TOWARDS TRUSTWORTHY MULTIAGENT AND MACHINE LEARNING SYSTEMS

BY

SHANGYU XIE

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science
in the Graduate College of the
Illinois Institute of Technology

Approved _____
Adviser

Chicago, Illinois
December 2022

ACKNOWLEDGEMENT

First of all, I would like to sincerely thank my supervisor Prof. Yuan Hong for his continuous support and help during my whole Ph.D. study. He leads me to privacy research, which has influenced me profoundly in my research career.

Second, I would like to thank my dissertation committee members, including Prof. Peng-Jun Wan, Prof. Zuyi Li and Prof. Yan Yan for their generous counsel and comments on my research. I also want to thank my collaborators for their work in our joint publications. I would additionally like to thank my undergraduate advisor Prof. Xinbing Wang and Prof. Lei Yang for their help prior to my Ph.D.

Last but not least, I am so grateful to my parents for their endless love.

AUTHORSHIP STATEMENT

I, Shangyu Xie (S. X.), attest that the work in this thesis is substantially my own. In accordance with the disciplinary norm of Computer Science (see IIT Faculty Handbook, Appendix S), the following collaborations occurred in the thesis:

Dr. Yuan Hong (Y. H.) contributed to general research ideas, helped the paper writing and guided the experimental evaluation as is the norm of Ph.D. supervisor (see IIT Faculty Handbook, Appendix S).

Dr. Peng-Jun Wan (P. J. W.) of Illinois Institute of Technology, Chicago IL contributed to the discussion of load balancing paper (Chapter 2).

Han Wang (H. W.) of Illinois Institute of Technology, Chicago IL collaborated on coding of energy trading and robustness evaluation projects (Chapter 3 and 4).

Dr. My T. Thai (M. T. T.) of University of Florida, Gainesville, FL contributed to the discussion of the energy trading project (Chapter 3).

Dr. Yu Kong (Y. K) of Rochester Institute of Technology, Rochester, NY contributed to the discussion and helped the paper writing (Chapter 4).

Dr. Yan Yan (Y. Y) of Illinois Institute of Technology, Chicago IL contributed to the discussion of the poisoning attack (Chapter 5) and helped the paper writing.

Bingyu Liu (B. L.) of Illinois Institute of Technology, Chicago IL collaborated on coding a cloud-based inference project (Chapter 7).

Dr. Meisam Mohamaddy (M. M.) of Iowa State University, Ames, IA collaborated on research of outsourcing paper (Chapter 8) and helped the paper writing.

Dr. Lingyu Wang (L. W) of Concordia University, Montreal CA contributed to the discussion of outsourcing paper (Chapter 8) and helped the paper writing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This dissertation aims to systematically research the "trustworthy" Multiagent and Machine Learning systems in the context of the Internet of Things (IoT) system, which mainly consists of two aspects: *data privacy* and *robustness*. Specifically, *data privacy* concerns about the protection of the data in one given system, i.e., the data identified to be sensitive or private cannot be disclosed directly to others; *robustness* refers to the ability of the system to defend/mitigate the potential attacks/threats, i.e., maintaining the stable and normal operation of one system.

Starting from the smart grid, a representative multiagent system in the IoT, I demonstrate two works on improving data privacy and robustness in aspects of different applications, load balancing and energy trading, which integrates secure multiparty computation (SMC) protocols for normal computation to ensure data privacy. More significantly, the schemes can be readily extended to other applications in IoT, e.g., connected vehicles, mobile sensing systems.

For the machine learning, I have studied two main areas, i.e., computer vision and natural language processing with the privacy and robustness correspondingly. I first present the comprehensive robustness evaluation study of the DNN-based video recognition systems with two novel proposed attacks in both test and training phase, i.e., adversarial and poisoning attacks. Besides, I also propose the adaptive defenses to fully evaluate such two attacks, which can thus further advance the robustness of system. I also propose the privacy evaluation for the language systems and show the practice to reveal and address the privacy risks in the language models.

Finally, I demonstrate a private and efficient data computation framework with the cloud computing technology to provide more robust and private IoT systems.

CHAPTER 1

INTRODUCTION

1.1 Overview

With the recent great development of the communication, computation and other computer-related technologies, the Internet of Things (IoT) is more than a future concept, which has tremendously changed the traditional sensing of the physical world and bridged the gap of the physical world to the digital world. IoT technologies have been boosting the connections between things and humans, which are widely studied and deployed in various applications, such as smart grid, connected vehicles, smart city, smart transportation, smart home. Considering the critical functionality of the IoT, how to build a trustworthy IoT system has been brought on the table, which are mainly defined with the two intriguing properties: 1) *data privacy*; 2) *robustness*.

On the one hand, the huge amount of data is generated, transferred and utilized inside the IoT systems for every moment, which could contain lots of privacy information of humans. The privacy concerns of the data in IoT systems can greatly limit the advancement and utilization of the whole system. Then how to privately and collaboratively use data in IoT is crucial and emerging. On the other hand, the IoT system has been facilitated with various advanced computer technologies such as machine learning and cloud computing to boost the performance. Such newly proposed systems may bring extra robustness and security issues while improving the efficiency. For example, a serious issue for the machine learning models is the adversarial attacks, which fool the model with adversarial examples and thus influence the safety-related system using ML, e.g., autonomous driving. Studying the robustness of a system against such attacks could help to improve stability of the system and ensure normal operation.

Focusing on the two properties, i.e., data privacy and robustness above, my dissertation systematically studies two representative components in regime of IoT systems:

- **Multiagent systems (MAS)**: an advanced computer system consisting of multiple interactive intelligent agents with the integration of communication and computer technology. One representative application of MAS is smart grid, which integrates the MAS with the power system to improve both stability and efficiency.
- **Machine learning (ML) systems**: the widely deployed smart systems with a core machine learning-based model, especially with deep neural network (DNN) model. For example, the DNN-based video recognition system can be integrated with security surveillance to detect the anomaly events more accurately. Language model can be integrated with the voice assistant system.

For the multiagent systems, I study the practice of smart grid system, which serves as an important component in IoT systems. I demonstrate privacy-enhancing research in two popular applications with smart grid: power load and supply balancing (Chapter 2) and energy trading market (Chapter 3). Both applications can help to improve the stability of the power grid and also economic benefits while protecting data.

For the machine learning systems, I present two popular domains, i.e., computer vision and natural language processing, respectively. First, I propose a complete robustness evaluation for video recognition systems in aspects of newly proposed adversarial attack (Chapter 4) and poisoning attack (Chapter 5). Second, I study the privacy risks of language models and give a comprehensive evaluation with previous protection schemes to show the privacy risks (Chapter 6). I also present a preliminary

work (Chapter 7) on utilizing cloud to do the secure DNN inference, e.g., image classification as advanced data analysis in IoT systems.

Table 1.1. Organization of Dissertation

System	Applications	Data Privacy	Robustness
Multiagent	Smart Grid	Chapter 2, 3	Chapter 2
Machine Learning	Computer Vision	Chapter 7	Chapter 4, 5
	Natural Language	Chapter 6	Chapter 6

Additionally, I present my work on the secure outsourcing computation (Chapter 8) with the facilitation of cloud computing for the general data in IoT-based domains, e.g., network trace data and location data. I propose a general outsourcing framework to protect various prefix-preserving data, which can further improve the utilization of data in IoT systems.

Table 1.1 demonstrates the organization of the dissertation with following chapters. The abstract of each chapter are also given below.

1.2 Privacy Preserving Load Balancing on the Power Grid

Microgrids equipped with renewable energy resources have proven to be critical building blocks on the power grid that can greatly improve the grid performance. A promising application would be enabling microgrids to utilize their local energy for further balancing the regional supply and demand at different times – ensuring better system economics and reliability. However, due to the privacy concerns on continuously revealing each microgrid's local data for deriving real-time optimal balancing decisions, the application of such promising cooperative technique is still limited. In this paper, we design an efficient cryptographic protocol for privately balancing the regional supply and demand, as well as each microgrid's local supply and

demand in real time. We prove the security of our protocol against both passive and active adversaries. Meanwhile, we implemented a prototype of the PAIRING system that integrates cryptographic protocol and the power transmission network. We mount the real smart grid datasets into PAIRING in real time for system evaluations. The experimental results demonstrate the practicality of our system by scaling to hundreds of microgrids with high accuracy and efficient system performance.

1.3 Private Distributed Energy Trading Market

The smart grid incentivizes distributed agents with local generation (e.g., smart homes, and microgrids) to establish multi-agent systems for enhanced reliability and energy consumption efficiency. Distributed energy trading has emerged as one of the most important multi-agent systems on the power grid by enabling agents to sell their excessive local energy to each other or back to the grid. However, it requests all the agents to disclose their sensitive data (e.g., each agent's fine-grained local generation and demand load). In this paper, to the best of our knowledge, we propose the first privacy preserving distributed energy trading framework, *Private Energy Market* (PEM), in which all the agents privately compute an optimal price for their trading (ensured by a Nash Equilibrium), and allocate pairwise energy trading amounts without disclosing sensitive data (via novel cryptographic protocols). Specifically, we model the trading problem as a non-cooperative Stackelberg game for all the agents (i.e., buyers and sellers) to determine the optimal price, and then derive the pairwise trading amounts. Our PEM framework can privately perform all the computations among all the agents without a trusted third party. We prove the *privacy*, *individual rationality*, and *incentive compatibility* for the PEM framework. Finally, we conduct experiments on real datasets to validate the effectiveness and efficiency of the PEM.

1.4 Robustness Evaluation of Video Recognition Systems

Widely deployed deep neural network (DNN) models have been proven to be vulnerable to adversarial perturbations in many applications (e.g., image, audio and text classifications). To date, there are only a few adversarial perturbations proposed to deviate the DNN models in video recognition systems by simply injecting 2D perturbations into video frames. However, such attacks may overly perturb the videos without learning the spatio-temporal features (across temporal frames), which are commonly extracted by DNN models for video recognition. To our best knowledge, we propose the first black-box attack framework that generates universal 3-dimensional (U3D) perturbations to subvert a variety of video recognition systems. U3D has many advantages, such as (1) as the transfer-based attack, U3D can universally attack multiple DNN models for video recognition without accessing to the target DNN model; (2) the high transferability of U3D makes such universal black-box attack easy-to-launch, which can be further enhanced by integrating queries over the target model when necessary; (3) U3D ensures human-imperceptibility; (4) U3D can bypass the existing state-of-the-art defense schemes; (5) U3D can be efficiently generated with a few pre-learned parameters, and then immediately injected to attack *real-time* DNN-based video recognition systems. We have conducted extensive experiments to evaluate U3D on multiple DNN models and three large-scale video datasets. The experimental results demonstrate its superiority and practicality.

1.5 Stealthy Poisoning Attacks on Video Classification

Deep Neural Networks (DNNs) have been proven to be vulnerable to poisoning attacks that poison the training data with a trigger pattern and thus manipulate the trained model to misclassify data instances. In this paper, we study the poisoning attacks on video recognition models. We reveal the major limitations of the state-of-the-art poisoning attacks on *stealthiness* and *attack effectiveness*: (i) the frame-by-frame poisoning trigger may cause temporal inconsistency among the video frames which

can be leveraged to easily detect the attack; (ii) the feature collision-based method for crafting poisoned videos could lack both generalization and transferability. To address these limitations, we propose a novel stealthy and efficient poisoning attack framework which has the following advantages: (i) we design a 3D poisoning trigger as natural-like textures, which can maintain temporal consistency and human-imperceptibility; (ii) we formulate an ensemble attack oracle as the optimization objective to craft poisoned videos, which could construct convex polytope-like adversarial subspaces in the feature space and thus gain more generalization; (iii) our poisoning attack can be readily extended to the black-box setting with good transferability. We have experimentally validated the effectiveness of our attack (e.g., up to 95% success rates with only less than $\sim 0.5\%$ poisoned dataset).

1.6 Privacy Evaluation of Language Models

A private learning scheme TextHide was recently proposed to protect the private text data via instance encoding. We propose a novel reconstruction attack to break TextHide, and thus unveil the privacy risks of instance encoding. Our attack would advance the development of privacy preserving machine learning in the context of natural language processing. As a complete privacy evaluation, we also integrate differential privacy into the instance encoding scheme, and thus provide a provable guarantee against privacy attacks. The experimental results also show that the proposed scheme can defend against privacy attacks while ensuring learning utility (as a trade-off).

1.7 Privacy Preserving Cloud-based DNN Inference

Deep learning as a service (DLaaS) has been intensively studied to facilitate the wider deployment of the emerging deep learning applications. However, DLaaS may compromise the privacy of both clients and cloud servers. Although some privacy

preserving deep neural network (DNN) techniques have been proposed by composing cryptographic primitives, the challenges on computational efficiency have not been fully addressed due to the complexity of DNN models and expensive cryptographic primitives. In this paper, we propose a novel privacy preserving cloud-based DNN inference framework ("PROUD"), which greatly improves the computational efficiency. Finally, we conduct experiments on two datasets to validate the effectiveness and efficiency for the PROUD while benchmarking with the state-of-the-art techniques.

1.8 Secure Outsourcing Computation on the Cloud

Property preserving encryption techniques have significantly advanced the utility of encrypted data in various data outsourcing settings (e.g., the cloud). However, while preserving certain properties (e.g., the prefixes or order of the data) in the encrypted data, such encryption schemes are typically limited to specific data types (e.g., prefix-preserved IP addresses) or applications (e.g., range queries over order-preserved data), and highly vulnerable to the emerging inference attacks which may greatly limit their applications in practice. In this paper, to the best of our knowledge, we make the first attempt to generalize the prefix preserving encryption via *prefix-aware* encoding that is not only applicable to more general data types (e.g., geo-locations, market basket data, DNA sequences, numerical data and timestamps) but also secure against the inference attacks. Furthermore, we present a generalized *multi-view outsourcing* framework that generates multiple *indistinguishable* data views in which one view fully preserves the utility for data analysis, and its accurate analysis result can be obliviously retrieved. Given any specified privacy leakage bound, the computation and communication overheads are minimized to effectively defend against different inference attacks. We empirically evaluate the performance of our outsourcing framework against two common inference attacks on two different real datasets: the check-in location dataset and network traffic dataset, respectively. The

experimental results demonstrate that our proposed framework preserves both privacy (with bounded leakage and indistinguishability of data views) and utility (with 100% analysis accuracy).

CHAPTER 2

PRIVACY PRESERVING LOAD BALANCING ON THE POWER GRID

2.1 Introduction

The smart grid has integrated a large number of renewable energy resources (e.g., solar panels and wind turbines) which provide supplementary power supply [1–3]. Microgrids equipped with such renewable energy resources can be isolated into autonomous power islands to feed their demand load with local energy sources (e.g., wind and solar) [1]. To date, many organizations or regional districts (e.g., hospitals, university campuses, groups of households and industrial sites) have deployed their microgrids by manipulating and maintaining their own energy resources in addition to the external feed. More importantly, besides feeding local demand, microgrids have been identified as the key building blocks of the power grid for cooperatively improving the grid performance with their flexibility, interoperability and scalability [4, 5]. Recently, novel cooperative models among microgrids have attracted significant interests in both industry and academia [2, 3, 6, 7].

Balancing the supply and demand at all times is essential for both energy saving and stability of the power system [8]. The goal is to balance supply and demand within a tight margin in real time: if supply exceeds demand, besides storing the extra energy (may result in huge energy loss), voltage spike would occur in the power system; when the supply lags behind demand, the voltage sags. Both of these unbalanced situations would be detrimental to power grid operations and devices connected to the grid [9]. In the recent smart grid infrastructure, the deployed microgrids (which are both power suppliers and consumers) could facilitate the main grid to further balance the regional supply and demand – ensuring better *system economics* and *reliability* [10]. For example, at peak times, the regional demand may exceed the supply, then microgrids can reduce their external supply and consume their local energy; at off-peak times,

microgrids can increase their external supply to some extent (while balancing their own supply and demand). Thus, the application of such supply and demand balancing involves multiple parties (main grid [1] and microgrids) to cooperate with each other, as shown in Figure 2.1 (any entity with local generation can be considered as a microgrid, e.g., a smart home with solar panels).

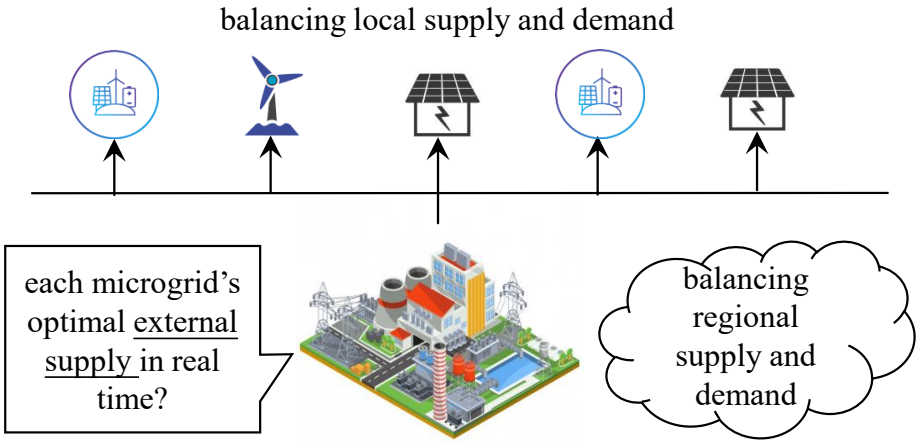


Figure 2.1. Balancing Multiparty Supply and Demand on the Power Grid

However, the above cooperation requests all the parties to jointly compute the real-time optimal energy allocation with their private local data (*most of which are generated in real time*), such as the regional supply, each microgrid's demand load, maximum local supply and maximum tolerable gap between its supply and demand. Clearly, disclosing these data for optimizing the supply and demand balancing decisions would explicitly compromise their privacy. For instance, the demand load of a household may leak the pattern of using appliances at different times [12, 13] while the local supply of a microgrid (e.g., hospital) could reveal its generation capacities and patterns in the organization [14, 15].

[1]Main grid is referred to the substation of the involved microgrids (which is a part of the main grid). Regional supply is considered as the supply from the substation, which is adjusted for its covered region [11].

Technical Contributions.[2] Although numerous privacy preserving schemes [12, 15, 18] have been proposed to address the privacy concerns in the smart grid, most of them focus on the smart metering data and propose relevant privacy preserving metering applications. Essentially, there are two main categories of such works: 1) encryption of the metering data for computation or aggregation in specific applications, e.g., regional statistics [19], aggregation [20], and billing [18]; 2) obfuscation of the smart metering data with a defined privacy model [13, 21]. Note that, the first category of techniques cannot be directly applicable to the studied balancing problem since they are designed for specific applications. The second category of techniques can be tailored to let the microgrids preprocess their metering data (i.e., consumption and generation) and the main grid solve the optimization problems in real time. However, they may result in significant utility loss by the metering data obfuscation and greatly increase the computational load for the main grid. To address these limitations, we propose a light-weight cryptographic protocol under Secure Multiparty Computation (SMC) [22, 23], and implement our PAIRING system based on the cryptographic protocol. The major technical contributions of this paper are detailed as follows.

- **Protocol and its Security.** To the best of our knowledge, we take the first step to propose a novel efficient cryptographic protocol for enabling secure collaborations among microgrids that address the *privacy*, *collusion* and *integrity* issues in the smart grid infrastructure. The cryptographic protocol securely seeks for each microgrid's *external supply (at different times)* for optimally balancing the multiagent (microgrids) supply and demand (both regional and local) in real time. To construct our cryptographic protocol, we leverage Homomorphic Encryption (e.g., Paillier Cryptosystem [24]) and Garbled Circuits [22] (e.g., the FAIRPLAY system [25]).

