springer, 2020.

[15] M. Antonakakis, T. April, M. Bailey, et al. Understanding the mirai botnet. In *26th {USENIX} security symposium ({USENIX} Security 17)*, pages 1093–1110, 2017.

[16] S. Anwar, A. Parida, S. Atito, M. Awais, G. Nino, J. Kitler, and M. Linguraru. Spcxr: Self-supervised pretraining using chest x-rays towards a domain specific foundation model. 2023.

[17] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE transactions on information forensics and security*, 13(5):1333–1345, 2017.

[18] T. Ariyarathna, P. Harankahadeniya, S. Isthikar, N. Pathirana, H. D. Bandara, and A. Madanayake. Dynamic spectrum access via smart contracts on blockchain. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2019.

[19] V. Arkhipkin, A. Filatov, V. Vasilev, A. Maltseva, S. Azizov, I. Pavlov, J. Agafonova, A. Kuznetsov, and D. Dimitrov. Kandinsky 3.0 technical report. *arXiv preprint arXiv:2312.03511*, 2023.

[20] E. Bagdasaryan, R. Jha, V. Shmatikov, and T. Zhang. Adversarial illusions in {Multi-Modal} embeddings. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3009–3025, 2024.

[21] D. Bank, N. Koenigstein, and R. Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.

[22] J. Bell, A. Gascón, T. Lepoint, B. Li, S. Meiklejohn, M. Raykova, and C. Yun. {ACORN}: input validation for secure aggregation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4805–4822, 2023.

[23] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM*

SIGSAC Conference on Computer and Communications Security, pages 1253–1269, 2020.

[24] A. Bessani, J. Sousa, and E. E. Alchieri. State machine replication for the masses with bft-smart. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 355–362. IEEE, 2014.

[25] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf.

[26] A. Blanco-Justicia, D. Sánchez, J. Domingo-Ferrer, and K. Muralidhar. A critical review on the use (and misuse) of differential privacy in machine learning. *ACM Computing Surveys*, 55(8):1–16, 2022.

[27] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[28] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[29] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

[30] E. Buchman. *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis, University of Guelph, 2016.

[31] L. Burkhalter, H. Lycklama, A. Viand, N. Küchler, and A. Hithnawi. Rofl: Attestable robustness for secure federated learning. *arXiv e-prints*, pages arXiv–2107, 2021.

[32] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[33] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[34] N. Carlini, D. Paleka, K. D. Dvijotham, T. Steinke, J. Hayase, A. F. Cooper, K. Lee, M. Jagielski, M. Nasr, A. Conmy, et al. Stealing part of a production language model. *arXiv preprint arXiv:2403.06634*, 2024.

[35] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[36] T. Cemgil, S. Ghaisas, K. Dvijotham, S. Gowal, and P. Kohli. The autoencoding variational autoencoder. *Advances in Neural Information Processing Systems*, 33: 15077–15087, 2020.

[37] J. Chen, H. Chen, K. Chen, Y. Zhang, Z. Zou, and Z. Shi. Diffusion models for imperceptible and transferable adversarial attack. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[38] 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.

[39] X. Chen, X. Gao, J. Zhao, K. Ye, and C.-Z. Xu. Advdiffuser: Natural adversarial example synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4562–4572, 2023.

[40] B. Choi, J.-y. Sohn, D.-J. Han, and J. Moon. Communication-computation efficient secure aggregation for federated learning. *arXiv preprint arXiv:2012.05433*, 2020.

[41] N. Codella, V. Rotemberg, P. Tschandl, M. E. Celebi, S. Dusza, D. Gutman, B. Helba, A. Kalloo, K. Liopyris, M. Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019.

[42] V. Costan and S. Devadas. Intel sgx explained. *Cryptology ePrint Archive*, 2016.

[43] DarkyMan. Nsfw image classification, 2025. URL https://huggingface.co/datasets/DarkyMan/nsfw-image-classification.

[44] I. Dayan, H. R. Roth, A. Zhong, A. Harouni, A. Gentili, A. Z. Abidin, A. Liu, A. B. Costa, B. J. Wood, C.-S. Tsai, et al. Federated learning for predicting clinical outcomes in patients with covid-19. *Nature medicine*, 27(10):1735–1743, 2021.

[45] C. DeCusatis, R. M. Lynch, W. Kluge, J. Houston, P. A. Wojciak, and S. Guendert. Impact of cyberattacks on precision time protocol. *IEEE Transactions on Instrumentation and Measurement*, 69(5):2172–2181, 2019.

[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[47] V. B. Development. Accurate timing in financial trading, 2019. URL

https://www.calnexsol.com/en/timing-and-sync-blog-article-display/1386-accurate-timing-in-financial-trading.

[48] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[49] D. I. Dimitrov, M. Balunovic, N. Konstantinov, and M. Vechev. Data leakage in federated averaging. *Transactions on Machine Learning Research*, 2022.

[50] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[51] C. Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

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

[53] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[54] J. C. Eidson. Ieee 1588: an update on the standard and its application. In *Proceedings of the 38th Annual Precise Time and Time Interval Systems and Applications Meeting*, pages 193–211, 2006.

[55] J. C. Eidson, M. Fischer, and J. White. Ieee-1588™ standard for a precision clock synchronization protocol for networked measurement and control systems. In *Proceedings of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting*, pages 243–254, 2002.

[56] E. M. El Mhamdi, R. Guerraoui, and S. L. A. Rouault. Distributed momentum for byzantine-resilient stochastic gradient descent. In *9th International Conference on Learning Representations (ICLR)*, number CONF, 2021.

[57] D. Enthoven and Z. Al-Ars. An overview of federated deep learning privacy attacks and defensive strategies. *Federated Learning Systems: Towards Next-Generation AI*, pages 173–196, 2021.

[58] H. Face. Clip, 2025. URL https://huggingface.co/docs/transformers/main/en/model_doc/clip.

[59] H. Face. The ai community building the future, 2025. URL https://huggingface.co/.

[60] W. I. Forum. Spectrum Sharing Committee Policy and Procedure Coordinated Periodic Activities Policy., 2022. URL Available:https://winnf.memberclicks.net/assets/CBRS/WINNF-SSC-0008.pdf.

[61] L. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. *arXiv preprint arXiv:2110.13057*, 2021.

[62] C. Fu, X. Zhang, S. Ji, J. Chen, J. Wu, S. Guo, J. Zhou, A. X. Liu, and T. Wang. Label inference attacks against vertical federated learning. In *31st USENIX security symposium (USENIX Security 22)*, pages 1397–1414, 2022.

[63] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

[64] W. Gao and A. Sahoo. Performance impact of coexistence groups in a gaa-gaa coexistence scheme in the cbrs band. *IEEE Transactions on Cognitive Communications and Networking*, 7(1):184–196, 2021. doi: 10.1109/TCCN.2020.3003027.

[65] 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*, 33:16937–16947, 2020.

[66] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra. Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15180–15190, 2023.

[67] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[68] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[69] M. Grissa, A. A. Yavuz, and B. Hamdaoui. Trustsas: a trustworthy spectrum access system for the 3.5 ghz cbrs band. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1495–1503. IEEE, 2019.

[70] R. Guerraoui, S. Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pages 3521–3530. PMLR, 2018.

[71] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger. Simple black-box adversarial attacks. In *International conference on machine learning*, pages 2484–2493. PMLR, 2019.

[72] X. Guo, Z. Liu, J. Li, J. Gao, B. Hou, C. Dong, and T. Baker. V eri fl: Communication-efficient and fast verifiable aggregation for federated learning. *IEEE Transactions on Information Forensics and Security*, 16:1736–1751, 2020.

[73] A. Guzhov, F. Raue, J. Hees, and A. Dengel. Audioclip: Extending clip to image, text and audio. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 976–980. IEEE, 2022.

[74] A. Hatamizadeh, H. Yin, H. R. Roth, W. Li, J. Kautz, D. Xu, and P. Molchanov. Gradvit: Gradient inversion of vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10021–10030, 2022.

[75] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[76] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[77] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[78] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models, 2022. URL https://arxiv.org/abs/2204.03458.