[2]This work has been published in ACM AAMAS [16] and IEEE TIFS [17].

Furthermore, we first provide formal security proof for our proposed protocol against passive adversaries under the SMC theory [22, 23] (each party's view is shown to be simulated in polynomial time to guarantee *privacy*). Furthermore, we conducted experiments to validate that the proposed cryptographic protocol can *mitigate the collusion attacks* [26] among the adversarial main grid and microgrids. Finally, we also show that our protocol can provide *verifiability* to detect data integrity attacks [27, 28] against active adversaries.

- **Privacy Preserving System Prototype.** Since energy should be routed from the main grid to multiple microgrids based on the computational decisions of the protocol (external supplies) for balancing multiparty supply and demand, the secure communication protocol should be *integrated with the power system*. Then, we design and implement a privacy preserving system PAIRING based on the proposed cryptographic protocol. To the best of our knowledge, our PAIRING system is the first *real-time* system to securely collaborate the microgrids on the power grid with the integration of both secure multiparty computation, communication and power distribution.

- **Real-time System Performance.** Since the time series demand load and supply of each party (including the regional supply) are frequently generated in real time, the deployed system (w.r.t. the cryptographic protocol) should be *highly efficient* and *scalable*. As validated in the system evaluations, our PAIRING system enables hundreds of microgrids and the main grid to optimally balance their supply and demand without disclosing their private data, and continuously transmit energy in real time with *high accuracy* and *negligible latency*.

2.2 Preliminaries

In this section, we present some preliminaries. Table 2.1 shows some frequently

used notations.

Table 2.1. The Notation Table in Chapter 2

Symbol	Definition
G	main grid
S^t	the regional supply of main grid at time t
M_i	the ith microgrid where $i \in [1, n]$
s_i^t, d_i^t	microgrid M_i's local supply and demand at time t
x_i^t	the external supply of microgrid M_i at time t
η_i	the energy transmission efficiency between G and M_i
ξ_i	the balancing margin of microgrid M_i
$\bar{x}_i^t$	the optimal external supply of microgrid M_i at time t

2.2.1 Cooperative Supply and Demand Balancing. Many microgrids are equipped with renewable energy sources and storage devices to generate local energy, which may not be sufficient to feed the local demand occasionally. At this time, external energy from the main grid should still be requested [29]. On the contrary, if the amount of any microgrid's local supply exceeds its local demand at any time t, we will consider the (*isolated*) mode [1] at that time, and the unconsumed energy will be locally stored in the energy storage device rather than sharing to other parties [3].

We now formulate such cooperative model. Given n microgrids $\forall i \in [1, n], M_i$, we denote the main grid G's regional supply allocated for all the microgrids at time t as S^t (excluding the supply for non-microgrid consumers). In addition, we denote each microgrid M_i's local demand load and supply as d_i^t and s_i^t, respectively, and its external supply as x_i^t. The variable x_i^t denotes how much energy is externally requested

from the main grid by microgrid M_i at time t. To facilitate the regional supply and demand balancing, it may not equal to $d_i - s_i$ since each microgrid has the capacity of local energy storage (e.g., a battery). While transmitting electricity from the main grid G to each microgrid M_i, the energy transmission efficiency [30] can be defined as $\eta_i \in [0, 1]$, which is mainly determined by the distance between M_i and G as well as the power quality data. Then, as microgrid M_i requests external supply x_i^t from main grid G, the amount $\frac{x_i^t}{\eta_i}$ should be routed by G since the energy transmission would result in the amount of $\frac{x_i^t(1-\eta_i)}{\eta_i}$ loss over the power transmission wires [30].

Specifically, at time t, a cooperative model is to find the *optimal external supply* $\bar{x}_i^t$ of individual microgrid $M_i, i \in [1, n]$ such that the overall deviation between the regional demand $\sum_{i=1}^{n} \frac{x_i^t}{\eta_i}$ and supply S^t is minimized.[2] Then, we can formulate the objective function as below:

$$\min : \left| \sum_{i=1}^{n} \frac{x_i^t}{\eta_i} - S^t \right| \tag{2.1}$$

Meanwhile, the deviation between each microgrid M_i's overall supply (local s_i^t and external x_i^t) and local demand d_i^t should be bounded by a tight balancing margin ξ_i (which can be specified by itself as a ratio or value; the smaller the margin, the closer the supply and demand) [31, 32].

$$\forall i \in [1, n], \left| x_i^t + s_i^t - d_i^t \right| \leq \xi_i \tag{2.2}$$

Hence, the cooperative supply and demand balancing problem at time t can be mathematically formulated as below:

---

[2]We minimize the deviation between regional supply and demand rather than bounding it within a margin. If strictly bounding the deviation is required, S^t can be readily adjusted by the main grid with energy dispatch [11].

$$\min : \left| \sum_{i=1}^{n} \frac{x_i^t}{\eta_i} - S^t \right| \quad \text{(at time t)}$$
$$s.t. \begin{cases} |x_1^t + s_1^t - d_1^t| \leq \xi_1 \\ |x_2^t + s_2^t - d_2^t| \leq \xi_2 \\ \vdots \qquad \vdots \qquad \vdots \\ |x_n^t + s_n^t - d_n^t| \leq \xi_n \\ x_i^t \geq 0, \eta_i \in [0,1], i \in [1,n] \end{cases} \tag{2.3}$$

Since we aim at proposing a privacy preserving system running continuously over any period, our designed protocol splits the real time balancing problem into individual balancing problems in continuous equal-length time slots (each time slot can be very short, e.g., close to real time), as shown in Equation 2.3. In each time slot, the nonlinear programming (NLP) problem is securely formulated and solved within the time slot. After securely deriving the optimal solution at time t, each microgrid will learn how much external supply it will request from the main grid x_i^t. Finally, if $x_i^t + s_i^t > d_i^t$, the excessive energy $x_i^t + s_i^t - d_i^t$ will be stored and rolled over to its local supply at the next time slot $t + 1$.

2.2.2 Cryptographic Building Blocks. We adopt homomorphic encryption [24,33,34] and garbled circuit [22,23] as the cryptographic building blocks to construct our protocols.

Homomorphic Encryption (e.g., Naccache-Stern cryptosystem [33], Paillier cryptosystem [24], Okamoto-Uchiyama cryptosystem [34] is a semantically-secure public key encryption with an additional property to generate the ciphertext of an arithmetic operation between two plaintexts by other operations between their individual ciphertexts. For instance, Pailler Cryptosystem [24] has the following properties:

1. Homomorphic addition: given two plaintexts A and B as well as the public-private key pair (pk, sk), the ciphertext of $(A + B)$ can be derived as $Enc_{pk}(A + B) = Enc_{pk}(A) \otimes Enc_{pk}(B)$, where $\otimes$ denotes the multiplication of ciphertexts (in some abelian group).

2. Self-blinding: any ciphertext can be transformed to another ciphertext without changing the plaintext.

3. Probabilistic: if one plaintext A is encrypted for different times, the ciphertexts are different.

Garbled Circuit was originally proposed by Yao [22]. It enables two semi-honest parties to jointly compute a function $f(x_1, x_2)$ without disclosing their private inputs x_1, x_2: one party creates the garbled circuit and the other party evaluates the circuit to generate the result. In our protocols, we leverage garbled circuit (e.g., the FAIRPLAY system [25]) to realize secure comparison in the protocol.

2.2.3 Operational Constraints in the Power System. Note that Equation 2.3 provides a core form for the studied problem which involves the balancing constraints. Moreover, some operational constraints within each microgrid (e.g., internal power quality, battery capacity for storing excessive energy) [35]) and the power transmission network (maintained by the main grid, e.g., external power quality, conversion/phase synchronization) [11] can be readily incorporated into the cooperative model and cryptographic protocol since such constraints are locally computed by either each microgrid or the main grid. For instance, internal power quality may also arouse energy loss within each microgrid, which can be calculated and updated in the protocol by the microgrid itself.

Thus, for simplifying notations, we design our cryptographic protocol based on the core form (Equation 2.3). In our PAIRING system implementation, additional

operational constraints are integrated with proper local computations on the OpenDSS platform [36] (with revised IEEE-123 bus system, refer to Section 2.5.1).

2.2.4 Private Data. The protocol/system will be executed in an $(n+1)$-party setting: the main grid G and n microgrids. All parties' private data over any period $\forall t \in [1, m]$ are illustrated as below:

- Main grid G privately holds its regional supply at different times $S^t, t \in [1, m]$. Moreover, the energy transmission efficiencies $i \in [1, n], \eta_i$ are originally known to the main grid G which maintains the power transmission network and chooses the power quality utilized for transmitting energy to each microgrid. Since η_i does not involve any private information, it can be shared to M_i.
- Microgrid $M_i, i \in [1, n]$ privately holds its local demand load d_i^t, local supply s_i^t and the selection of running modes as well as its requested external energy x_i^t.
- The global objective (for regional balancing) is jointly held by G and n microgrids while each constraint (for local balancing) is privately held by each microgrid.

2.2.5 Threat Model and Security Properties. We now discuss the threat model and security properties of our protocol. The details of security analysis are given in Section 3.5. Recall that all the $(n+1)$ parties generate their own private data for secure computation. We assume the main grid G is *semi-honest* but may *collude* with ℓ microgrids (where $\ell < n$) to compromise the remaining parties' private data. In addition, besides colluding with the main grid G, all the microgrids would be *malicious* by tampering with the protocol messages to compromise data integrity (e.g., injecting false data [28]) in the system. In our protocol, we assume that all the messages are transmitted via a secure channel.

To mitigate the above security threats, our proposed protocol/system will

desire the following security properties.

- **Privacy**: while executing the protocol over any period $\forall t \in [1, m]$, main grid G and each microgrid M_i will only learn the microgrid's requested external supplies $\forall t \in [1, m], \bar{x}_i^t$, nothing else; microgrids cannot learn any private information from each other.
- **Collusion Mitigation**: the confidence of successful collusion attacks (among main grid and microgrids) is limited.
- **Verifiability**: the integrity of exchanged messages/data can be verified in the protocol to detect the false data.

Notice that, we assume that the main grid ("substation") tries to provide true data for computation in practice since correctly balancing the regional supply and demand would be beneficial to the grid performance. Thus, our protocol/system is assumed to not protect against the case that malicious main grid tampers with the protocol messages.

2.3 Protocol Design

2.3.1 Overview. Recall that each party's demand and supply (which are the inputs of the problem) are continuously generated in sequence. Thus the nonlinear balancing problem (Equation 2.3) should be solved in real time. Then, at any time $t \in [1, m]$, all the parties (G and $\forall i \in [1, n], M_i$) securely solve the NLP problem to get their share of the optimal solution (i.e., M_i obtains $\bar{x}_i^t$ at time t), and M_i requests the external energy amount $\bar{x}_i^t$ from G.

As shown in Protocol 1, in initialization, main grid G and all the microgrids generate their own key pairs (pk, sk) and $\forall i \in [1, n], (pk_i, sk_i)$, and share the public

// Sub-protocols: **SHPA, SC, SA, SR**

1 main grid G generates a key pair (pk, sk), and distributes its public key pk to all the parties

2 **for** *microgrid* $M_i : i \leftarrow 1$ **to** n **do**

3 microgrid M_i generates a key pair (pk_i, sk_i), and distributes its public key pk_i to all the parties

4 **for** *timestamp* $t \leftarrow 1$ **to** m **do**

5 **for** *microgrid* $M_i : i \leftarrow 1$ **to** n **do**

6 M_i generates two hash values: $h(\frac{d_i^t - s_i^t + \xi_i}{\eta_i})$ and $h(\frac{d_i^t - s_i^t - \xi_i}{\eta_i})$, and then sends them to G (for verifiability)

7 G and all the microgrids jointly call sub-protocol **SC** (which calls 2-round sub-protocol **SHPA** and Secure Comparison)

8 **if** *Case (III) is returned in SC (Line 7)* **then**

9 G and all the microgrids jointly call sub-protocol **SA** (which calls λ-round sub-protocol **SHPA** and Secure Comparison)

// At time t, M_i has received the optimal or near-optimal $\bar{x}_i^t$ in three Cases

10 **for** *microgrid* $M_i : i \leftarrow 1$ **to** n **do**

11 M_i requests the amount of $\bar{x}_i^t$ energy from G

12 G verifies the data integrity (details are given in Section 2.4.3)

13 **if** ***ACCEPT*** *is returned by verifying the data from all of* $M_1, \ldots, M_n$ **then**

14 G transmits energy $\frac{\bar{x}_i^t}{\eta_i}$ to M_i where $i \in [1, n]$

15 G and all the microgrids jointly call sub-protocol **SR**

Algorithm 1: Overview of PAIRING

keys pk and $pk_1, \ldots, pk_n$ to all the parties (keys are generated per Homomorphic Encryption, e.g., Paillier Cryptosystem [24]).

Then at each time $t \in [1, m]$, all the parties (G and $\forall i \in [1, n], M_i$) jointly call sub-protocol Secure Categorization (SC) and possibly call sub-protocol Secure Approximation (SA) to derive the optimal external supplys $\forall i \in [1, n], \bar{x}_i^t$ (SA is only called in a certain output case of SC). Then, each microgrid $M_i, i \in [1, n]$ requests the energy with amount $\bar{x}_i^t$ from G at time t. Note that both SC and SA also call another sub-protocol Secure Hierarchically Paired Aggregation (SHPA).

Finally, main grid G verifies the integrity of messages (detailed in Section 2.4.3), and transmits energy $\frac{\bar{x}_i^t}{\eta_i}$ to M_i. Before moving to the next time slot, all the parties (G and $\forall i \in [1, n], M_i$) jointly call the sub-protocol Secure Rollover (SR) to locally store the excessive energy for the next time slot. Sub-protocols SC, SA, SHPA and SR will be elaborated in the upcoming subsections. Figure 2.2 outlines the primary components of the designed protocol for our PAIRING system.

2.3.2 Securely Seeking Optimal External Supplies at Time t. Our PAIRING system securely solves the balancing problem at each specific time t in real time. We then first look at time t. Indeed, the constraints in Equation 2.3 are equivalent to $\forall i \in [1, n], d_i^t - s_i^t - \xi_i \leq x_i^t \leq d_i^t - s_i^t + \xi_i$.

Intuitively, the objective function $\left|\sum_{i=1}^{n} \frac{x_i^t}{\eta_i} - S^t\right|$ can be minimized to 0 if the variables $\forall i \in [1, n], x_i^t$ can make $\sum_{i=1}^{n} \frac{x_i^t}{\eta_i} = S^t$ hold. Thus, we have:

Lemma 1. *The optimal solution of the supply and demand balancing problem at time t can be derived as below:*

- *Case (I): if $S^t \geq \sum_{i=1}^{n} \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$, then external supply $\forall i \in [1, n], \bar{x}_i^t = \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$ are optimal.*

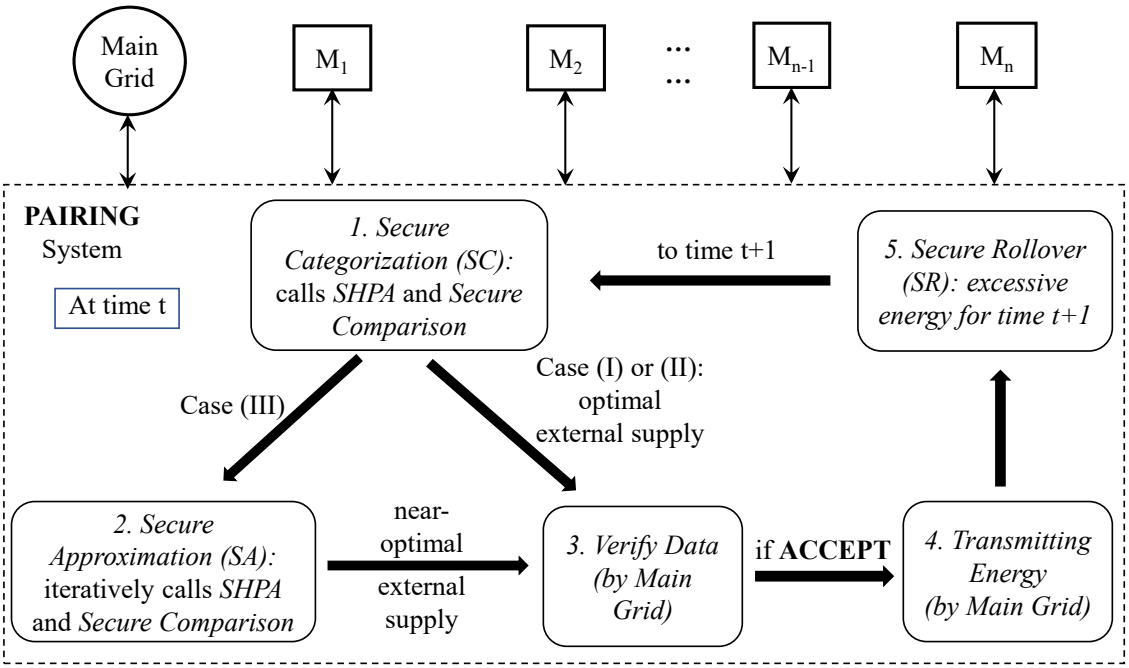


Figure 2.2. PAIRING System

- *Case (II): if* $S^t \leq \sum_{i=1}^{n} \frac{d_i^t - s_i^t - \xi_i}{\eta_i}$*, then external supply* $\forall i \in [1,n], \bar{x}_i^t = \frac{d_i^t - s_i^t - \xi_i}{\eta_i}$ *are optimal.*

- *Case (III): if* $\sum_{i=1}^{n} \frac{d_i^t - s_i^t - \xi_i}{\eta_i} < S^t < \sum_{i=1}^{n} \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$*, then multiple optimal solutions minimize* $\left| \sum_{\forall i=1}^{n} \frac{x_i^t}{\eta_i} - S^t \right|$ *to* 0.

Proof. We prove Case (I) and (II) using contradictions.

In Case (I), if $S^t \geq \sum_{i=1}^{n} \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$, we assume there exists another solution $\forall i \in [1,n], x_i^* \in [\frac{d_i^t - s_i^t - \xi_i}{\eta_i}, \frac{d_i^t - s_i^t + \xi_i}{\eta_i}]$ such that $|\sum_{\forall i=1}^{n} x_i^* - S^t| < |\sum_{\forall i=1}^{n} \bar{x}_i^t - S^t|$ holds. Then, $|\sum_{\forall i=1}^{n} \frac{d_i^t - s_i^t + \xi_i}{\eta_i} - S^t| = S^t - \sum_{i=1}^{n} \frac{d_i^t - s_i^t + \xi_i}{\eta_i} > |\sum_{\forall i=1}^{n} x_i^* - S^t| = S^t - \sum_{\forall i=1}^{n} x_i^*$. Thus, we have $\sum_{\forall i=1}^{n} x_i^* > \sum_{i=1}^{n} \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$, which contradicts with $\forall i \in [1,n], x_i^* \leq \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$. Such contradiction also exists in Case (II).