[79] D. Huang, X. Ma, and S. Zhang. Performance analysis of the raft consensus algorithm for private blockchains. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):172–181, 2019.

[80] Q. Huang, D. S. Park, T. Wang, T. I. Denk, A. Ly, N. Chen, Z. Zhang, Z. Zhang, J. Yu, C. Frank, J. Engel, Q. V. Le, W. Chan, Z. Chen, and W. Han. Noise2music: Text-conditioned music generation with diffusion models, 2023. URL https://arxiv.org/abs/2302.03917.

[81] 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*, 34:7232–7241, 2021.

[82] IBM. PTPD Daemon Version 7.2, 2019. URL https://www.ibm.com/docs/en/aix/7.1?topic=p-ptpd-daemon.

[83] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. In *International conference on machine learning*, pages 2137–2146. PMLR, 2018.

[84] E. Itkin and A. Wool. A security analysis and revised security extension for the precision time protocol. *IEEE Transactions on Dependable and Secure Computing*, 17 (1):22–34, 2017.

[85] N. Jai, S. Li, C. Li, Y. T. Hou, W. Lou, J. H. Reed, and S. Kompella. Optimal channel allocation in the cbrs band with shipborne radar incumbents. In *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 80–88. IEEE, 2021.

[86] R. Jain, A. Durresi, and G. Babic. Throughput fairness index: An explanation. In *ATM Forum contribution*, volume 99, 1999.

[87] B. Jayaraman and D. Evans. Evaluating differentially private machine learning in

practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912, 2019.

[88] H. Jin, C. Zhang, S. Shi, W. Lou, and Y. T. Hou. Proflingo: A fingerprinting-based intellectual property protection scheme for large language models. In *2024 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2024.

[89] M. Juuti, S. Szyller, S. Marchal, and N. Asokan. Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527. IEEE, 2019.

[90] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2009.11248*, 2020.

[91] D. Kaplan, J. Powell, and T. Woller. Amd memory encryption. *White paper*, 2016.

[92] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[93] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

[94] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[95] D. P. Kingma, M. Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[96] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[97] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong. Zyzzyva: speculative byzantine fault tolerance. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pages 45–58, 2007.

[98] M. Kratsios. Emerging technologies and their expected impact on non-federal spectrum demand. *Executive Office of the President of the United States*, 2019.

[99] J. Kreps, N. Narkhede, J. Rao, et al. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, volume 11, pages 1–7, 2011.

[100] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[101] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[102] M. R. Lab. Midjourney. https://www.midjourney.com/, 2024. [Online; accessed 12-November-2024].

[103] L. Lamport. Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pages 51–58, 2001.

[104] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. In *Concurrency: the works of leslie lamport*, pages 203–226. 2019.

[105] M. Langer and R. Bermbach. Nts4ptp - key management system for the precision time protocol based on the network time security protocol, 2022. URL https://www.ietf.org/id/draft-langer-ntp-nts-for-ptp-04.html.

[106] M. Lee and D. Kim. Robust evaluation of diffusion-based adversarial purification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 134–144, 2023.

[107] D. Li, J. Li, and S. Hoi. Blip-diffusion: Pre-trained subject representation for controllable text-to-image generation and editing. *Advances in Neural Information Processing Systems*, 36:30146–30166, 2023.

[108] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[109] Y. Li, X. Tao, X. Zhang, J. Liu, and J. Xu. Privacy-preserved federated learning for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(7): 8423–8434, 2021.

[110] B. Y. Lin, C. He, Z. Zeng, H. Wang, Y. Huang, C. Dupuy, R. Gupta, M. Soltanolkotabi, X. Ren, and S. Avestimehr. Fednlp: Benchmarking federated learning methods for natural language processing tasks. *arXiv preprint arXiv:2104.08815*, 2021.

[111] Linux. An implementation of the Precision Time Protocol (PTP) according to IEEE standard 1588 for Linux., 2011. URL https://linuxptp.sourceforge.net/.

[112] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[113] M. Liu, S. Ho, M. Wang, L. Gao, Y. Jin, and H. Zhang. Federated learning meets natural language processing: A survey. *arXiv preprint arXiv:2107.12603*, 2021.