In Case (III), $\sum_{i=1}^{n} \frac{d_i^t - s_i^t - \xi_i}{\eta_i} < S^t < \sum_{i=1}^{n} \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$ holds. Since $\sum_{i=1}^{n} \frac{d_i^t - s_i^t - \xi_i}{\eta_i} < \sum_{i=1}^{n} x_i^t < \sum_{i=1}^{n} \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$. It is straightforward to see that $\sum_{i=1}^{n} x_i^t$ can equal S^t with

multiple solutions (since all the variables $\forall i \in [1,n], x_i^t$ have the same coefficient in the constraints and the objective function). □

The optimal solutions for Case (I) and (II) are constants while the optimal solution for Case (III) can be securely searched among all the parties (as long as $\sum_{i=1}^{n} \frac{x_i^t}{\eta_i} = S^t$ holds). As shown in Protocol 1, at time t, all parties first securely categorize Case (I), (II) or (III) by securely comparing S^t with $\sum_{i=1}^{n} \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$ and $\sum_{i=1}^{n} \frac{d_i^t - s_i^t - \xi_i}{\eta_i}$, respectively (via the sub-protocol illustrated in Section 2.3.2.2). Then, if Case (I) or (II), per Lemma 1, the optimal solution can be locally derived by each party; if Case (III), all parties securely approximate the optimal solution (via the sub-protocol SA illustrated in Section 2.3.2.3). Since sub-protocols SC and SA securely aggregate data (from all the microgrids, e.g., $\sum_{i=1}^{n} \frac{d_i^t - s_i^t - \xi_i}{\eta_i}$) for comparison with the data (from the main grid G, e.g., S^t), we first propose a sub-protocol for aggregation (denoted as Secure Hierarchically Paired Aggregation (SHPA)) in Section 2.3.2.1.

2.3.2.1 Secure Hierarchically Paired Aggregation (SHPA). As discussed above, SHPA is invoked to aggregate shares of the data from all the parties for *"Two Rounds"* in which both aggregated results will be securely compared later. For instance, while comparing $\sum_{i=1}^{n} \frac{d_i^t - s_i^t - \xi_i}{\eta_i}$ and S^t, each microgrid M_i will generate a random nonce r_i such that $\sum_{i=1}^{n} (\frac{d_i^t - s_i^t - \xi_i}{\eta_i} + r_i)$ (Round A) and $S^t + \sum_{i=1}^{n} r_i$ (Round B) are aggregated for comparison (to securely obtain an *equivalent result* as the original comparison).

SHPA primarily utilizes the homomorphic encryption building block (e.g., Paillier Cryptosystem [24]) for summing up the distributed shares as pairs. Specifically, at the beginning of SHPA, a microgrid (say $M_r, r \in [1,n]$) is randomly picked to utilize its public key pk_r for encryption in Round A. The main grid G's public key pk is used for Round B's encryption.

Furthermore, both Round A and B adopt the *hierarchical pairing* to expedite the secure aggregation via parallelization. It can also mitigate the collusion threats (via iterative random pairing), as validated in Section 2.4.2. In Protocol 2 and Figure 2.3: (1) Round A requests $\lceil \log(n-1) \rceil$ levels of secure sum for pairs of shares (M_r is not involved). SHPA randomly decides the paired parties at each level, and also picks a party out of each pair for aggregation of next level. The aggregation terminates until the last microgrid (at the root of the hierarchy, denoted as "root microgrid") has collected the encrypted sum $Enc_{pk_r}\{\sum_{i=1,i\neq r}^{n}\lfloor\frac{d_i^t-s_i^t-\xi_i}{\eta_i}+r_i\rfloor\}$. Then, the root microgrid sends the encrypted sum to M_r which thus decrypts the ciphertext using its private key sk_r and computes the aggregated value $\sum_{i=1}^{n}\lfloor\frac{d_i^t-s_i^t-\xi_i}{\eta_i}+r_i\rfloor$ with its share; (2) Similarly, Round B requests $\lceil \log(n) \rceil$ levels of secure sum for pairs of shares (M_r is involved). Finally, the main grid G receives $Enc_{pk}(\sum_{i=1}^{n} r_i)$, decrypts the ciphertext and computes $S^t + \sum_{i=1}^{n} r_i$ with its input S^t.

Note that $\frac{d_i^t-s_i^t-\xi_i}{\eta_i}$ can be replaced with other private inputs in SHPA for aggregation, e.g., $\frac{d_i^t-s_i^t+\xi_i}{\eta_i}$.

2.3.2.2 Secure Categorization (SC). At each time $t \in [1, m]$, Secure Categorization (SC) only executes once to securely decide the case of the current supply and demand balancing (per Lemma 1). To decide Case (I), (II) or (III), two secure comparisons should be executed:

- $S^t + \sum_{i=1}^{n} r_i$ (held by G) and $\sum_{i=1}^{n}(\frac{d_i^t-s_i^t+\xi_i}{\eta_i} + r_i)$ (held by a random microgrid M_r)
- $S^t + \sum_{i=1}^{n} r'_i$ (held by G) and $\sum_{i=1}^{n}(\frac{d_i^t-s_i^t-\xi_i}{\eta_i} + r'_i)$ (held by a random microgrid M'_r)

Each of the above comparisons calls sub-protocol SHPA once to get the two random numbers for G and M_r (*we use M_r and M'_r to represent two randomly picked*

1 randomly pick M_r and $\Psi \leftarrow \{M_1, \ldots, M_n\} \setminus M_r$

// Round A (using public key pk_r)

2 $\forall M_i \in \Psi$: $v_i \leftarrow \frac{d_i^t - s_i^t - \xi_i}{\eta_i} + r_i$ (note that v_i can be also initialized as $\frac{d_i^t - s_i^t + \xi_i}{\eta_i} + r_i$, etc.)

3 **while** $sizeof(\Psi) > 1$ **do**

4 randomly pair all the microgrids in Ψ

5 **for** *every pair:* M_i, M_j **do**

6 randomly pick a receiver, w.l.o.g., M_i

7 M_j sends its encrypted share $Enc_{pk_r}(v_j)$ to M_i

8 M_i computes $Enc_{pk_r}(v_i + v_j)$ with its locally encrypted data $Enc_{pk_r}(v_i)$: $Enc_{pk_r}(v_i + v_j) = Enc_{pk_r}(v_i) \otimes Enc_{pk_r}(v_j)$

9 $Enc_{pk_r}(v_i) \leftarrow Enc_{pk_r}(v_i + v_j)$

10 $\Psi \leftarrow \Psi \setminus M_j$

11 **if** $sizeof(\Psi)$ mod $2 = 1$ **then**

12 keep the last unpaired microgrid in Ψ

13 root microgrid sends $Enc_{pk_r}(\sum_{i=1, i \neq r}^{n} v_i)$ to M_r

14 M_r decrypts $Enc_{pk_r}(\sum_{i=1, i \neq r}^{n} v_i)$ with its private key sk_r and computes $\sum_{i=1}^{n}(\frac{d_i^t - s_i^t - \xi_i}{\eta_i} + r_i)$ as the output

// Round B (using $G's$ public key pk)

15 repeat Line 2-12 with $\Psi \leftarrow \{M_1, \ldots, M_n\}$ and initialize $\forall M_i \in \Psi, v_i \leftarrow r_i$

16 root microgrid sends $Enc_{pk}(\sum_{i=1}^{n} r_i)$ to G

17 G decrypts $Enc_{pk}(\sum_{i=1}^{n} r_i)$ with its private key sk and computes $S^t + \sum_{i=1}^{n} r_i$ as the output

Algorithm 2: Secure Hierarchically Paired Agg. (SHPA)

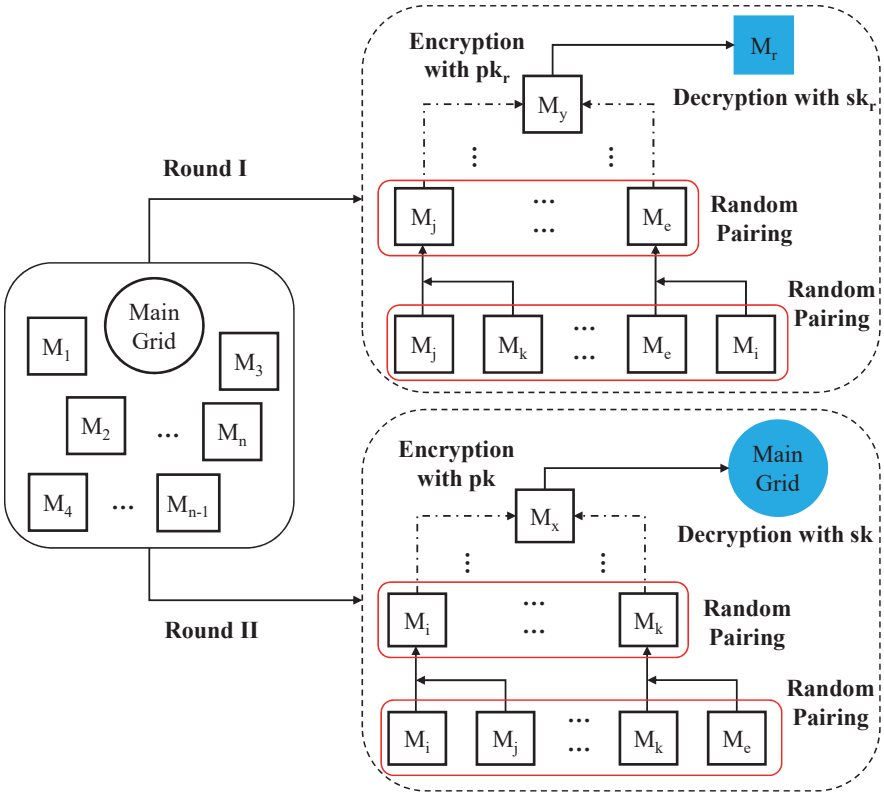


Figure 2.3. Secure Hierarchically Paired Aggregation (SHPA)

microgrids in two different SHPA executions, though they might be the same microgrid). Our PAIRING system integrates FAIRPLAY [25] to securely compare every pair of results held by two different parties. Then, FAIRPLAY will be called twice in the sub-protocol SC (Line 1 and 2 in Protocol 3).

2.3.2.3 Secure Approximation (SA). If Case (III) is identified in sub-protocol SC, another sub-protocol Secure Approximation (SA) will be called to jointly identify a near-optimal solution such that the deviation between the regional supply and demand lies close to 0. SA is established by performing λ-round secure *distributed binary search* by all the microgrids (which also calls SHPA and secure comparison with the main grid G for locating each microgrid's upper/lower bounds of the search). As discussed before, secure distributed binary search can be locally performed to ensure

// At time $t \in [1, m]$

call FAIRPLAY between G and M_r to securely compare $S^t + \sum_{i=1}^{n} r_i$ and $\sum_{i=1}^{n}(\frac{d_i^t - s_i^t + \xi_i}{\eta_i} + r_i)$

call FAIRPLAY between G and M_r' to securely compare $S^t + \sum_{i=1}^{n} r_i'$ and $\sum_{i=1}^{n}(\frac{d_i^t - s_i^t - \xi_i}{\eta_i} + r_i')$

broadcast the two comparison results to all the parties

if $S^t + \sum_{i=1}^{n} r_i \geq \sum_{i=1}^{n}(\frac{d_i^t - s_i^t + \xi_i}{\eta_i} + r_i)$ **then**

    Case (I): return $\bar{x}_i^t = \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$ as the external supply of Microgrid $M_i, i \in [1, n]$ at time t

else

    if $S^t + \sum_{i=1}^{n} r_i' \leq \sum_{i=1}^{n}(\frac{d_i^t - s_i^t - \xi_i}{\eta_i} + r_i')$ **then**

        Case (II): return $\bar{x}_i^t = \frac{d_i^t - s_i^t - \xi_i}{\eta_i}$ as the external supply of Microgrid $M_i, i \in [1, n]$ at time t

    else

        Case (III): return (SA will be called next)

Algorithm 3: Secure Categorization (SC)

strong security and parallelization of computation.

Specifically, each microgrid M_i securely conducts their λ-round binary search for $\bar{x}_i^t$ in range $[\frac{d_i^t - s_i^t - \xi_i}{\eta_i}, \frac{d_i^t - s_i^t + \xi_i}{\eta_i}]$ such that $|\sum_{i=1}^n \frac{\bar{x}_i^t}{\eta_i} - S_i^t|$ (*a global objective*) is minimized to 0. M_i's lower and upper bounds are denoted as lb_i (initialized as $\frac{d_i^t - s_i^t - \xi_i}{\eta_i}$) and ub_i (initialized as $\frac{d_i^t - s_i^t + \xi_i}{\eta_i}$). In each iteration of SA, M_i's lb_i or ub_i is updated as $\frac{lb_i + ub_i}{2}$ (depending on the secure comparison result of two random numbers aggregated in the SHPA): if $\sum_{i=1}^n \frac{lb_i + ub_i}{2} < S_i^t$ (random nonces are securely added to both sides in SHPA), then $\forall i \in [1, n], lb_i \leftarrow \frac{lb_i + ub_i}{2}$ (by M_i); Else $\forall i \in [1, n], ub_i \leftarrow \frac{lb_i + ub_i}{2}$ (by M_i).

After λ iterations, the solution lies close to one of the true optimal solutions at time t (where $|\sum_{\forall i=1}^n \frac{x_i^t}{\eta_i} - S^t| = 0$). Figure 2.4 and Protocol 4 illustrate the details of the sub-protocol SA.

Theorem 1. *Secure Approximation (SA) approximates the optimal solution with a negligible deviation* $\sum_{i=1}^n [\xi_i / 2^{(\lambda-1)}]^2$.

Proof. Sub-protocol SA securely invokes distributed binary search for λ iterations, each of which consists of an SHPA and a secure comparison (among all the microgrids and main grid G). Notice that, M_i's lower and upper bounds of the search are initialized as $lb_i = \frac{d_i^t - s_i^t - \xi_i}{\eta_i}$ and $ub_i = \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$, thus we have $(ub_i - lb_i) = 2\xi_i$. After λ iterations, the range $[lb_i, ub_i]$ (continuous) can be divided to 2^λ ranges with equal-length $\xi_i / 2^{(\lambda-1)}$. Since M_i's share in the optimal solution (denoted as x_i) falls into one of the 2^λ ranges, the deviation between the search output $\bar{x}_i^t$ and x_i is:

$$\begin{aligned}
\Delta_i &= \int_0^{\frac{\xi_i}{2^{(\lambda-1)}}} (\frac{\xi_i}{2^{(\lambda-1)}} - x_i) dx_i + \int_{\frac{\xi_i}{2^{(\lambda-1)}}}^{\frac{\xi_i}{2^{(\lambda-2)}}} (x_i - \frac{\xi_i}{2^{(\lambda-1)}}) dx_i \\
&= [\xi_i / 2^{(\lambda-1)}]^2
\end{aligned}$$

In summary, the overall deviation $\sum_{i=1}^{n}[\xi_i/2^{(\lambda-1)}]^2$ converges to 0 quickly as λ increases. □

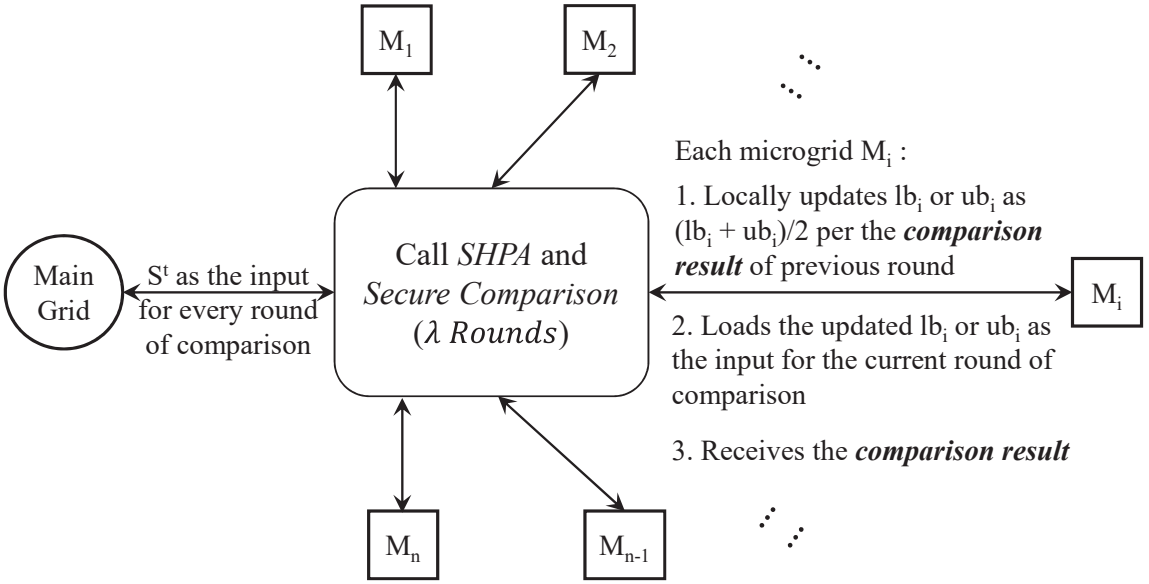


Figure 2.4. Secure Approximation (SA)

2.3.3 Real-time Cryptographic Protocol. The substations (as main grid) or microgrid are generally equipped with a battery that can store excessive energy for balancing the load at different times [10]. As a consequence, if the (local or regional) supply exceeds the demand (still balanced with a tight margin) at time t, the excessive energy should be stored by each microgrid or the main grid.[3]

To achieve this, we design our sub-protocol Secure Rollover (SR) from three perspectives:

- If Case (I) is derived, main grid G stores the excessive energy and updates the regional supply at time $(t+1)$ with the excessive energy. In the meanwhile, each

[3]In case of short time intervals (e.g., 1 minute), the battery capacity of each party is greater than its locally rolled over amount, and energy loss in the storage can be negligible in general.

// At time t, Case (III) occurs in SC

for *iteration* $k \leftarrow 1$ **to** λ **do**

  for *microgrid* $M_i : i \leftarrow 1$ **to** n **do**

    M_i calculates $\frac{lb_i+ub_i}{2}$

  call SHPA to aggregate $\sum_{i=1}^{n}(\frac{lb_i+ub_i}{2}+r_i)$ and $S_i^t+\sum_{i=1}^{n} r_i$

  call FAIRPLAY to compare $\sum_{i=1}^{n}(\frac{lb_i+ub_i}{2}+r_i)$ (held by M_r) and $S_i^t+\sum_{i=1}^{n} r_i$ (held by G)

  broadcast the comparison result to all the parties

  if $\sum_{i=1}^{n}(\frac{lb_i+ub_i}{2}+r_i) < S_i^t+\sum_{i=1}^{n} r_i$ **then**

    for *microgrid* $M_i : i \leftarrow 1$ **to** n **do**

      $lb_i \leftarrow \frac{lb_i+ub_i}{2}$

  else

    $ub_i \leftarrow \frac{lb_i+ub_i}{2}$

return $\bar{x}_i^t = \frac{lb_i+ub_i}{2}$ as the external supply of Microgrid $M_i, i \in [1, n]$ at time t

Algorithm 4: Secure Approximation (SA)

microgrid M_i also stores its the excessive energy ξ_i and updates its local supply at time $(t+1)$ with the excessive energy.

- If Case (II) is derived in SC, no excessive energy to roll over for all the parties.
- If Case (III) is derived in SC, the regional margin is close to 0 (G does not need to roll over) while each microgrid M_i may roll over their excessive energy (if $\sum_{i=1}^{n} \frac{d_i^t - s_i^t}{\eta_i} < S^t$ holds, every microgrid has excessive energy at time t due to nature of binary search) or not (otherwise, no excessive energy at time t).

  Notice that, sub-protocol SA has decided whether $\sum_{i=1}^{n} \frac{d_i^t - s_i^t}{\eta_i} < S^t$ holds or not while calling SHPA and FAIRPLAY for the first time (all the microgrids have known that before calling sub-protocol SR).

In summary, while streamlining the sub-protocols for securely solving the cooperative balancing problems in real time, our PAIRING system locally rolls over each party's excessive energy at time t to time $(t+1)$.

2.4 Security Analysis

2.4.1 Privacy in the Protocol. We first prove that our PAIRING system preserves privacy under the Secure Multiparty Computation (SMC) theory [22, 23]. Privately computing a function in semi-honest model has been defined in [37] – simulating each party's received messages (viz. view) from the protocol in polynomial time.

Given any time period $\forall t \in [1, m]$, the PAIRING system calls sub-protocols Secure Categorization (SC), Secure Approximation (SA) and Secure Rollover (SR) for at most m times, where SC and SA invoke Secure Hierarchically Paired Aggregation (SHPA) for constant times. Then, we first examine the security of the aforementioned four sub-protocols.

Lemma 2. *Secure Hierarchically Paired Aggregation (SHPA) only reveals two public*

```
// At time t, M_i requested x̄_i^t from G
switch Case do
    case (I) do
        at G: S^{t+1} ← S^{t+1} + S^t − Σ_{i=1}^n (d_i^t − s_i^t + ξ_i)/η_i
        at ∀i ∈ [1, n], M_i: s_i^{t+1} ← s_i^{t+1} + ξ_i
    case (II) do
        return
    case (III) do
        if Σ_{i=1}^n (d_i^t − s_i^t)/η_i < S_i^t then
            at ∀i ∈ [1, n], M_i: s_i^{t+1} ← s_i^{t+1} + x̄_i^t + s_i^t − d_i^t
        else
            return
```

Algorithm 5: Secure Rollover (SR)

keys and the IDs (i.e., IP address for communication) of at most $\lceil \log(n^2 - n) \rceil$ *randomly paired microgrids to each party.*

Proof. Recall that sub-protocol SHPA includes two rounds secure aggregation to get two random numbers for comparison: Round I aggregates $\sum_{i=1}^{n}(v_i + r_i)$ using a random microgrid's public key pk_r where v_i can be $\frac{d_i^t - s_i^t + \xi_i}{\eta_i}$, $\frac{d_i^t - s_i^t - \xi_i}{\eta_i}$ or updated lower/upper bounds in sub-protocol SA, and Round II aggregates $S^t + \sum_{i=1}^{n} r_i$ using the main grid G's public key pk. Then, SHPA reveals two public keys (pk_r and pk).

In addition, since participants in Round I are $\Psi = \{M_1, \ldots, M_n\}$ (including M_r) while participants in Round II are main grid G and Ψ, we then analyze their views in both rounds (besides receiving the two public keys).

M_r's view (in both Round I and II). In Round I, M_r only receives a ciphertext

$Enc_{pk_r} \sum_{i=1,i\neq r}^{n}(v_i + r_i)$ from the SHPA, which can be simulated in polynomial time by repeating the encryption to the (random) output with its public key pk_r. To send and receive messages, M_r learns only the ID of another microgrid randomly picked out of $\Psi \setminus M_r$ (the last microgrid at the root of the hierarchy). In Round II, M_r's view is similar to any random microgrid $\forall M_i \in \Psi \setminus M_r$, which is discussed as below.

$\forall i \in [1, n], i \neq r, M_i$'s view (in both Round I and II). In Round I, M_i will be paired for at most $\lceil \log(n-1) \rceil$ times with different microgrids out of $\Psi \setminus M_r$. At each level of the hierarchy, M_i receives a ciphertext from the other paired microgrid (encrypted by pk_r), but cannot decrypt it without the private key sk_r. Similarly, such message can be simulated by executing the encryption. Therefore, M_i learns the IDs of at most $\lceil \log(n-1) \rceil$ microgrids out of $\Psi \setminus M_r$.

Similarly, in Round II, M_i learns the IDs of at most $\lceil \log(n) \rceil$ microgrids out of Ψ (including M_r). Therefore, in both rounds, SHPA reveals the IDs of at most $\lceil \log(n-1) + \log(n) \rceil = \lceil \log(n^2 - n) \rceil$ different randomly paired microgrids to each party M_i (paired with M_i at different levels).

G's view (only in Round II). G only receives the ciphertext of a random number $Enc_{pk}(\sum_{i=1}^{n} r_i)$ and can decrypt it to learn the random number $\sum_{i=1}^{n} r_i$. The random number $\sum_{i=1}^{n} r_i$ (denoted by ϕ) can be polynomially simulated by generating a random number from the uniform probability distribution over $\mathcal{F}$ (note that the random numbers are scaled to fixed precision over a closed field, enabling such a selection). Thus, $Prob[\sum_{i=1}^{n} r_i \; is \; simulated] = \frac{1}{\mathcal{F}}$. □

Note that both the public keys and participants' IDs are not generally considered as private information since secure communication protocols may need IDs (i.e., IP addresses) of the participants for communication. If necessary, we can also implement an anonymized network to hide such IDs using existing tools, e.g., Tor [38]. Thus, we

will consider that SHPA does not disclose private information in this paper.

Lemma 3. *Secure Categorization (SC) securely compares two pairs of numbers, revealing only the results.*

Proof. Sub-protocol SC calls two times SHPA and two times secure comparisons (for comparing $\sum_{i=1}^{n}(\frac{d_i^t - s_i^t + \xi_i}{\eta_i} + r_i)$ and $\sum_{i=1}^{n}(\frac{d_i^t - s_i^t - \xi_i}{\eta_i} + r_i)$ with $S^t + \sum_{i=1}^{n} r_i$, respectively). Besides executing the sub-protocol SHPA, SC calls two times FAIRPLAY [25] (for secure comparison) which outputs a pair of comparison results (as 0 or 1 to M_r and G, which then broadcast them to all the parties). The security of this sub-protocol can be proven as the composition of two secure comparisons [25] with garbled circuits.

Essentially, two pairs of comparison results $\{0,1\} \times \{0,1\}$ can be simulated in polynomial time with 1/4 probability for each. This completes the proof. □

Lemma 4. *Secure Approximation (SA) does not reveal any private information.*

Proof. Sub-protocol SA calls λ times SHPA and λ times secure comparisons (via FAIRPLAY) among G and all the n microgrids, and then each microgrids locally approximates the global optimal solution simultaneously by updating the lower or upper bounds (no message exchange before the next iteration). Since SHPA does not reveal any private information, we only look at the results of λ times secure comparisons. Each party (G and $\forall i \in [1,n], M_i$) receives a sequence of λ comparison results $\in \{0,1\}$. We can build a simulator for each party to locally run binary research on its own range (its input of the protocol) to look for $\bar{x}_i^t$ (its output of the protocol). Clearly, a sequence of λ comparison results $\in \{0,1\}$ can be simulated in polynomial time (actually linear time).

In summary, per the Composition theory [39], sub-protocol SA does not reveal any private information to all the parties (similar to SHPA, each party can only learn

the public keys and the IDs of some paired peers for communication). □

Theorem 2. PAIRING *system does not reveal any private information against passive adversaries.*

Proof. Since PAIRING system runs continuously, we examine the protocol security over any period $t \in [1, m]$. In any m time slots, it calls m times SC (including $2m$ times SHPA and $2m$ times secure comparison), and possibly m times SA (if Case (III) in SC; including $m\lambda$ times SHPA, $m\lambda$ times secure comparison) and m times Secure Rollover (SR).

Per Lemmas 2, 3 and 4, sub-protocols SC and SA (including SHPA) do not disclose private information (except the public keys and microgrid IDs). Since sub-protocol SR only involves local computation (rolling over local energy and updating local supply for the next time slot), we conclude that our PAIRING system does not reveal any private information while running in real time [4] (per the Composition Theorem [37]). □

2.4.2 Mitigating Collusion Attacks. We now analyze the security/privacy of our protocol against colluding main grid and microgrids. In the collusion attacks, we assume colluding parties do not corrupt the protocol, e.g., tampering with messages.

At time t, our PAIRING system calls one time SC, possibly λ times SA (if Case (III)), and one time SR (right after time t). Recall that in every SHPA and secure comparison of our PAIRING system, each microgrid M_i locally generates a new private random nonce r_i to be securely aggregated with each of the two true numbers

[4] Amounts of energy $\forall i \in [1, n], \bar{x}_i^t$ (G receives them from each microgrid at time t) are the outputs of the protocol. Notice that, in Case (II) or (III), even if G can learn that $\frac{d_i^t - s_i^t + \xi_i}{\eta_i}$ or $\frac{d_i^t - s_i^t - \xi_i}{\eta_i}$ as M_i's output (for transmitting energy to M_i at time t), it cannot disaggregate d_i^t, s_i^t and ξ_i from the result.

for secure comparison (eventually decrypted and held by M_r and the main grid G, respectively). Specifically, in such two rounds of aggregation, some microgrids and the main grid can share all their information via collusion to infer other microgrids' local information, e.g., local demand d_i^t, supply s_i^t and the nonce r_i.

Our protocol can mitigate such collusion threats. We simulate such collusion attacks in our deployed PAIRING system. In every attack scenario, we select ℓ microgrids to share all their information with each other and the main grid G (e.g., their local data, private keys, random nonces). Then, all the $(\ell + 1)$ colluding parties try to infer other $(n - \ell)$ microgrids' information. If any local information of another microgrid (e.g., local demand, supply, random nonce) can be inferred, then we consider such collusion attack as a "successful attack". Then, we plot the *"Confidence of Collusion Attacks"* in Figure 2.5 (given $n = 150$ and 300, ℓ varies from 1 to 75; the confidence is calculated based on averaging the results of simulating any single attack for 20 times). It shows that the confidence of collusion attacks is below 8% even if 50% of microgrids and the main grid are colluding with each other.

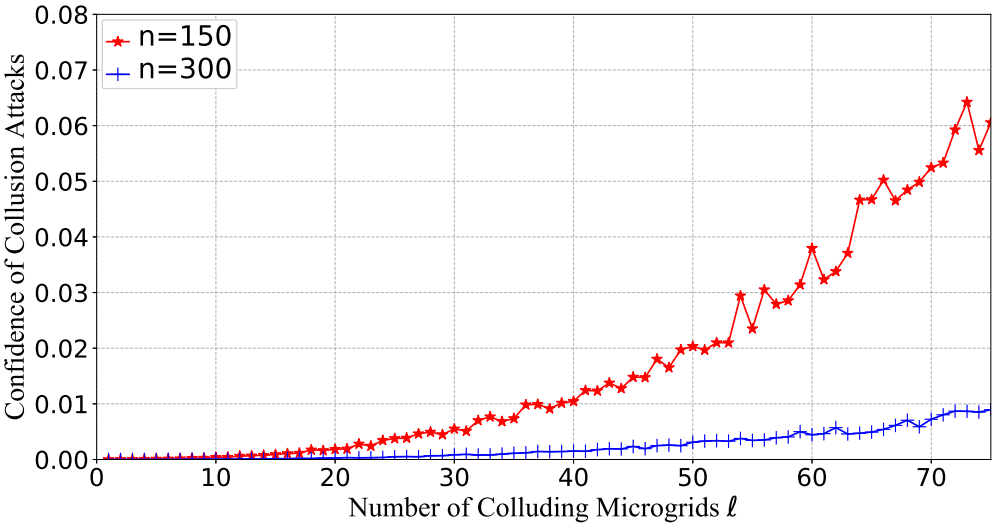


Figure 2.5. Confidence of Collusion Attack vs. ℓ

2.4.3 Verifiability. At time t, all the microgrids call SC and possibly SA (if Case (III) in SC), both of which include SHPA and secure comparison (via FAIRPLAY [25]). Possible data integrity attacks [27, 28] in the protocol are illustrated as follows.

I. Any M_i modifies its received cipthertext in SHPA.

II. In SC and SA (any round of SHPA and/or secure comparison), the randomly picked microgrid M_r can fake $\sum_{i=1}^{n}(v_i+r_i)$ so as to further minimize the difference between its local supply and demand to 0 (by tampering with the comparison result, the output case in SC, and/or the result in SA).

III. Any M_i can intentionally inject a false random nonce r_i (e.g., extremely large or small; similar to false data injection attack [28]) in one of Round I and II to modify the comparison result (and know the result beforehand). The modified comparison results may lead to an inaccurate external supply of each microgrid: $\bar{x}_i^t$.

IV. Any M_i can directly tamper with its external supply $\bar{x}_i^t$ to pursue its local optimum $|\frac{\bar{x}_i^t}{\eta_i} + s_i^t - d_i^t| = 0$.

Attack I can be detected when M_r or G decrypts the ciphertext of the aggregated result – if modified, decryption may not work, then the attack can be directly detected. Attacks II-IV can be summarized as *tampering with any microgrid's $\bar{x}_i^t$ in different phases of the protocol and by different parties (since all of them lead to the same outcome).* Thanks to the inherent constraints on power supply, our protocol can effectively detect such attacks at any time t.

Case (I) and (II) in sub-protocol SC. We have $\forall i \in [1, n]$, $\bar{x}_i^t = \frac{d_i^t - s_i^t + \xi_i}{\eta_i}$ and $\frac{d_i^t - s_i^t - \xi_i}{\eta_i}$, respectively. To ensure integrity verification, at the beginning of each time slot t, our protocol requests each microgrid $M_i, i \in [1, n]$ to generate two hash values

$h(x)$ and $h(x)'$ (e.g., MD5 or SHA-256/512 [40]) for $\frac{d_i^t - s_i^t + \xi_i}{\eta_i}$ and $\frac{d_i^t - s_i^t - \xi_i}{\eta_i}$ (as the checksums), and then send $h(x)$ and $h(x)'$ to main grid G (before calling SC).

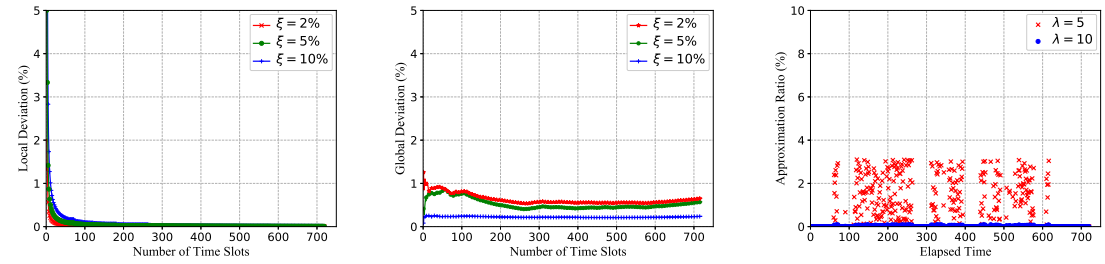
After receiving $h(x)$ and $h(x)'$ before calling SC, main grid G generates another hash value $h(\bar{x})$ (using the same MD5 or SHA-256/512) for the output $\bar{x}_i^t$ (received from M_i) to verify the integrity of the outputs for the entire time slot t. In Case (I), **ACCEPT** if $h(\bar{x}) \equiv h(x)$; otherwise, **REJECT**. Similarly, In Case (II), **ACCEPT** if $h(\bar{x}) \equiv h(x)'$; otherwise, **REJECT**. Due to the non-invertible property of the hash function, if SC returns Case (III) at time t, then main grid cannot reconstruct $\frac{d_i^t - s_i^t + \xi_i}{\eta_i}$ and $\frac{d_i^t - s_i^t - \xi_i}{\eta_i}$ from $h(x)$ and $h(x)'$.

Case (III) in sub-protocol SC. We have $\sum_{i=1}^{n}(\bar{x}_i^t) \approx S^t$. Once $\bar{x}_i^t$ is faked while executing the protocol (without collusion), main grid G can explicitly detect it with a high probability by checking $|\sum_{i=1}^{n} \frac{\bar{x}_i^t}{\eta_i} - S^t| \stackrel{?}{\approx} 0$ (if $\sum_{i=1}^{n} \frac{\bar{x}_i^t}{\eta_i}$ is close to S^t in the approximation).

- If no microgrid's $\bar{x}_i^t$ is modified, then return **ACCEPT**.
- If only one microgrid's $\bar{x}_i^t$ is modified, we thus have $|\sum_{i=1}^{n} \frac{\bar{x}_i^t}{\eta_i} - S^t| \not\approx 0$ (resulting in a detectable deviation). Then, return **REJECT** with detected false data.
- If more than one microgrid's $\bar{x}_i^t$ is modified, the probability of $|\sum_{i=1}^{n} \frac{\bar{x}_i^t}{\eta_i} - S^t| \not\approx 0$ is extremely low (if not all the adversaries collude with each other). Then, return **REJECT** with detected false data.

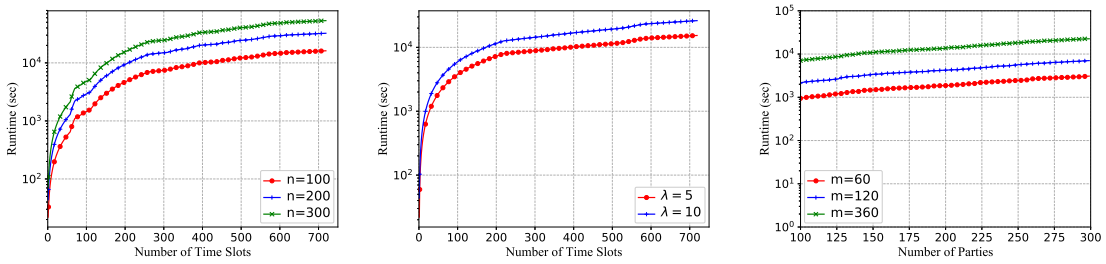
As shown above, it is straightforward to prove *completeness*, *soundness*, and *zero-knowledge* [41] for the verifiability of our protocol against malicious microgrids. Thus, data integrity attacks can be greatly mitigated in our PAIRING system.

2.5 Experimental Evaluation



(a) Local Deviation vs. m (b) Global Deviation vs. m (c) μ at Time 1 to 720

Figure 2.6. Accuracy Evaluation. (a): $(n = 100, \lambda = 5)$. (b): $(n = 100, \lambda = 5)$. (c): $(n = 100, \xi = 2\%)$.



(a) Runtime vs. m (b) Runtime vs. m (c) Runtime vs. n

Figure 2.7. Computational Performance Evaluation (no idle time for minute-level inputs). (a): $(\lambda = 5, \xi = 2\%, 2048\text{-bit})$. (b):($n = 100, \xi = 2\%, 2048\text{-bit}$). (c): $(\lambda = 5, \xi = 2\%, 2048\text{-bit})$.

2.5.1 System Implementation. Our system is deployed on the NSF CloudLab platform (see https://docs.cloudlab.com/) at the University of Utah. Each server has eight 64-bit ARMv8 cores with 2.4 GHZ, 64GB memory and 120GB of flash storage. The OS is Ubuntu:16.04. We leverage Docker (https://docs.docker.com/) to start a container for each party (both main grid and microgrids). We created the image for container based on the raw image of Ubuntu 16.04 by integrating all the environments required by the system (e.g., JRE and JDK), and source codes.

We also integrate OpenDSS (https://sourceforge.net/projects/electricdss/files/) by revising IEEE-123 bus (https://www.xendee.com/home/testcase123node) into our system, which is an electrical power distribution system simulator for real-world

smart grid simulation. OpenDSS collects results from all parties (each of which has a counterpart in OpenDSS), calculates the optimal power distribution plan, and simulate the power flow with practical operational constraints in power system. Note that the energy transmission efficiency η_i (extremely close to 1 in regional supply) are also simulated on the IEEE-123 bus, provided by OpenDSS. After outputs are generated from the protocol (requested energy amounts to main grid), the power flow from main grid to all the microgrids can be simulated in OpenDSS.

2.5.2 Experimental Setup. We conducted experiments for our system evaluations on 300 real microgrids' power generation (via solar panels) and consumption (aka. the demand load) data over a period of 24 hours (which is available at UMass Trace Repository http://traces.cs.umass.edu/index.php/Smart/Smart). We start 301 containers for 300 microgrids and the main grid, and mount their time series input datasets into each container where the power grid topology is shown in Figure 2.9. According to our protocol, we tune the following factors in our experiments.

1. The number of microgrids $n \in [100, 300]$.
2. The number of time slots $m \in [1, 720]$: from 7:00AM to 7:00PM (which covers most of peak times every day).
3. The number of iterations in sub-protocol SA $\lambda \in [5, 10]$.
4. The key size: 512-bit, 1024-bit, and 2048-bit.
5. The balancing margin as ratios: $\xi \in [2\%, 10\%]$.