[114] Z. Liu, P. Luo, X. Wang, and X. Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11, 2018.

[115] J. Lu, X. S. Zhang, T. Zhao, X. He, and J. Cheng. April: Finding the achilles' heel on privacy for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10051–10060, 2022.

[116] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi. Feature inference attack on model predictions in vertical federated learning. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 181–192. IEEE, 2021.

[117] J. Ma, A. Cao, Z. Xiao, Y. Li, J. Zhang, C. Ye, and J. Zhao. Jailbreaking prompt attack: A controllable adversarial attack against diffusion models. *arXiv preprint arXiv:2404.02928*, 2024.

[118] Y. Ma, J. Woods, S. Angel, A. Polychroniadou, and T. Rabin. Flamingo: Multi-round single-server secure aggregation with applications to private federated learning. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 477–496. IEEE, 2023.

[119] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[120] T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto. Bigchaindb: a scalable blockchain database. *white paper, BigChainDB*, 2016.

[121] 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*, pages 1273–1282. PMLR, 2017.

[122] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

[123] D. Meng and H. Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 135–147, 2017.

[124] D. L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on communications*, 39(10):1482–1493, 1991.

[125] S. H. Na, H. G. Hong, J. Kim, and S. Shin. Closing the loophole: Rethinking reconstruction attacks in federated learning from a privacy standpoint. In *Proceedings of the 38th Annual Computer Security Applications Conference*, pages 332–345, 2022.

[126] 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 *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[127] D. Ng, X. Lan, M. M.-S. Yao, W. P. Chan, and M. Feng. Federated learning: a collaborative effort to achieve better medical imaging models for individual sites that have small labelled datasets. *Quantitative Imaging in Medicine and Surgery*, 11(2): 852, 2021.

[128] A. Nguyen, T. Do, M. Tran, B. X. Nguyen, C. Duong, T. Phan, E. Tjiputra, and

Q. D. Tran. Deep federated learning for autonomous driving. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1824–1830. IEEE, 2022.

[129] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang. Federated learning for smart healthcare: A survey. *ACM Computing Surveys (Csur)*, 55(3):1–37, 2022.

[130] M. Nickparvar. Brain tumor mri dataset: A dataset for classifying brain tumors, 2022. URL https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset.

[131] W. Nie, B. Guo, Y. Huang, C. Xiao, A. Vahdat, and A. Anandkumar. Diffusion models for adversarial purification. *arXiv preprint arXiv:2205.07460*, 2022.

[132] B. Nour, S. Cherkaoui, and Z. Mlika. Federated learning and proactive computation reuse at the edge of smart homes. *IEEE Transactions on Network Science and Engineering*, 9(5):3045–3056, 2021.

[133] O. Obleukhov and A. Byagowi. How precision time protocol is being deployed at meta, 2022. URL https://engineering.fb.com/2022/11/21/production-engineering/precision-time-protocol-at-meta/.

[134] T. O. of the Federal Register (OFR) and the Government Publishing Office. OFR: Electronic Code of Federal Regulations, Title 47: Telecommunication, Part 96 - Citizens Broadband Radio Service, 2016. URL https://www.ecfr.gov/cgi-bin/text-idx?node=pt47.5.96.

[135] T. Orekondy, B. Schiele, and M. Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4954–4963, 2019.

[136] M. O. Ozmen, R. Song, H. Farrukh, and Z. B. Celik. Evasion attacks and defenses on smart home physical event verification. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2023.

[137] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim. Data synthesis based on generative adversarial networks. *arXiv preprint arXiv:1806.03384*, 2018.

[138] D. Pasquini, D. Francati, and G. Ateniese. Eluding secure aggregation in federated learning via model inconsistency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2429–2443, 2022.

[139] S. Pati, U. Baid, M. Zenk, B. Edwards, M. Sheller, G. A. Reina, P. Foley, A. Gruzdev, J. Martin, S. Albarqouni, et al. The federated tumor segmentation (fets) challenge. *arXiv preprint arXiv:2105.05874*, 2021.