2.5.3 Accuracy Evaluation. We first evaluate the accuracy of the results returned by our PAIRING system. Before demonstrating the results, we define the following three metrics:

Definition 1 (Global Deviation)**.** *Given the (near) optimal solutions for m time slots returned by* PAIRING *as $\forall t \in [1, m], \forall i \in [1, n], \bar{x}_i^t$, global deviation is defined as*

$$G\text{-}Dev = \frac{\sum_{t=1}^{m} |\sum_{i=1}^{n} \frac{\bar{x}_i^t}{\eta_i} - S^t|}{\sum_{t=1}^{m} S^t} \tag{2.4}$$

Definition 2 (Local Deviation)**.** *Given the (near) optimal solutions for m time slots returned by* PAIRING *as $\forall t \in [1, m], \forall i \in [1, n], \bar{x}_i^t$, local deviation is defined as*

$$L\text{-}Dev = \frac{\sum_{t=1}^{m} \sum_{i=1}^{n} |\bar{x}_i^t + s_i^t - d_i^t|}{\sum_{t=1}^{m} \sum_{i=1}^{n} |\bar{x}_i^t + s_i^t|} \tag{2.5}$$

To benchmark our (near) optimal solutions, we solve the NLP problems at time $t \in [1, m]$ to obtain the true optimal solutions without privacy/security consideration. Since our PAIRING system efficiently finds the near-optimal solutions while calling sub-protocol SA, *approximation ratio* is defined to measure the accuracy of our near optimal solutions.

Definition 3 (Approximation Ratio)**.** *Given the (near) optimal solution at time t returned by* PAIRING *as $\forall i \in [1, n]$, $\bar{x}_i^t$, and the corresponding true optimal solution as $\forall i \in [1, n]$, $\hat{x}_i^t$, approximation ratio is defined as*

$$\mu = \frac{\sum_{i=1}^{n} |\hat{x}_i^t - \bar{x}_i^t|}{\sum_{i=1}^{n} |\hat{x}_i^t|} \tag{2.6}$$

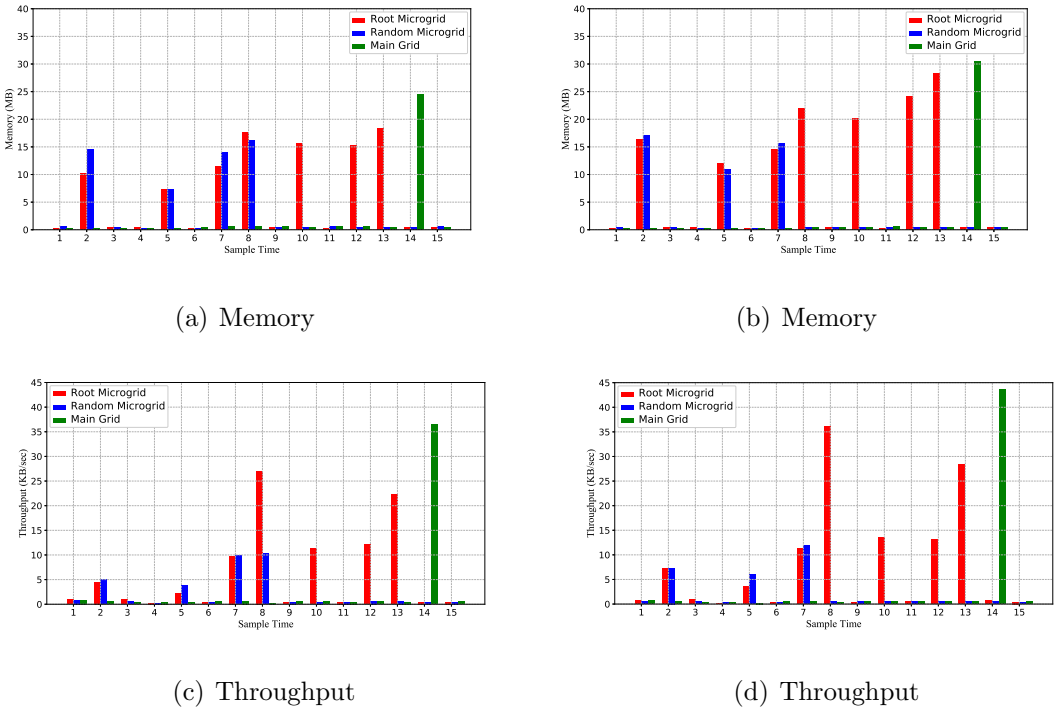
Figure 2.6(a) and 2.6(b) demonstrate the local and global deviation on different ξ and different number of time slots (100 microgrids, $\lambda = 5$ in sub-protocol SA, m

grows to all 720 time slots). Local deviation is low (less than 5%) and quickly converges to 0 as the number of time slots m increases. Also, smaller ξ results in lower local deviation (which balances the supply and demand better in each microgrid). Global deviation is even lower ($< 1\%$ most of the time), and it has an opposite trend on ξ as local deviation – larger ξ would generate lower global deviation. This reflects the fact that more flexibility of microgrids could better contribute to the balancing of regional supply and demand. Figure 2.6(c) shows the evaluated approximation ratio for all the $m = 720$ time slots (x axis presents each time slot). Given different $\lambda = 5$ and 10, all the approximation ratios at different time slots are less than 4% ($\lambda = 5$) and close to 0 ($\lambda = 10$). This result experimentally validated our proof for Theorem 1.

Furthermore, we can anticipate that Case (I) and (II) would result in 0 approximation ratio while Case (III) may have a difference between $\hat{x}_i^t$ and $\bar{x}_i^t$. The approximation ratio in Figure 2.6(c) also reveals the fact that Case (I) starts from the beginning in the morning (the overall supply exceeds the demand load approximately between 7:00AM and 7:30AM) while at the peak times (approximately between 6:00PM and 7:00PM), Case (II) would occur more frequently. Finally, Case (III) indicates the regional supply and demand are well balanced (as approximation ratio would be non-zero but less than 3.08% as $\lambda = 5$ and 0.087% as $\lambda = 10$).

2.5.4 Computational Performance Evaluation. We evaluate the computational cost of our protocol among $n = 100$ to 300 different microgrids, using $\lambda = 5$ and 10, as well as three different key lengths (512/1024/2048-bit), with different λ (5 and 10). Note that the runtime evaluation is performed by skipping the idle time for minute-level inputs (note that our protocol can finish the secure computation in the time interval between two coming inputs, which may result in some idle time and negligible latency).

Indeed, balancing involving sub-protocol SA (Case (III)) would require more



(a) Memory

(b) Memory

(c) Throughput

(d) Throughput

Figure 2.8. System Evaluation for Building Blocks – SHPA and Secure Comparison (via Fairplay [25]). (a) and (b): ($n = 100$, $\lambda = 5$, $\xi = 2\%$, 1024-bit key). (c) and (d), 2048-bit key.

computational cost by calling λ times SHPA and secure comparison. The average runtime for balancing involving sub-protocol SA among 100 microgrids at a single time slot is 23.6s (2048-bit key), and among 200 microgrids is 41.0s (2048-bit key). As n grows to 300, the average runtime is 71.6s (2048-bit key). If sub-protocol SA is not called (Case (I) and (II)), the runtime is *much less*. This indicates that our PAIRING system can handle minute-level real-time energy input data (with negligible latency), which can be sufficiently efficient for the current smart grid deployment (e.g., 10 or 15 minutes per time slot).

Figure 2.7 demonstrates the cumulative runtime with m different time slots (2048-bit key) in 12 hours (from 7:00AM to 7:00PM). Intuitively, larger λ, m and n requires more total computational costs. Table 2.2 shows the computational performance on different key size (512/1024/2048-bit) – the average runtime in each time

slot as m grows. Given the same number of time slots m and different key sizes, we observe that key sizes for encryption/decryption do not influence the runtime too much (since encryption/decryption are independently executed in the docker container in parallel, the one-time encryption/decryption in each round aggregation can be computed during idle time and considered as "offline cost").

Table 2.2. Average Runtime (sec) over m Time Slots ($n = 100$, $\xi = 2\%$, $\lambda = 5$)

m	240	300	360	420	480	540	600	660	720
512-bit	21.6	20.3	20.7	21.5	20.4	20.3	21.2	20.6	20.1
1024-bit	22.2	21.7	22.8	22.7	21.8	21.5	21.7	21.0	20.6
2048-bit	24.3	23.6	23.0	23.2	22.9	23.1	23.0	22.2	21.5

2.5.5 System Evaluation. To investigate the performance of our PAIRING system, we have also evaluated the docker container memory, throughput, and bandwidth at different times when executing the protocol among the 100 microgrids with different key size 1024-bit and 2048-bit (note that PAIRING system has *negligible latency* as the number of microgrids n grows to 300).

Docker container is lightweight in initialization, then the most memory-consuming part of the protocol would be secure computation (e.g., encryption after each pairing, decryption, and invoking Fairplay). Moreover, since sub-protocol SHPA and secure comparison (via Fairplay [25]) are the most frequently called (and also expensive) building block, we capture the memory, throughput and bandwidth of different parties for the lifecycle of a pair of SHPA and secure comparison. Recall that SHPA includes two rounds secure aggregation in which the role and performance of a randomly selected microgrid M_r in Round I is similar to main grid G in Round II. Therefore, we plot the evaluation results for three representative parties' containers in Round

II (which involves main grid G and performs similar to Round I) and Fairplay at 15 different sample time slots: (1) main grid G, (2) root microgrid M_i, and (3) a randomly selected microgrid M_j (other than M_i, denoted as "random microgrid"). Figure 2.8 demonstrates the memory and throughput for 1024-bit and 2048-bit keys, respectively. As seen from the figures, the required memory and throughput are very small, even for peak times.

More specifically, we denote 15 sample time slots as $T_1, T_2, \ldots, T_{15}$. The allocated memory for main grid G is only high at T_{14} (for decryption and executing Fairplay with M_r). The throughput for G is also high at T_{14} (for receiving the ciphertext from the root microgrid M_i). In addition, the root microgrid M_i will participate in $\lceil \log(100) \rceil = 7$ rounds pairing in which it receives ciphertext from other parties and perform multiplication. As shown in Figure 2.8, its allocated memory and throughput are high and relatively fluctuated at the peak times, e.g., $T_2, T_5, T_7, T_8, T_{10}, T_{12}$ and T_{13} for 1024-bit key size. Similarly, another random microgrid M_j only participated in 4 rounds pairing (when using 1024-bit key, as shown in Figure 2.8(a) and 2.8(c)) which correspond to 4 peak times for both memory and throughput. Comparing Figure 2.8(a) and 2.8(b) (and Figure 2.8(c) and 2.8(b)), we discover that the allocated memory and throughput for each container does not change much per different key size. Notice that, both root microgrid and random microgrid are very likely to be different in two experiments with different key sizes, then the random microgrid might be involved in different rounds of pairing in two experiments (e.g., 4 for 1024-bit and 3 for 2048-bit).

In addition, Table 2.3 shows the average bandwidth over m time slots (of all the parties). With such minor bandwidth consumption, our PAIRING system can be deployed in most of the networking environments. Finally, we demonstrate a sample power flow example after executing the protocol with OpenDSS on IEEE-123 bus

Table 2.3. Average Bandwidth (MB) over m Time Slots ($n = 100$, $\xi = 2\%$, $\lambda = 5$)

m	240	300	360	420	480	540	600	660	720
512-bit	0.73	0.65	0.64	0.68	0.72	0.67	0.68	0.65	0.66
1024-bit	1.29	1.24	1.18	1.13	1.28	1.19	1.28	1.16	1.11
2048-bit	2.37	2.12	2.38	2.43	2.40	2.37	2.25	2.01	1.92

system in Figure 2.9.

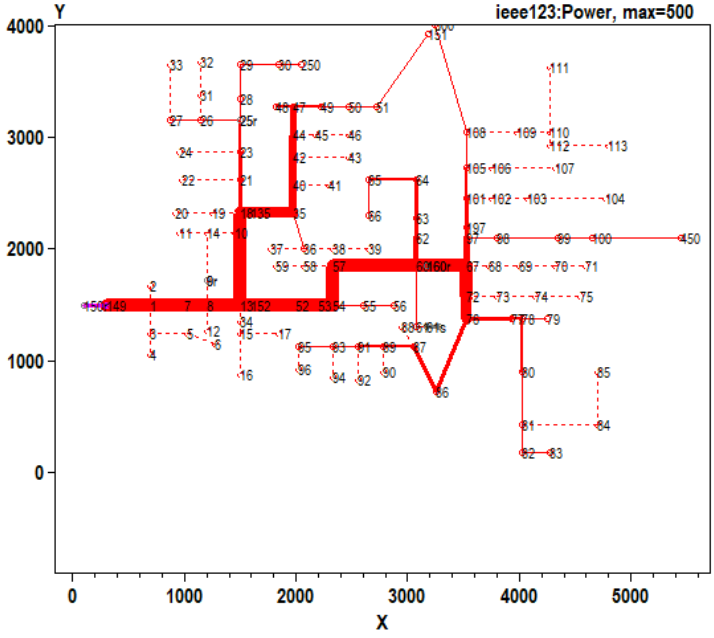


Figure 2.9. Topology of 100 microgrids (based on IEEE-123 bus) and a sample power flow from the main grid to different microgrids. ID "150" represents the main grid (substation) while the remaining IDs represent either microgrids or devices such as transformers. X and Y axes are defined as coordinates. The line thickness is proportional to power relative to a maximum scale of 500kW.

2.6 Related Work

Secure Computation. Secure Multiparty Computation (SMC) [22, 23] has significantly advanced the development of privacy preserving collaborative computation

Table 2.4. Priavcy Preserving Cooperation among Microgrids

Property	Privacy	Collusion Mitigat.	Verifi.	Real Time	System Impl.
Rottondi et al. [21]	✓	✗	✗	✗	✓
Hong et al. [42]	✓	✓	✗	✗	✗
Wang et al. [43]	✓	✗	✗	✗	✗
Zhu et al. [44]	✓	✗	✗	✗	✓
PAIRING	✓	✓	✓	✓	✓

among multiple parties. Specifically, SMC ensures that many functions can be securely computed with private inputs via garbled circuits, such that all parties can only learn the output or their shares of the output. If extending the function to solve complex problems (e.g., data mining [45]), novel secure communication protocols are generally designed by composing the cryptographic building blocks with corresponding security/privacy analysis [22]. Besides the semi-honest model, many existing works have been proposed against malicious adversaries [46, 47]. Recently, secure computation has been leveraged to design privacy preserving systems in different contexts, e.g., location-based services [48], medical data analysis [49] and smart grid [7]. As far as we know, we take the first step to design a system for entities on the power grid to privately cooperate with each other (for improving the grid performance).

Smart Grid Privacy. In literature, mitigating privacy risks in smart grid systems primarily focuses on the protection of metering data which is the consumer's fine-grained meter readings [50]. Researchers have developed various of techniques to resolve the privacy issues for smart meters [12, 18]. For instance, Ács and Castelluccia [12] developed a differentially private smart metering scheme which allows power suppliers to periodically collect data from smart meters and compute aggregated statistics

with rigorous privacy guarantees. Rottondi et al. [18] presented a privacy preserving infrastructure along with a multiparty communication protocol (based on applied cryptography) which allows utilities and data consumers to collect measurement data by securely aggregating smart meters. Moreover, Rottondi et al. [21] proposed a distributed perturbation technique via Gaussian Noise to aggregate user's data to achieve the demand side management. However, all of these may cause the latency or the high utility loss because of the centralized setting or the obfuscated data input. Renewable energy sources like batteries can also be utilized to hide the load/metering information of individual households, which are studied in [51, 52]. Energy harvesting is also an effective solution to increase smart meter privacy [52] via diversifying the energy source. We list the differences of our work from previous works in Table 2.4.

Furthermore, privacy preserving schemes are also proposed for applications functioned by metering data analysis [53, 54], including spatial and temporal power consumption [54], load monitoring [55], billing protocols [53], regional statistics [19], dynamic pricing [56]. However, only a few privacy preserving schemes have been presented for microgrids in literature very recently, to facilitate applications such as energy routing [44], energy exchange/sharing [6, 42], and energy scheduling [43]. For instance, Hong et al. [42] proposed a privacy energy sharing scheme among the microgrids while minimizing the energy losses during transmission. To the best of our knowledge, such schemes can neither quantify the privacy risks with formal security/privacy analysis, nor has been implemented as systems on the power grid. We propose and design a novel prototype of system to address this deficiency.

CHAPTER 3
PRIVATE DISTRIBUTED ENERGY TRADING MARKET

3.1 Introduction

Distributed energy resources (DERs) have been increasingly deployed in the smart grid infrastructure to supplement the power supply with renewable energy such as solar and wind. Equipped with DERs, electricity consumers (e.g., small homes with installed solar panels, hospitals and campuses with deployed microgrids) can also be considered as suppliers that have reduced their dependence on the electricity grid [57]. Recently, multi-agent systems in the smart grid [58] have attracted significant interests by considering the smart homes or microgrids as distributed agents [59, 60]. In reality, smart homes or microgrids may generate excessive energy that cannot be consumed immediately during routine operations. A current solution to deal with the excessive energy is to either consume/waste it or sell it to the main grid [61, 62], even if many of the smart homes/microgrids have been equipped with local storage devices. Essentially, from the economic perspective of such multi-agent systems, transmitting excessive energy back to the main grid or storing the energy is not an ideal outcome, compared to involving more consumers (which requests external energy) to receive the excessive electricity and consume them immediately.

To this end, the smart grid begins to incentivize agents with local energy to cooperate with each other, e.g., decentralized power supply restoration [63], energy sharing [64] and three-party energy trading [61]. Inspired by them, we study the *distributed energy trading* problem which enables smart homes or microgrids to sell their excessive energy to other consumers besides selling back to the power market monopoly, the main grid [65, 66]. It will greatly benefit all the agents: (1) sellers can receive more rewards with a trading price generally higher than the price requested by the main grid, (2) buyers can reduce their costs (i.e., electricity bill) with the

trading price generally lower than the retail price of the main grid [67], and (3) interactions/loads between the consumers and the main grid can be reduced to provide better reliability via autonomy [68].

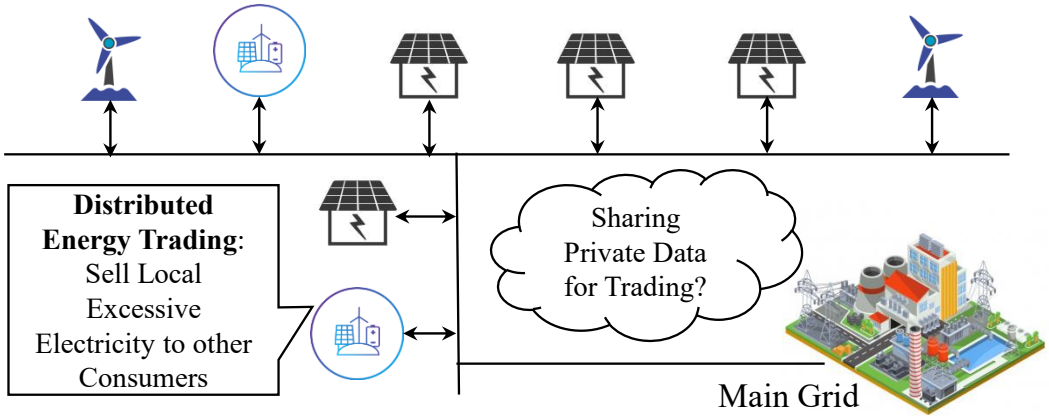


Figure 3.1. Distributed Energy Trading

However, as shown in Figure 3.1, distributed energy trading requests significant amounts of local data from all the agents (e.g., each seller/buyer's local generation and demand load at different times) to compute the *optimal price* and allocate the *energy trading amounts* for all the sellers and buyers [61]. Disclosing such local data for computation would explicitly compromise their privacy. For instance, local generation reveals the generation capacities and time series generation patterns [20], and the local demand load reveals consumption patterns (e.g., which appliance is used at which time) [12, 13].

To address such privacy concerns, we propose a novel privacy preserving distributed energy trading framework, namely "Private Energy Market (PEM)" in which all the agents privately compute the optimal price (ensured by a Nash Equilibrium of a designed Stackelberg game) and allocate pairwise energy trading amounts without disclosing local data. To this end, our PEM framework ensures that all the computations are performed in novel cryptographic protocols under the theory of secure multiparty computation (MPC) [22, 23] which provides provable privacy guarantee.

Thus, the major contributions of this paper are summarized as follows: [4]

1. To our best knowledge, the propose PEM is the first privacy preserving distributed energy trading framework, which enables all the agents on the electric grid to privately compute their optimal trading price (ensured by a Nash Equilibrium) and pairwise trading amounts, as well as complete their pairwise transactions without disclosing their private data (via cryptographic protocols).

2. We model a Stackelberg game [70] in the PEM framework, which ensures *privacy* [22], *individual rationality*, and *incentive compatibility* [71] for all the agents. Theoretical analyses are given to prove all of the three properties.

3. We implement a prototype for the proposed PEM framework with negligible latency in real time. We also conduct substantial experimental evaluations on real datasets to validate the system performance of the PEM.

3.2 Problem Formulation

In this section, we present some preliminaries for the distributed energy trading and the PEM framework. Table 3.1 shows some frequently used notations.

3.2.1 Distributed Energy Trading. We first introduce the background [6,59,61,67] on the power grid, where agents represent the consumers with local generation, e.g., smart homes, and microgrids.

- Energy trading occurs over a fixed length of periods, each of which is referred as a "trading window". All the agents complete their transactions (either selling or buying energy) within each trading window.

[4] This work has been published in IEEE ICDCS [69].

Table 3.1. The Notation Table in Chapter 3

Symbol	Definition
M	the main grid obtains unlimited power supply
H_i	the ith agent $i \in [1, \|\Phi\|]$, Φ: set of agents
g_i^t, l_i^t	H_i's local generation and demand at time window t
b_i^t	H_i's energy amount charging/discharging at t
sn_i^t	H_i's net energy ($sn_i^t = g_i^t - l_i^t - b_i^t$)
Φ_s^t	Seller set $\Phi_s^t = \{\forall H_i \in \Phi, sn_i^t > 0\}$
Φ_b^t	Buyer set $\Phi_b^t = \{\forall H_j \in \Phi, sn_j^t < 0\}$
E_s^t	Market supply $E_s^t = \sum_{H_i \in \Phi_s^t} sn_i^t$
E_b^t	Market demand $E_b^t = \sum_{H_j \in \Phi_b^t} \|sn_j^t\|$
p^t	the optimal price in the PEM (to reach an equilibrium in the game)
pb_g^t	the electricity price offered via the main grid
ps_g^t	the regular retail electricity price from the main grid
$[p_l, p_h]$	acceptable market price range to incentivize agents to join the trading

- Agent can be a buyer in a trading window, and a seller in another trading window, but cannot be both in any trading window (otherwise, its payoff would not be optimal [6]).
- Each seller can decide how much energy it consumes (including charging its battery if available [62]) and how much energy is available in the current trading window.

- We assume that the main grid has unlimited power supply with a higher price than distributed trading [67], and energy is transmitted with a negligible loss.

We denote the main grid as M and the set of agents as Φ (with cardinality $|\Phi|$). For each agent $H_i, i \in [1, |\Phi|]$, we denote its generation (e.g., from solar panels) and demand load in trading window t as g_i^t and l_i^t, respectively. Each agent H_i optionally installs an energy storage device or battery [72] with capacity Cap_i (the maximum energy storage after charging), which can be specified as 0 (if "no battery"). Denoting the energy amount charging into or discharging out of the battery as b_i^t (in trading window t), if charging, we have $b_i^t > 0$; if discharging, we have $b_i^t < 0$. Then, we can define the net energy of H_i as sn_i^t:

$$sn_i^t = g_i^t - l_i^t - b_i^t \tag{3.1}$$

In every trading window t, each agent H_i will be classified as either buyer or seller according to their net energy: (1) if $sn_i^t > 0$, H_i is a seller, (2) if $sn_i^t < 0$, H_i is a buyer, and (3) if $sn_i^t = 0$, H_i will be off market. Then, we formally define $\Phi_s^t = \{\forall H_i \in \Phi, sn_i^t > 0\}$ as the set of sellers and $\Phi_b^t = \{\forall H_j \in \Phi, sn_j^t < 0\}$ as the set of buyers, where the market supply of sellers E_s^t and the market demand of buyers E_b^t can be derived as:

$$E_s^t = \sum_{H_i \in \Phi_s^t} sn_i^t > 0 \text{ and } E_b^t = \sum_{H_j \in \Phi_b^t} |sn_j^t| \tag{3.2}$$

Optimal Trading Price. At the end of every trading window, a seller can store the unsold energy or sell the unsold energy to the main grid [62]. However, the price offered by the main grid (denoted as pb_g^t) is much lower than the regular retail electricity price for purchasing from the grid (denoted as ps_g^t) [67]. In the energy

trading market, while trading energy in window t, all the buyers and sellers will jointly learn an optimal price p^t between pb_g^t and ps_g^t [67] where all the players achieve an equilibrium in a game (with *individual rationality* and *incentive compatibility* [73]).

PEM also sets an acceptable market price range $[p_l, p_h]$ to incentivize the sellers or buyers to join the trading [67] such that the price p^t in the trading window t satisfies:

$$pb_g^t < p_l \leq p^t \leq p_h < ps_g^t \tag{3.3}$$

which is set by the PEM rather than specific agents. If $p^t > ps_g^t$, all the rational buyers will purchase energy directly from the main grid; if $p^t < pb_g^t$, all the rational sellers will sell the energy directly to the main grid. Thus, PEM specifies a reasonable price range. Section 3.3 will illustrate how to derive the optimal price and allocate energy trading amounts in every trading window.

3.2.2 Threat Model. More importantly, our PEM framework addresses the privacy concerns of all the participants (e.g.,. agents with local energy) in the distributed energy trading. Specifically, to realize the energy trading, all the agents $\forall i \in [1, |\Phi|], H_i$ should share its local private information to a trusted third party so as to compute their optimal price as well as allocating pairwise energy trading amounts. However, such shared local information are sensitive in general [13, 15, 52], e.g., H_i's local energy generation amount, energy consumption amount, battery storage amount, and its utility parameter (which are detailed in Section 3.3).

To tackle the above concerns, we propose the PEM framework (without a trusted third party) based on efficient cryptographic protocols [22, 23] to privately function distributed energy trading without disclosing local information. We define the *threat model* in the distributed energy trading as below:

- We assume semi-honest adversarial model for preserving the privacy in our cryptographic protocols: all the agents are curious to learn private information from each other [22, 39] but do not maliciously corrupt the protocol.
- Besides the semi-honest model, all the agents have the incentive to improve its payoff by cheating on its data.
- All the messages in the framework are assumed to be transmitted in a secure channel.

3.2.3 PEM Framework. To sum up, PEM will provide the following three properties against the adversaries:

- **Privacy**: each seller/buyer's privacy is protected in the PEM with provable privacy guarantee.
- **Individual Rationality**: each seller/buyer has a higher payoff by participating in the PEM.
- **Incentive Compatibility**: each seller/buyer cannot improve its payoff by untruthfully changing its strategy.

Section 3.4 will illustrate the cryptographic protocols for our PEM framework, and Section 3.5 will analyze the privacy/security and incentive compatibility to protect the trading under the threat model defined earlier.

3.3 Distributed Energy Trading

In this section, we first present the distributed trading scheme for PEM without privacy consideration.

3.3.1 Incentive Measurement. We first define two functions to measure the incentives for both sellers and buyers in the trading [61]. The utility function measures the payoff received by each seller while the cost function measures how much each buyer pays.

Seller's Utility Function [72, 74] is defined to quantify the total utility of any seller $H_i \in \Phi_s^t$ in trading window t:

$$U_i^t = k_i^t \log(1 + l_i^t + \epsilon_i^t * b_i^t) + p^t * (g_i^t - l_i^t - b_i^t) \tag{3.4}$$

where $k_i^t > 0$ is the load behavior preference parameter of the seller H_i (*either locally consuming more energy or selling them*), p^t is the market price. l_i^t and g_i^t are defined as the load and generation of H_i. For the battery, b_i^t is defined as the energy charging/discharging amount: *charging if positive (as additional load)* and *discharging if negative (as additional supply)*; $\epsilon_i^t \in (0, 1)$ represents the battery loss coefficient, which measure the ratio of battery's contribution amount as load (charging) utility.

Buyer's Cost Function is defined to measure the cost of any buyer $H_j \in \Phi_b^t$ from the energy market and main grid:

$$C_j^t = p^t * x_j^t + ps_g^t * (l_j^t + b_j^t - g_j^t - x_j^t) \tag{3.5}$$

Similarly, l_j^t, g_j^t, and b_j^t denote the buyer's local load, generation, and battery charging/discharging amounts, respectively. Moreover, x_j^t is defined as the energy amount that H_j purchased from the trading market, thus we have $0 < x_j^t \leq l_j^t + b_j^t - g_j^t$.

3.3.2 Stackelberg Game for PEM. To further pursue the cooperation of agents, two coalitions are formed based on each agent's net energy in every trading window (seller coalition and buyer coalition; the agents in two coalitions change over time). In

our PEM framework, the seller coalition sells energy with the total supply while the buyer coalition purchases energy with their total demand, and their shares of energy to sell/buy are allocated proportional to their input shares (as detailed in Section 3.3.4). Such trading mechanism could make the market more stable, and guarantee the payoffs for conservative sellers/buyers who may not want to fully compete with other sellers/buyers.

Stackelberg Game Per the two (utility and cost) functions defined for sellers and buyers, the objectives of two coalitions consist of two aspects: (1) buyers incline to minimize their costs (as a coalition); (2) sellers incline to maximize their utility. To learn the optimal price, we propose a Stackelberg game for seller and buyer coalitions [70].

Specifically, the market supply (from agents) is generally less than market demand (since renewable energy cannot feed all the load in current practice [61]). Therefore, in the Stackelberg game, the buyer coalition is specified as the leader while the seller coalition is defined as the follower (otherwise, sellers will dominate the market). Then, the game $\mathcal{G}$ can be formally defined as:

$$\mathcal{G} = \{\Phi_b^t \cup \Phi_s^t, \{l_i^t\}_{H_i \in \Phi_s^t}, \{U_i^t\}_{H_i \in \Phi_s^t}, p^t, \Gamma^t\} \tag{3.6}$$

with the following components in each trading window t:

- the buyer coalition Φ_b^t is the leader to set up the price while the seller coalition Φ_s^t chooses their strategies as a response to the proposed price.
- $\{l_i^t\}_{H_i \in \Phi_s^t}$ is the set of load profiles of all the sellers (strategies) to maximize their payoffs.
- U_i^t is the utility function of seller H_i.

- p^t is the price proposed by the buyer coalition.
- Γ^t is the total cost for the buyer coalition:

$$\Gamma^t = \sum_{H_j \in \Phi_b^t} C_j^t = p^t * E_s^t + ps_g^t * (E_b^t - E_s^t) \tag{3.7}$$

where E_s^t and E_b^t are the market supply/demand (Eq. 3.2).

Then, the objective of the model is to minimize the total costs of the buyers/leader and to maximize the individual utility function of each seller/follower (such that the seller coalition's total utility is also maximized) by choosing their own strategies. We define the equilibrium as below:

Definition 4. *The set of strategies* $(\{l_i^{t*}\}_{H_i \in \Phi_s^t}, p^{t*})$ *is an equilibrium for the game* $\mathcal{G}$*, if and only if it satisfies:*

$$U_i^t(\{l_i^{t*}\}_{H_i \in \Phi_s^t}, p^{t*}) \geq U_i^t(\{l_k^t, \{l_i^{t*}\}_{H_i \in \Phi_s^t, i \neq k}\}, p^{t*})$$
$$\Gamma^t(\{l_i^{t*}\}_{H_i \in \Phi_s^t}, p^{t*}) \leq \Gamma^t(\{l_i^{t*}\}_{H_i \in \Phi_s^t}, p)$$

where $p_l \leq p^{t*} \leq p_h$.

Therefore, we seek for the equilibrium of this game in which the follower (aka. sellers) derives the best response to the optimal price proposed by the leader (aka. buyers). At this equilibrium, neither the leader nor any follower can increase its payoff via any *unilateral strategic move*. In other words, when the game reaches the equilibrium, the buyers cannot reduce the cost by decreasing the price p^t while the sellers cannot improve their utility by adjusting their strategies on load profiles $\{l_i^t\}_{H_i \in \Phi_s^t}$.

Optimal Price We prove the existence and uniqueness of the equilibrium [61, 70] for $\mathcal{G}$:

Lemma 5. *A unique equilibrium* $(\{l_i^t\}_{H_i \in \Phi_s^t}, p^{t*})$ *exists.*

Proof. First, we get the second derivative of the utility function U_i^t (Eq. 3.4):

$$\frac{\partial^2 U_i^t}{\partial {l_i^t}^2} = \frac{-k_i^t}{(1 + l_i^t + \epsilon_i^t b_i^t)^2} \tag{3.8}$$

which is always less than 0 since $k_i^t > 0$. The utility function is concave with l_i^t. Then given any price p^t, each seller $H_i \in \Phi_s^t$ can only find a unique l_i^t to get its maximum utility. On the contrary, the buyers can also find the optimal price while the sellers specify their load profiles in the Nash Equilibrium. Thus, the equilibrium $(\{l_i^t\}_{H_i \in \Phi_s^t}, p^{t*})$ exists.

Second, to prove the uniqueness of the equilibrium, we need to prove that the optimal price is unique for the minimum cost of the buyer coalition (leader of the game). We first find the optimal load profile for each seller $H_i \in \Phi_s^t$: l_i^t. We then get the first derivative of H_i's utility function (whose value should be 0 for the maximum utility):

$$\frac{\partial U_i^t}{\partial l_i^t} = \frac{k_i^t \epsilon_i^t}{(1 + l_i^t + \epsilon_i^t b_i^t)} - p^t = 0 \tag{3.9}$$

Thus, we get the optimal load profile for seller H_i:

$$l_i^t = \frac{k_i^t \epsilon_i^t}{p^t} - 1 - \epsilon_i^t b_i^t \tag{3.10}$$

Replacing l_i^t in the total cost function Γ^t, then we get the second derivative of Γ^t:

$$\frac{\partial^2 \Gamma^t}{\partial {p^t}^2} = \sum_{H_i \in \Phi_s^t} \frac{2 p s_g^t k_i^t}{(p^t)^3} > 0 \tag{3.11}$$

Then, Γ^t is strictly convex with p^t, which generates a unique optimal price. Thus, equilibrium $(\{l_i^t\}_{H_i \in \Phi_s^t}, p^{t*})$ is unique. This completes the proof. □

To find the optimal price p^{t*} in game $\mathcal{G}$, we calculate the first derivative of Γ^t:

$$\frac{\partial \Gamma^t}{\partial p^t} = \sum_{H_i \in \Phi_s^t} (g_i^t + 1 + \epsilon_i^t b_i^t - b_i^t) - \frac{p s_g^t \sum_{H_i \in \Phi_s^t} k_i^t}{(p^t)^2} = 0 \tag{3.12}$$

Solving Eq. 3.12, we have

$$\widehat{p^t} = \sqrt{\frac{p s_g^t \sum_{H_i \in \Phi_s^t} k_i^t}{\sum_{H_i \in \Phi_s^t} (g_i^t + 1 + \epsilon_i^t b_i^t - b_i^t)}} \tag{3.13}$$

Therefore, we can get the optimal price p^{t*} by integrating Eq. 3.13 and 3.3.

$$p^{t*} = \begin{cases} \widehat{p^t}, & \widehat{p^t} \in [p_l, p_h] \\ p_l, & \widehat{p^t} < p_l \\ p_h, & \widehat{p^t} > p_h \end{cases} \tag{3.14}$$

Replacing p^t in the load profile l_i^t (Eq. 3.10) with p^{t*}, we can get the optimal load profile (strategy) for each seller H_i:

$$l_i^{t*} = \frac{k_i^t \epsilon_i^t}{p^{t*}} - 1 - \epsilon_i^t b_i^t \tag{3.15}$$

Note that if there is no battery installed for the seller, we thus have $b_i^t = 0$.

3.3.3 Trading Scheme in an Extreme Market. If the market supply in the PEM is greater than or equal to the market demand (this rarely occurs in the current smart grid infrastructure, "extreme market"), to maintain a robust market, the market electricity price should be set to the lower bound p_l which is still greater than the

price pb_g^t offered by the main grid. Different from the general market case, the sellers also maximize their utilities by selling the remaining energy to the main grid and the buyer coalition will buy the electricity for all its demand from the market (to minimize their costs).

3.3.4 Energy Distribution and Payment. Considering $E_s^t < E_b^t$ as the general market and $E_s^t \geq E_b^t$ as the extreme market, our PEM framework allocates trading amount for each pair of buyer and seller based on the demand (general market) or supply ratio (extreme market) out of the total market supply and demand to ensure fairness of distribution. We now discuss the allocation strategies for the two markets.

1. **General Market:** the optimal price p^{t*} is proposed by the buyer coalition in the Stackelberg Game and all the market supply should be sold to the buyer coalition with price p^{t*}. In the buyer coalition, the amount of electricity should be allocated in terms of their demand ratio out of the total demand E_b^t. Then, each buyer $H_j \in \Phi_b^t$ requests energy with the amount $e_{ij} = sn_i^t * \frac{|sn_j^t|}{E_b^t}$ from seller $H_i \in \Phi_s^t$, and pays $m_{ji} = p^{t*} e_{ij}$ to seller H_i.

2. **Extreme Market:** the price is directly set as p_l. Similarly, each seller $H_i \in \Phi_s^t$ sells the amount of $e_{ij} = |sn_j^t| * \frac{sn_i^t}{E_s^t}$ to buyer $H_j \in \Phi_b^t$ and receives the payment of $m_{ji} = p_l e_{ij}$ from buyer H_j.

3.4 Cryptographic Protocols

In this section, we present the cryptographic protocols for the distributed energy trading in PEM.

3.4.1 Cryptographic Building Blocks. We adopt homomorphic encryption [24] and garbled circuit [22, 23] as the building blocks to construct our protocols.

Homomorphic Encryption (e.g., Paillier cryptosystem [24]) is a semantically-secure public key encryption to generate the ciphertext of an arithmetic operation between two plaintexts by other operations between their ciphertexts. It has the additional property that given any two encrypted messages $E(A)$ and $E(B)$, we have $E(A + B) = E(A) * E(B)$, where $*$ denotes the multiplication of ciphertexts (in some abelian group).

Garbled Circuit was originally proposed by Yao [22]. It enables two parties to jointly compute a function without disclosing their private inputs where one party creates the garbled circuit and the other party evaluates the circuit to derive the result of the secure computation. Our protocols only incorporate garbled circuit (e.g., the FAIRPLAY system [25]) for realizing some light-weight computations (e.g., secure comparison) instead of the entire trading scheme.