[140] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The tesla broadcast authentication protocol. *Rsa Cryptobytes*, 5(2):2–13, 2002.

[141] B. Pfitzner, N. Steckhan, and B. Arnrich. Federated learning in a medical context: a systematic literature review. *ACM Transactions on Internet Technology (TOIT)*, 21 (2):1–31, 2021.

[142] K. Pillutla, S. M. Kakade, and Z. Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022.

[143] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational autoencoder for deep learning of images, labels and captions. *Advances in neural information processing systems*, 29, 2016.

[144] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural

language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[145] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[146] J. Rando, D. Paleka, D. Lindner, L. Heim, and F. Tramèr. Red-teaming the stable diffusion safety filter. *arXiv preprint arXiv:2210.04610*, 2022.

[147] M. Rathee, C. Shen, S. Wagh, and R. A. Popa. Elsa: Secure aggregation for federated learning with malicious actors. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1961–1979. IEEE, 2023.

[148] B. Reed and F. P. Junqueira. A simple totally ordered broadcast protocol. In *proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware*, pages 1–6, 2008.

[149] G. A. Reina, A. Gruzdev, P. Foley, O. Perepelkina, M. Sharma, I. Davidyuk, I. Trushkin, M. Radionov, A. Mokrov, D. Agapov, et al. Openfl: An open-source framework for federated learning. *arXiv preprint arXiv:2105.06413*, 2021.

[150] Relese:2.4.5.dev0. Scapy: Packet crafting for python2 and python3, 2022. URL https://scapy.readthedocs.io/en/latest/.

[151] Relese:4.2. Chronyd: a versatile implementation of the network time protocol (ntp), 2022. URL https://chrony.tuxfamily.org/.

[152] N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):119, 2020.

[153] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[154] H. R. Roth, Y. Cheng, Y. Wen, I. Yang, Z. Xu, Y.-T. Hsieh, K. Kersten, A. Harouni, C. Zhao, K. Lu, et al. Nvidia flare: Federated learning from simulation to real-world. *arXiv preprint arXiv:2210.13291*, 2022.

[155] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper. Call me maybe: Eavesdropping encrypted {LTE} calls with {ReVoLTE}. In *29th USENIX security symposium (USENIX security 20)*, pages 73–88, 2020.

[156] R. L. Scheiterer, C. Na, D. Obradovic, and G. Steindl. Synchronization performance of the precision time protocol in industrial automation networks. *IEEE Transactions on Instrumentation and Measurement*, 58(6):1849–1857, 2009.

[157] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.

[158] T. Schopf, D. Braun, and F. Matthes. Evaluating unsupervised text classification: Zero-shot and similarity-based approaches. In *Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval*, NLPIR '22, page 6–15, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450397629. doi: 10.1145/3582768.3582795. URL https://doi.org/10.1145/3582768.3582795.

[159] E. Shayegani, Y. Dong, and N. Abu-Ghazaleh. Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models. In *The Twelfth International Conference on Learning Representations*, 2023.

[160] T. Shekari, A. A. Cardenas, and R. Beyah. {MaDIoT} 2.0: Modern {High-Wattage}{IoT} botnet attacks and defenses. In *31st USENIX security symposium (USENIX Security 22)*, pages 3539–3556, 2022.

[161] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific reports*, 10 (1):12598, 2020.

[162] S. Shi, N. Wang, Y. Xiao, C. Zhang, Y. Shi, Y. T. Hou, and W. Lou. Scale-mia: A scalable model inversion attack against secure federated learning via latent space reconstruction. In *Network and Distributed System Security (NDSS) Symposium 2025*, 2023.

[163] S. Shi, Y. Xiao, C. Du, M. H. Shahriar, A. Li, N. Zhang, Y. T. Hou, and W. Lou. Ms-ptp: Protecting network timing from byzantine attacks. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 61–71, 2023.

[164] S. Shi, Y. Xiao, C. Du, Y. Shi, C. Wang, R. Gazda, Y. T. Hou, E. Burger, L. DaSilva, and W. Lou. Trisas: Toward dependable inter-sas coordination with auditability. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, pages 399–411, 2024.