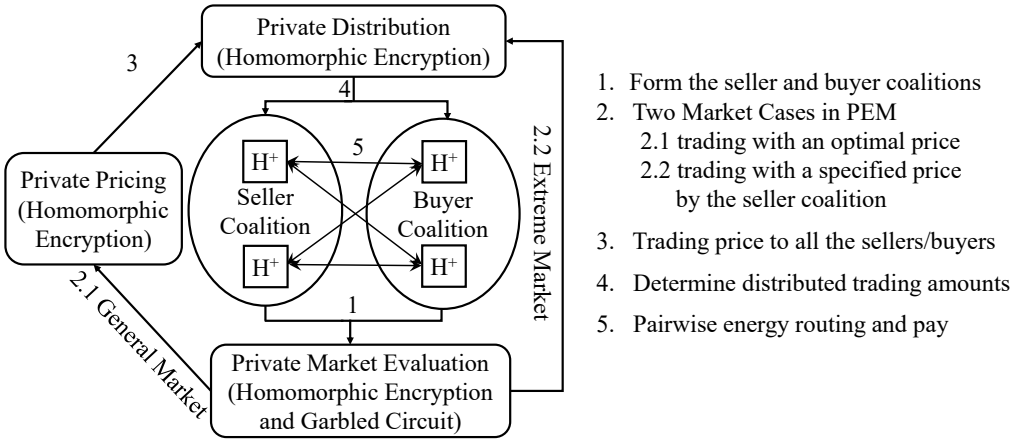


Figure 3.2. Overview of the PEM Framework

3.4.2 Overview of the PEM. As shown in Figure 3.2 and Protocol 6, in the PEM framework, all the agents firstly form the seller and buyer coalitions (Initialization). Then, in Private Market Evaluation, the two coalitions securely evaluate the market. If a general market case is returned, Private Pricing is executed to securely compute the optimal price. For both general and extreme market cases, Private Distribution is

executed to complete the trading.

for *agent* $H_i \in \Phi$ **do**
    H_i generates key pair (pk_i, sk_i), and shares pk_i in Φ
for *each trading window* t **do**
    Initialize seller and buyer coalitions: Φ_s^t, Φ_b^t
    Φ_s^t and Φ_b^t execute **Private Market Evaluation**
    if $E_s^t < E_b^t$ *(general market)* **then**
        Execute **Private Pricing** (Protocol 8): $p^t = p^{t*}$
    else
        Set the current price $p^t = p_l$ (extreme market)
    Φ_s^t and Φ_b^t execute **Private Distribution** (Protocol 9)

Algorithm 6: Private Energy Market (PEM)

3.4.3 Initialization. Since secure computation in the PEM primarily utilizes the Homomorphic encryption (e.g., Paillier Cryptosystem [24]), each seller/buyer locally generates its own public-private key pair and shares all their public keys. At the beginning of each trading window, each agent claims its role as buyer or seller or off the market to form the seller and buyer coalitions. If the seller coalition is empty ($E_s^t = 0$), all the buyers should buy energy from the main grid with the retail electricity price.

3.4.4 Private Market Evaluation. After the initialization, PEM determines the market to be a general or extreme market, where the seller coalition Φ_s^t and buyer coalition Φ_b^t jointly aggregate their private net energy, and then compare the overall supply E_s^t and demand E_b^t. Specifically, there are "two rounds" of aggregations. In the first round, an arbitrary seller H_{r1} will be chosen, then each buyer $H_j \in \Phi_b^t$ encrypts the demand $|sn_j^t|$ plus a random nonce r_j using H_{r1}'s public key pk_{r1}, i.e., $Enc_{pk_{r1}}(|sn_j^t| + r_j)$ for summing up $\sum_{H_j \in \Phi_b^t} Enc_{pk_{r1}}(|sn_j^t| + r_j)$. The ciphertext will be

1 Randomly choose $H_{r1} \in \Phi_s^t$ with key pair (pk_{r1}, sk_{r1})
2 **for** *each* $H_j \in \Phi_b^t$ **do**
3 H_j generates random nonce r_j and initializes $C = 1$
4 H_j computes $C \leftarrow C * Enc_{pk_{r1}}(|sn_j^t| + r_j)$
5 The last agent in Φ_b^t sends C to $\Phi_s^t \setminus H_{r1}$
6 **for** *each* $H_i \in \Phi_s^t$ **do**
7 H_i generates random nonce r_i
8 H_i computes $C \leftarrow C * Enc_{pk_{r1}}(r_i)$
9 The last agent in Φ_s^t sends C to H_{r1}
10 H_{r1} obtains $R_b = \sum_{H_j \in \Phi_b^t}(|sn_j^t| + r_j) + \sum_{H_i \in \Phi_s^t} r_i$ by decrypting the ciphertext with sk_{r1}
11 Randomly choose $H_{r2} \in \Phi_b^t$ with key pair (pk_{r2}, sk_{r2})
12 **Repeat** Lines 2-10 with $H_{r1} \leftarrow H_{r2}$
13 H_{r2} obtains $R_s = \sum_{H_i \in \Phi_s^t}(sn_i^t + r_i) + \sum_{H_j \in \Phi_b^t} r_j$ by decrypting the ciphertext with sk_{r2}
14 H_{r1}, H_{r2} execute secure comparison with input R_b, R_s
15 **if** $R_s < R_b$ **then**
16 **return** general market
17 **else**
18 **return** extreme market

Algorithm 7: Private Market Evaluation

sent to one random seller in the seller coalition except H_{r1}. Similarly, each seller H_i of the seller coalition generates a nonce r_i and encrypts it for summing up the value in the ciphertext. Finally, H_{r1} decrypts the ciphertext to get the aggregated value $R_b = \sum_{H_j \in \Phi_b^t}(|sn_j| + r_j) + \sum_{H_i \in \Phi_s^t} r_i$ with its private key (see Lines 2-10 in Protocol 7). The second round is similar to the first: one random selected buyer H_{r2}'s public

key pk_{r2} is used to aggregate the $sn_i + r_i$ of each seller $H_i \in \Phi_s^t$ and the nonce r_j of each buyer to get $R_s = \sum_{H_i \in \Phi_s^t}(sn_i + r_i) + \sum_{H_j \in \Phi_b^t} r_j$ (see Lines 11-13).

As shown in Protocol 7, neither the chosen seller H_{r1} nor buyer H_{r2} knows the value of E_b^t or E_s^t and they only obtain the aggregated random value (R_b or R_s). Furthermore, Private Market Evaluation securely compares R_b and R_s by H_{r1} and H_{r2} to determine the market case using the garbled circuits (e.g., the FAIRPLAY system [75], see Lines 14-18). Note that the comparison result of R_b and R_s is equivalent to the comparison result of E_b^t or E_s^t since the same sum of random nonces are added to E_b^t and E_s^t to obtain R_b and R_s.

3.4.5 Private Pricing. If general market is returned in Protocol 7, Private Pricing will be executed to find the optimal price p^{t*} of the Stackelberg Equilibrium. Specifically, a seller H_b will be chosen at random to securely aggregate two local values of each seller $H_i \in \Phi_s^t$: (1) k_i^t, and (2) $g_i^t + 1 + \epsilon_i^t b_i^t - b_i^t$ (locally computed). Then H_b derives the optimal price p^{t*} per the Eq. 3.14 in Section 3.3.2 once getting $\widehat{p^t}$, and broadcasts the optimal price p^{t*} (see Lines 8-9 in Protocol 8).

1 Choose randomly $H_b \in \Phi_b^t$ with key pair (pk_b, sk_b)
2 **for** *each* $H_i \in \Phi_s^t$ **do**
3 $\quad \prod_{s=1}^{i} Enc_{pk_b}(k_s^t) \leftarrow \prod_{s=1}^{i-1} Enc_{pk_b}(k_i^t) * Enc_{pk_b}(k_i^t)$
4 The last agent sends $\prod_{i=1}^{|\Phi_s^t|} Enc_{pk_b}(k_i^t)$ to H_b
5 H_b decrypts $\prod_{i=1}^{|\Phi_s^t|} Enc_{pk_b}(k_i^t)$ using sk_b for $\sum_{H_i \in \Phi_s^t} k_i^t$
6 **Repeat** Lines 2-5 with $k_i^t \leftarrow g_i^t + 1 + \epsilon_i^t b_i^t - b_i^t$
7 H_b decrypts the ciphertext to obtain $\sum_{H_i \in \Phi_s^t}(g_i^t + 1 + \epsilon_i^t b_i^t - b_i^t)$
8 H_b calculates: $\widehat{p^t} = \sqrt{\frac{ps_g^t \sum_{H_i \in \Phi_s^t} k_i^t}{\sum_{H_i \in \Phi_s^t}(g_i^t + 1 + \epsilon_i^t b_i^t - b_i^t)}}$
9 H_b derives p^{t*} per the Eq. 3.14 and broadcasts it

Algorithm 8: Private Pricing

3.4.6 Private Distribution. As discussed in Section 3.3.4, the trading amount of electricity between each pair of seller and buyer should be allocated in proportion to its demand/supply ratio out of the market demand/supply in both general and extreme market case. W.l.o.g., we discuss the protocol for the general market (which can be simply extended for the extreme market). For buyer H_j, the allocated amount of electricity from seller H_i should be $e_{ij} = \frac{|sn_j^t|}{E_b^t} * sn_i^t$. Since any buyer may intend to cheat by using a larger demand sn_i^t to increase its share in the allocation (reduce costs with a lower price to buy energy), the market demand *cannot be directly disclosed to the buyers*. The seller coalition cannot get the market demand considering the *privacy* and *fairness*. Since the homomorphic encryption schemes (e.g., Paillier Cryptosystem [24]) only obtain additive and/or multiplicative property (not fully homomorphic [76] to securely compute "division"), we cannot directly adopt homomorphic encryption for privately computing the pairwise allocated amounts using their input ratios.

To address such issue, we transform the ciphertext computation for the "division/ratio" $\frac{|sn_j^t|}{E_b^t}$ in e_{ij}. Specifically, each buyer H_j locally computes $Enc_{pk_s}(\frac{E_b^t}{|sn_j^t|})$ with $\frac{1}{|sn_j^t|}$. Note that $\frac{1}{|sn_j^t|}$ should be multiplied by an integer k to be converted to an integer. Then $Enc_{pk_s}(\frac{E_b^t}{|sn_j^t|})$ and k will be sent to the seller H_s, and H_s decrypts it to get the allocation ratio via $(\frac{E_b^t}{|sn_j^t|})^{-1} = \frac{|sn_j^t|}{E_b^t}$. The only information that the seller H_s knows is the allocation ratio for buyer coalition (while E_b^t and $|sn_j^t|$ are unknown). Thus, the seller H_s can broadcast the allocation ratio in the seller coalition. Finally, each seller H_i calculates the allocated amount of energy e_{ij} and routes the energy to each buyer H_j; H_j pays m_{ji} to H_i (see Lines 10-12 in Protocol 9). Similarly, in an extreme market, the protocol can be implemented by swapping their roles. Figure 3.3 illustrates the major procedures of Private Distribution (note that the random seller H_s is chosen as H_1^+).

3.5 Analysis

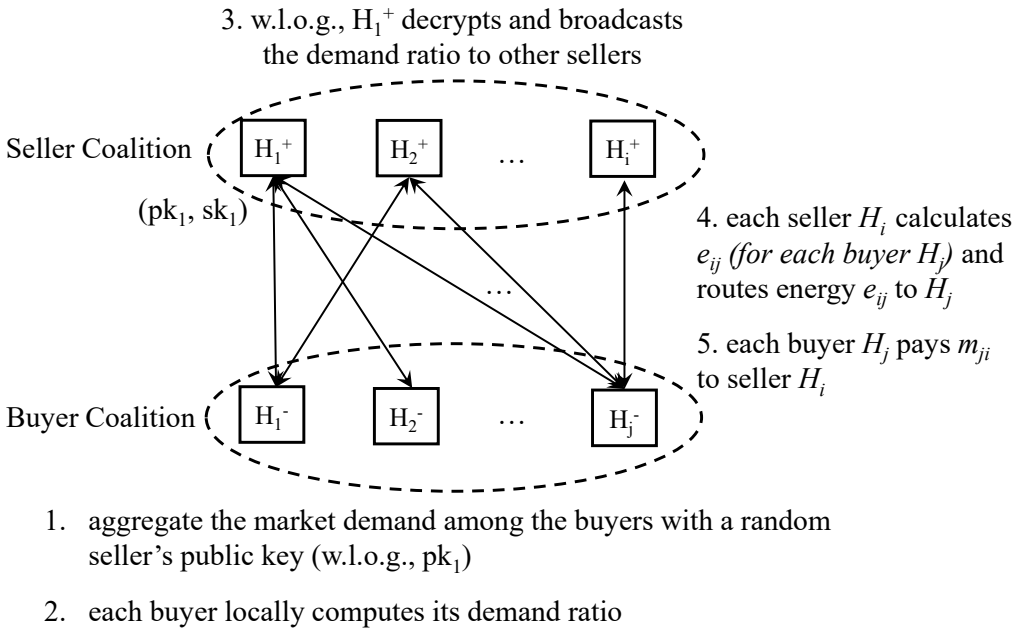


Figure 3.3. Private Distribution for General Market (which can be adapted for extreme market by swapping the roles of two coalitions: each buyer H_j calculates e_{ij} and pays m_{ji}).

In this section, we give theoretical analysis for privacy, incentives, and the complexity in our PEM framework.

3.5.1 Security/Privacy Analysis. We now prove the security/privacy for the protocols in our PEM framework under the theory of secure multiparty computation [22, 23], which requires each party to simulate all its received messages with only its input and output in polynomial time ("*Computational Indistinguishability*") [39]. The PEM framework executes Private Market Evaluation, Private Distribution and possibly Private Pricing in each trading window. Then, we first examine the security of the three protocols and then discuss the composition [39].

Lemma 6. *The Private Market Evaluation (Protocol 7) does not reveal any private information.*

Proof. Three different types of parties are involved in Protocol 7: a randomly selected

1 **if** *general market* **then**

2 Randomly choose $H_s \in \Phi_s^t$ with key pair (pk_s, sk_s)

3 **for** *each* $H_j \in \Phi_b^t$ **do**

4 H_j computes

$$\prod_{s=1}^{j} Enc_{pk_s}(|sn_s^t|) \leftarrow \prod_{s=1}^{j-1} Enc_{pk_s}(|sn_j^t|) * Enc_{pk_s}(|sn_j^t|)$$

5 The last agent broadcasts $\prod_{j=1}^{|\Phi_b^t|} Enc_{pk_s}(|sn_j^t|)$ in Φ_b^t

6 **for** *each* $H_j \in \Phi_b^t$ **do**

7 H_j computes and sends $\prod_{j=1}^{|\Phi_b^t|} Enc_{pk_s}(|sn_j^t|)^{\frac{1}{|sn_j^t|}}$ to H_s

8 H_s decrypts the ciphertexts and broadcasts the allocation ratio within the seller coalition Φ_s^t

9 **repeat**

10 H_i computes $e_{ij} = \frac{|sn_j^t|}{E_b^t} * sn_i^t$

11 H_i routes e_{ij} to H_j

12 H_j pays $m_{ji} = e_{ij} * p^t$ to H_i

13 **until** *each* $H_i \in \Phi_s^t$ *finishes transaction*;

14 **else**

15 **Repeat** Lines 2-13 by replacing Φ_s^t with Φ_b^t

16 **return** $e_{ij} = \frac{sn_i^t}{E_s^t} * |sn_j^t|$ and $m_{ji} = e_{ij} * p^t$

Algorithm 9: Private Distribution

seller H_{r1}, a randomly selected buyer H_{r2}, and the remaining sellers/buyers.

We first examine the received messages of the remaining sellers/buyers. Each of them only receives a ciphertext of a random number (which cannot be decrypted without the private key), which can be polynomially simulated by repeating the encryption with the public key. Thus, the protocol does not reveal private information to them.

H_{r1} and H_{r2} can decrypt the ciphertexts to obtain two different random

numbers R_b and R_s (which are decrypted with the private keys), respectively. Each random number can be polynomially simulated by generating a random number from the uniform probability distribution over range $\mathcal{F}$. Notice that the random numbers are scaled to fixed precision over a closed field (after decryption), enabling such a selection. Thus, $Pr[\sum_{i=1}^{n} R_b \textit{ is simulated}] = Pr[\sum_{i=1}^{n} R_s \textit{ is simulated}] = \frac{1}{\mathcal{F}}$. Finally, H_{r1} and H_{r2} also securely execute Fairplay to compare two random numbers for market evaluation, which does not reveal any private information (as proven in [25]). □

Lemma 7. *The Private Pricing (Protocol 8) only reveals non-private information* $\sum_{H_i \in \Phi_s^t} k_i^t$ *and* $\sum_{H_i \in \Phi_s^t} (g_i^t + 1 + \epsilon_i^t b_i^t - b_i^t)$ *to a randomly selected buyer* H_b.

Proof. This Protocol involves two different types of parties: a randomly selected buyer H_b and all the sellers.

We first analyze H_b's received messages. H_b can decrypt the received ciphertexts with its private key to obtain $\sum_{H_i \in \Phi_s^t} k_i^t$ and $\sum_{H_i \in \Phi_s^t} (g_i^t + 1 + \epsilon_i^t b_i^t - b_i^t)$. Although such two aggregated values are revealed to H_b, H_b cannot learn any seller's private data, e.g., $k_i^t, g_i^t, b_i^t, \epsilon_i^t$ from the aggregated results.

On the other hand, all the sellers receive only two ciphertexts and cannot decrypt them without the private key. Since each seller can polynomially simulate its received two ciphertexts using the public key (by repeating the encryption), we can claim that the protocol does not reveal any information to the sellers. □

Lemma 8. *The Private Distribution (Protocol 9) only reveals the non-private market demand ratios* $\frac{E_b^t}{|sn_j^t|}, H_j \in \Phi_b^t$ *to the seller coalition (in the general market), or the non-private market supply ratios* $\frac{E_s^t}{|sn_i^t|}, H_i \in \Phi_s^t$ *to the buyer coalition (in the extreme market).*

Proof. Similar to the proof in Lemma 7, we can prove that the seller coalition can

only receive the demand ratios $\frac{E_b^t}{|sn_j^t|}, H_j \in \Phi_b^t$ (from the buyer coalition) in general market. Moreover, buyer coalition can only receive the supply ratios $\frac{E_s^t}{|sn_i^t|}, H_i \in \Phi_s^t$ (from the seller coalition) in extreme market. However, they cannot learn any supply or demand from the ratios in these two cases. □

Theorem 3. *The PEM framework only reveals the aggregated information articulated in Lemma 7 and 8.*

Proof. Since Private Market Evaluation does not reveal any privacy where the secure comparison result (either 0 or 1) can be polynomially simulated, PEM only reveals some trivial information articulated in Lemma 7 and 8 per the composition theory of secure multiparty computation [39]. □

3.5.2 Incentive Analysis. We give the formal proof of incentive for our protocols.

Theorem 4. *The PEM framework ensures individual rationality and incentive compatibility.*

Proof. We first evaluate the individual rationality. In the general market, if each buyer H_j directly purchases energy from the main grid at the price ps_g^t, which is greater than $p^{t*} \in [p_l, p_h]$, the cost will increase; if each seller H_i directly sells the energy to the grid at the price pb_g^t, which is less than the p^{t*}: the payoff will decrease. In the extreme market, the buyer can buy the energy from the PEM with a lower price ($p_l < ps_g^t$) and the seller can still sell the energy with a higher price ($p_l > pb_g^t$), both of which receive more payoffs. This proves the individual rationality.