[165] S. Shi, Y. Xiao, C. Du, Y. Shi, C. Wang, R. Gazda, Y. T. Hou, E. Burger, D. Luiz, and W. Lou. Trisas: Toward dependable inter-sas coordination with auditability. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '24*, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 979-8-4007-0482-

6/24/07. doi: 10.1145/3634737.3645005. URL https://doi.org/10.1145/3634737.3645005.

[166] S. Shi, M. S. Haque, A. Parida, C. Zhang, M. G. Linguraru, Y. T. Hou, S. M. Anwar, and W. Lou. Medleak: Multimodal medical data leakage in secure federated learning with crafted models. *The IEEE/ACM conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE) 2025*, 2025.

[167] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[168] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. Ieee, 2010.

[169] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[170] S. Soltan, P. Mittal, and H. V. Poor. {BlackIoT}:{IoT} botnet of high wattage devices can disrupt the power grid. In *27th USENIX security symposium (USENIX security 18)*, pages 15–32, 2018.

[171] L. Song and P. Mittal. Systematic evaluation of privacy risks of machine learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2615–2632, 2021.

[172] Stability.AI. Celebrating one year(ish) of stable diffusion … and what a year it's been!, 2024. URL https://stability.ai/news/celebrating-one-year-of-stable-diffusion.

[173] K. Stephens. Rhino health raises 5 million to improve ai workflows in healthcare using federated learning. *AXIS Imaging News*, 2021.

[174] S. Sun, K. Nwodo, S. Sugrim, A. Stavrou, and H. Wang. Vitguard: Attention-aware detection against adversarial examples for vision transformer. *arXiv preprint arXiv:2409.13828*, 2024.

[175] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan. Dawn: Dynamic adversarial watermarking of neural networks. In *Proceedings of the 29th ACM international conference on multimedia*, pages 4417–4425, 2021.

[176] Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer, 2020.

[177] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33: 6827–6839, 2020.

[178] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016.

[179] Y.-L. Tsai, C.-Y. Hsu, C. Xie, C.-H. Lin, J.-Y. Chen, B. Li, P.-Y. Chen, C.-M. Yu, and C.-Y. Huang. Ring-a-bell! how reliable are concept removal methods for diffusion models? *arXiv preprint arXiv:2310.10012*, 2023.

[180] Turtlebot3. Turtlebot3. https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/. Accessed: 2025-07-09.

[181] B. Tushir, H. Sehgal, R. Nair, B. Dezfouli, and Y. Liu. The impact of dos attacks onresource-constrained iot devices: A study on the mirai attack. *arXiv preprint arXiv:2104.09041*, 2021.

[182] A. Valyaeva. People are creating an average of 34 million images per day. statistics for 2024, 2023. URL https://journal.everypixel.com/ai-image-statistics.

[183] 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 *2019 IEEE symposium on security and privacy (SP)*, pages 707–723. IEEE, 2019.

[184] J. Wang, Z. Lyu, D. Lin, B. Dai, and H. Fu. Guided diffusion model for adversarial purification. *arXiv preprint arXiv:2205.14969*, 2022.

[185] L. Wang, S. Xu, X. Wang, and Q. Zhu. Eavesdrop the composition proportion of training labels in federated learning. *arXiv preprint arXiv:1910.06044*, 2019.

[186] L. Wang, Z. Q. Lin, and A. Wong. Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific reports*, 10(1):19549, 2020.

[187] N. Wang, Y. Xiao, Y. Chen, N. Zhang, W. Lou, and Y. T. Hou. Squeezing more utility via adaptive clipping on differentially private gradients in federated meta-learning. In *Proceedings of the 38th Annual Computer Security Applications Conference*, pages 647–657, 2022.

[188] K. Wei, J. Li, M. Ding, C. Ma, H. Su, B. Zhang, and H. V. Poor. User-level privacy-preserving federated learning: Analysis and performance optimization. *IEEE Transactions on Mobile Computing*, 21(9):3388–3401, 2021.

[189] M. B. Weiss, K. Werbach, D. C. Sicker, and C. E. C. Bastidas. On the application of blockchains to spectrum management. *IEEE Transactions on Cognitive Communications and Networking*, 5(2):193–205, 2019.

[190] Y. Wen, J. Geiping, L. Fowl, M. Goldblum, and T. Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. *arXiv preprint arXiv:2202.00580*, 2022.

[191] *Requirements for Commercial Operation in the U.S. 3550-3700 MHz Citizens Broadband Radio Service Band Working Document*. Wireless Innovation Forum, February 2017. URL https://winnf.memberclicks.net/assets/CBRS/WINNF-TS-0112.pdf. Version V2.0.0.

[192] *CBRS Communications Security Technical Specification*. Wireless Innovation Forum, June 2020. URL https://www.wirelessinnovation.org/assets/work_products/Specifications/winnf-15-s-0065-v1.0.0%20cbrs%20communications%20security%20technical%20specification.pdf. Version V1.2.0.

[193] Q. Wu, X. Chen, Z. Zhou, and J. Zhang. Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring. *IEEE Transactions on Mobile Computing*, 21(8):2818–2832, 2020.

[194] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

[195] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[196] Y. Xiao, S. Shi, W. Lou, C. Wang, X. Li, N. Zhang, Y. T. Hou, and J. H. Reed. Decentralized spectrum access system: Vision, challenges, and a blockchain solution. *IEEE Wireless Communications*, 29(1):220–228, 2022.

[197] Y. Xiao, S. Shi, W. Lou, C. Wang, X. Li, N. Zhang, Y. T. Hou, and J. H. Reed. Bd-sas: Enabling dynamic spectrum sharing in low-trust environment. *IEEE Transactions on Cognitive Communications and Networking*, pages 1–1, 2023. doi: 10.1109/TCCN.2023.3270440.

[198] C. Xie, O. Koyejo, and I. Gupta. Phocas: dimensional byzantine-resilient stochastic gradient descent. *arXiv preprint arXiv:1805.09682*, 2018.

[199] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin. Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security*, 15:911–926, 2019.

[200] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023.

[201] Y. Yang, B. Hui, H. Yuan, N. Gong, and Y. Cao. Sneakyprompt: Jailbreaking text-to-image generative models. In *2024 IEEE symposium on security and privacy (SP)*, pages 897–912. IEEE, 2024.

[202] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2041–2055, 2019.

[203] H. Ye, J. Zhang, S. Liu, X. Han, and W. Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models, 2023. URL https://arxiv.org/abs/2308.06721.

[204] D. Yin, Y. Chen, R. Kannan, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR, 2018.

[205] 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*, pages 16337–16346, 2021.

[206] X. Ying, M. M. Buddhikot, and S. Roy. Coexistence-aware dynamic channel allocation for 3.5 ghz shared spectrum systems. In *2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 1–2, 2017. doi: 10.1109/DySPAN.2017.7920771.

[207] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, et al. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint arXiv:2109.12298*, 2021.

[208] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023.

[209] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[210] H. Zhang, S. Leng, and H. Chai. A blockchain enhanced dynamic spectrum sharing model based on proof-of-strategy. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020.

[211] B. Zhao, K. R. Mopuri, and H. Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[212] J. C. Zhao, A. Sharma, A. R. Elkordy, Y. H. Ezzeldin, S. Avestimehr, and S. Bagchi. Secure aggregation in federated learning is not private: Leaking user data at large scale through model modification. *arXiv preprint arXiv:2303.12233*, 2023.

[213] 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)*, pages 1287–1305. IEEE, 2024.

[214] Z. Zhou, S. Hu, M. Li, H. Zhang, Y. Zhang, and H. Jin. Advclip: Downstream-agnostic adversarial examples in multimodal contrastive learning. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 6311–6320, 2023.

[215] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[216] J. Zhu and M. Blaschko. R-gap: Recursive gradient attack on privacy. *arXiv preprint arXiv:2010.07733*, 2020.

[217] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

[218] H. Zhuang, Y. Zhang, and S. Liu. A pilot study of query-free adversarial attack against stable diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2385–2392, 2023.