Second, we discuss the incentive compatibility for two different markets: for the general market, we assume that there exists one seller $H_i \in \Phi_s^t$ which untruthfully utilizes its net energy $sn_i^{t'}$ by adjusting its load profile to $l_i^{t'}$. Per Lemma 5, there exists only one load profile l_i^{t*} to reach the equilibrium and return the optimal price

p^{t*}. Then, it is impossible to find another $l_i^{t'} \neq l_i^{t*}$ since p^{t*} is derived only if all the sellers hold the $l_i^{t*}, H_i \in \Phi_s^t$ profile. On the contrary, as all the sellers hold the optimal load profile, the buyers cannot reduce the total costs by decreasing market price.

In addition, for the extreme market, the buyers purchase all the energy from the PEM with a lower price $p_l < ps_g^t$, then rational buyers cannot gain more payoff with untruthful participation (since the payment cannot be lower). For any rational seller H_i, if H_i untruthfully utilizes a higher supply to increase its allocated amount of sold energy, the market price would be reduced (no additional payoff, either). This proves the incentive compatibility. □

3.5.3 Complexity Analysis.

We present the complexity analysis as following.

Lemma 9. *The complexity of protocols in the PEM is* $O(n^2)$.

Proof. It is straightforward to analyze the complexity of algorithms in our PEM framework. First, Private Market Evaluation algorithm has complexity $O(n)$ – securely aggregating random values is $O(n)$ while secure comparison is $O(1)$. Similarly, Private Pricing algorithm has complexity $O(n)$, and Private Distribution algorithm has complexity $O(n^2)$. Therefore, the complexity of the PEM framework is $O(n^2)$. □

3.6 Discussion

Generalization of PEM. PEM can be extended to Vehicle-to-Grid (V2G) applications [77] by considering electrical vehicles as agents with local energy. Last but not least, the proposed PEM is a general framework for privacy preserving energy trading (focusing on privacy and incentive compatibility), which can be readily extended for ensuring privacy and incentive compatibility for other applications on the power grid (e.g., energy trading w.r.t. future prices, energy trading by possibly storing energy for the future, and demand response [78]). Finally, PEM can also be adapted for trading

other products, such as the allocation of spectrum in the cognitive radio networks [79], and the Wifi & LTE sharing [80].

Seller/Buyer Coalitions. We forms coalitions for sellers and buyers in our PEM. First, the formation of coalition can enable the agents to cooperate to achieve more benefits/social welfare compared with trading directly with the monopoly, the main grid. Recall that coalitions make the market more stable for such emerging applications, e.g., ensuring the fairness among the seller/buyer coalition by allocating the amounts based on sellers/buyers' shares in the market supply/demand. Such setting would be more applicable for conservative agents. Nevertheless, it is also worth exploring the privacy preserving schemes for non-cooperative energy trading or fully competitive energy market [81], which left for future work.

Malicious Model. PEM is based on the semi-honest model, and each agent (rational) is also assumed to have incentives to cheat for payoffs. Our model can also be extended to defend against malicious agents, which may deviate the protocol (regardless of their payoffs) by faking the trading data, colluding with other agents, and/or performing advanced attacks. For instance, we can design verifiable and collusion-resistant schemes (e.g., detect the violation of data integrity, and prevent collusion by randomly picking agents [82]).

Scalability. With the advancement of distributed computation [2, 82], secure computation [22, 23] can be applied to perform complex computation on the smart grid. Each distributed agent (e.g., a smart home) can also locally compute the data, such that the computational load of whole system can be greatly reduced. As shown in the experimental settings in Section 3.7.1, we take advantage of the container technology, e.g., Docker, to emulate local computing agents for different smart homes in the PEM. High efficiency and scalability of PEM have been demonstrated.

Blockchain Deployment. PEM can also be integrated with the emerging blockchain technology [83]. Specifically, the final distribution and transaction between the sellers and buyers can be realized by the smart contract of the blockchain to ensure the integrity and truthfulness (extra anonymity and privacy should be ensured on the blockchain) [84]. Moreover, the on-line blockchain can also facilitate the communication of the MPC protocols in the PEM.

Secure Computation. The recent protocols/systems on secure computation (e.g., MPC-as-a-service [85], against both semi-honest and malicious adversaries [86], MPC for small number parties [87]) cannot be adapted to solve our problem for the following two major reasons: (1) whether the system can function real time transactions has not been validated in most of such systems (we have validated the feasibility and scalability of deploying PEM in real time in Section 3.7), and (2) incentive problems are not studied in most of such systems. Thus, the proposed cryptographic protocols in PEM can also complement the literature of secure computation for privacy preserving trading (which is limited to our best knowledge).

3.7 Evaluation

In this section, we illustrate our system implementation for the PEM framework and demonstrate the experimental results.

3.7.1 Experimental Setup. Our PEM framework is deployed on the NSF CloudLab platform (https://docs.cloudlab.com), of which the server has eight 64-bit ARMv8 cores with 2.4 GHZ, 64GB memory and 120GB of flash storage with Ubuntu:16.04 OS. Docker (https://docs.docker.com/) is utilized to start a container for each buyer/seller. We created the image for container based on the raw image of Ubuntu 16.04 by integrating all the system environments (e.g., JRE and JDK), and source codes.

We conducted the experiments on 300 smart homes' real power generation data (solar panels) and load data over one day (available at UMASS Trace Repository [88]). We tune the following parameters in evaluations:

- the number of smart homes $n \in [100, 300]$;
- the number of trading windows $m \in [1, 720]$: from 7:00AM to 7:00PM (a trading window per minute);
- the key size: 512/1024/2048-bit.

Benchmark. There is no existing schemes which can be directly applicable to our problem setting. Then, we use the traditional energy trading (without PEM) as the benchmark: all the agents directly purchase energy from the main grid. Specifically, if a seller (with excessive energy) will sell them back to the main grid with the offered price pb_g^t, and the buyer (short of energy) will buy energy from the main grid with the retail electricity price ps_g^t. We set the retail price as ps_g^t=120 cents/kWh and offered price from the main grid pb_g^t= 80 cents/kWh. We also set the price range of PEM as $[90, 110]$ cents/kWh. Note the interaction between agents and the main grid will increase greatly for trading without PEM.

Figure 3.4 illustrates the sizes of seller and buyer coalitions (the number of smart homes) in all 720 trading windows. The roles of smart homes change over time.

3.7.2 Computational Performance Evaluation. We evaluate the computational cost of PEM among 100 to 300 agents, using three different key sizes (512/1024/2048-bit). Fig. 3.5(a) shows the average runtime for a single trading window (including securely evaluating the market, computing the optimal price as well as the energy trading distribution amounts) as the number of trading windows varies from 1 to 720. The average runtime for each trading window is around 1 sec. This indicates that our

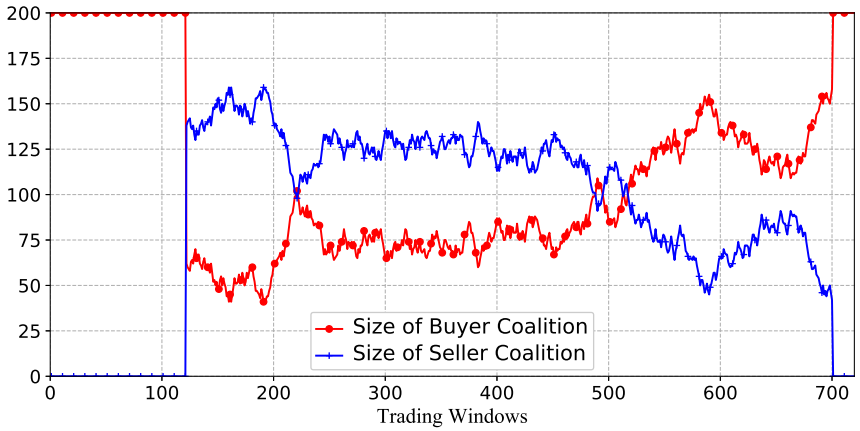
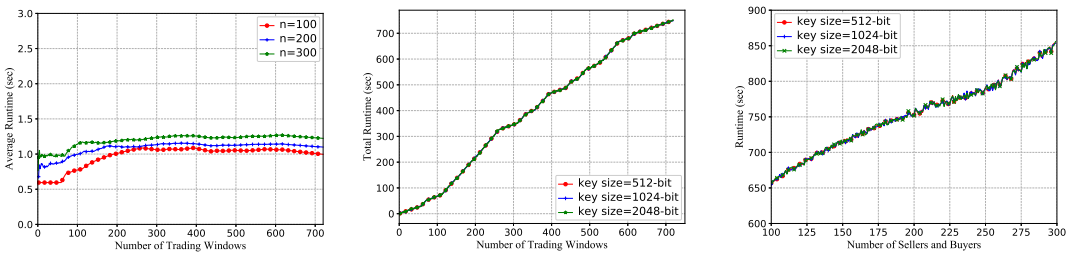


Figure 3.4. Coalition Sizes vs. Trading Windows

PEM framework can efficiently function real-time trading in practice (*with negligible latency*).

Fig. 3.5(b) demonstrates the total runtime on different number of trading windows among 200 agents (with three different key sizes 512/1024/2048-bit). Given the same number of trading windows, we observe that the key size for encryption and decryption executed in our protocols does not affect the runtime (since the encryption and decryption are independently executed in parallel during idle time). Finally, Fig. 3.5(c) validates that the total runtime increases as the number of agents increases.



(a) Runtime (2048-bit) (b) Runtime (c) Runtime (720 Windows)

Figure 3.5. Computational Performance Evaluation for the PEM Framework (negligible latency for minute-level inputs)

3.7.3 Energy Trading Performance Evaluation. We have also evaluated the trading performance of our PEM framework from the following perspectives:

- the optimal price in all the trading windows;
- utility received by some representative sellers;
- total cost Γ^t for the buyer coalition;
- interactions with the main grid.

3.7.3.1 Optimal Trading Price. Fig. 3.6(a) shows the optimal prices in all the 720 trading windows. We can observe that the price changes over time: in the first few trading windows, the price equals ps_g^t (purchasing all the energy from the grid). This shows that at the beginning of the day, the generation is close to 0, all the agents have to buy energy from the main grid. Similarly, at the end of day (around 7:00pm), the price is still ps_g^t for the same reason. Furthermore, in many trading windows in the middle of the day, the trading price would be lower bounded: either the optimal price in the general market is out of range (this also applies to the upper bound), or the extreme market occurs.

3.7.3.2 Utility and Total Cost. We fix the preference parameter $k = 20, 40$ for all the sellers in different trading windows. Fig. 3.6(b) presents the utility of two agents (*which are sellers in all 720 trading windows*). We have the following observations:

- The utility of the agents with our PEM framework is higher than their utility without PEM (buyers only purchase energy from the main grid).
- The utility improvement (with the PEM) in case of $k = 40$ is higher than $k = 20$. since lower preference parameter would make the sellers to sell more local energy (which results in more payoff).

In addition, Fig. 3.6(c) shows the total cost of buyer coalition in the PEM (for 100 and 200 agents), which can be greatly reduced in all trading windows (e.g., 25.3% in the current setting on average).

3.7.3.3 Interaction with the Main Grid. Our PEM framework can also benefit the main grid by reducing the interactions between the agents and the grid, which is measured by the amount of electricity all the agents request from or feed into the grid. As shown in Fig. 3.6(d), since more energy can be traded in the PEM framework among agents, the interactions with the PEM are much lower than the original energy consumption (without the PEM).

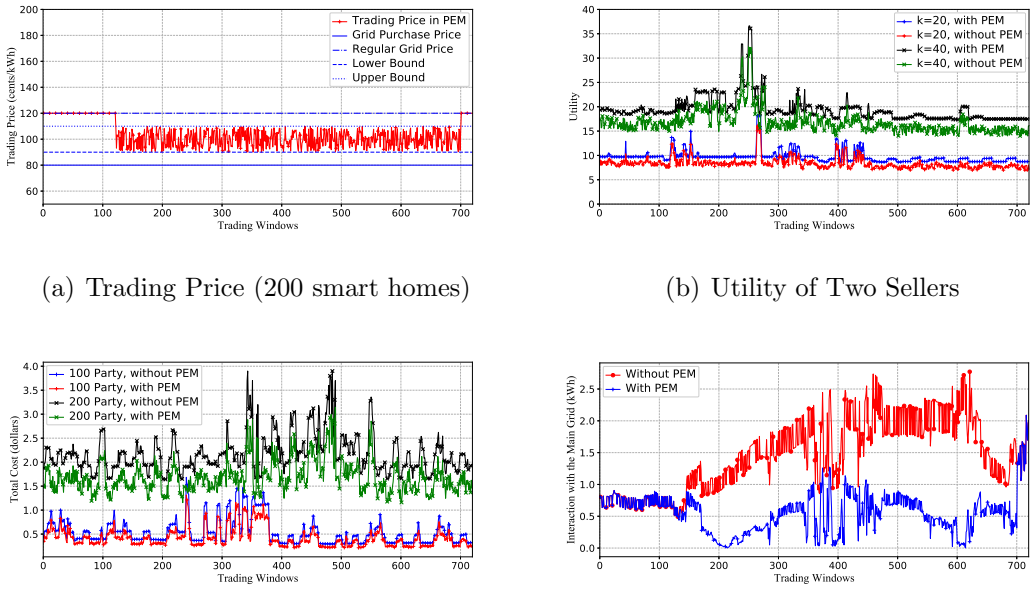
3.7.4 Communication Overheads. We have also evaluated the bandwidth consumption of all the smart homes while executing the secure computation and communication among the 200 smart homes with different key sizes (512-bit, 1024-bit and 2048-bit). Table 3.2 shows the average bandwidth over different numbers of trading windows (of all the smart homes). With such minor bandwidth consumption, our PEM framework can be deployed in most of the networking environments.

Table 3.2. Average Bandwidth (MB) over m Trading Windows

m	300	360	420	480	540	600	660	720
512-bit	0.45	0.54	0.48	0.52	0.47	0.48	0.55	0.46
1024-bit	0.84	0.88	1.02	0.93	0.98	1.06	0.97	0.96
2048-bit	1.87	2.12	2.05	2.11	2.20	2.16	2.05	2.01

3.8 Related Work

Smart Grid Privacy. Most smart grid privacy research focuses on protecting data collected from smart meters integrated in the power grid [50]. Different privacy



(a) Trading Price (200 smart homes)

(b) Utility of Two Sellers

(c) Total Costs of Buyer Coalition

(d) Interactions with Main Grid

Figure 3.6. Energy Trading Performance Evaluation for the PEM Framework

preserving techniques have been proposed to tackle such privacy concerns [12, 18]. For instance, He et al. [89] presented a distortion based privacy preserving metering scheme by introducing tolerable noise to obfuscate the consumption data. Rottondi et al. [18] leveraged secure communication protocols to implement a privacy preserving infrastructure which allows utilities and data consumers to collect measurement data by securely aggregating smart metering data. [72] studied how to utilize the renewable energy sources (i.e., batteries) to hide the load/metering information of individual households. [90] proposed a privacy-preserving way to aggregate smart metering data for the billing of utility provider. Recently, [60] researches on privately balancing the power load between the main grid and agents (microgrids). However, none of such techniques can be applied to multiagent energy trading.

Secure Computation. The theory of Secure multiparty computation (MPC) [22, 23] has significantly advanced the development of collaborative computation among multiple parties, which guarantees that functions can be securely computed with limited

disclosure. Recently, secure computation has been intensively applied for privacy preserving system design in different contexts such as location-based services [48], and medical data analysis [49]. Moreover, Furukawa et al. [86] have recently proposed a three-party secure computation against both semi-honest and malicious adversaries, which achieves low communication complexity and simple computation. Barak et al. [85] have proposed the MPC-as-a-service concept and implemented an end-to-end system for large scale P2P secure computation with low bandwidth.

Energy Trading. Energy trading has been widely discussed with the development of smart grid. The integration of renewable energy sources has greatly motivated studies of energy market, e.g., incentive mechanisms for trading [91] and multi-agent energy management [92], which could improve the stability and utility of the grid. Furthermore, distributed energy trading has been identified as a promising scheme for the energy market in [65, 66]. There are also many ongoing projects, e.g., LO3 Energy (`https://lo3energy.com/`) which focuses on the commercial energy trading to encourage residential units to trade with the neighborhoods. [93] focuses on the energy trading by blockchain. However, the scalability of the proposed scheme is not very clear. To the best of our knowledge, we design and implement the first privacy preserving distributed energy trading framework.

CHAPTER 4

ROBUSTNESS EVALUATION OF VIDEO RECOGNITION SYSTEMS

4.1 Introduction

Deep neural network (DNN) models have been extensively studied to facilitate a wide variety of intelligent video recognition systems, such as face recognition [94], action recognition [95] and anomaly detection [96]. For instance, self-driving vehicles are equipped with many cameras to capture the visual information. Then, DNNs are adopted to accurately recognize road signs, detect and predict trajectories of pedestrians and vehicles, and thus make the driving decisions [97, 98]. Video anomaly detection systems [96, 99] integrate DNNs to monitor the activities under surveillance, and trigger alarms once anomalies (e.g., traffic accident, theft, and arson) are visually identified to advance the public safety. [5]

However, DNNs have been revealed to be inherently vulnerable to adversarial attacks, where attackers can add well-crafted imperceptible perturbations to the inputs to deviate the learning results. Such attacks are initially identified in the image domain [101–105], and have also attracted significant interests in other contexts, e.g., text understanding [106], and voice recognition [107–109]. Similarly, adversarial perturbations to the DNNs in video recognition systems could potentially cause severe physical and financial damages. For instance, they may misdirect the DNN models in autonomous vehicles to inaccurately recognize objects and make detrimental decisions towards accidents. Furthermore, DNN-based anomaly detection models in video surveillance or CCTV might be deviated via the perturbations to misclassify anomalous activities to routine ones, and vice-versa [96].

Although the adversarial attacks on images have been well-explored, there are

[5] This work has been published in IEEE S&P [100].

very limited works on attacking DNN models for videos [110–113], which need to address additional challenges, e.g., larger data sizes, a new set of DNN models for learning actions in the videos, different types of features extracted with additional temporal convolution, and different realizability. To our best knowledge, current video attacks [110–113] adapt image perturbations in a frame-by-frame fashion to subvert DNNs for video classification, which have the following major limitations.

1. Frame-by-frame image perturbations may overly perturb the videos (human perceptible), and also lack the temporal consistency in the perturbations. These make the attacks not robust against the state-of-the-art detection schemes (e.g., AdvIT [114]). Adversarial examples crafted by [111–113] can be accurately detected by AdvIT (as evaluated in our experiments).

2. Frame-by-frame image perturbations may not be well aligned with the video frames (*boundary effect* by misaligning the perturbation and video frames) [111].

3. Crafting adversarial examples for videos frame-by-frame results in heavy computation overheads and lacks universality. It limits the application to attack large-scale videos or streaming videos (e.g., CCTV surveillance).

To address the above limitations, we propose a black-box attack framework that generates *universal 3-dimensional (U3D)* perturbations to subvert a wide variety of video recognition systems. U3D has the following major advantages: (1) as a transfer-based black-box attack [115–117], U3D can universally attack multiple DNN models for video recognition (each of which can be considered as the target model) without accessing to the target DNN models; (2) the high transferability of U3D makes such black-box attacks easy-to-launch, which can be further enhanced by integrating queries over the target model when necessary (validated); (3) U3D ensures good human-imperceptibility (validated by human survey); (4) U3D can bypass the existing

state-of-the-art defense schemes (extended towards defending against U3D), including universal adversarial training (UAT) [118, 119], detection schemes [114, 120], and certified schemes (e.g., PixelDP [121] and randomized smoothing [122]); (5) U3D perturbations can be generated on-the-fly with very low computation overheads (e.g., $\sim$0.015s per frame) to attack DNN models for streaming videos.

Specifically, in the attack design, we generate perturbations by maximally deviating features over the feature space representation of the DNNs while strictly bounding the maximum perturbations applied to the videos. We aim at generating more transferable adversarial examples (to be misclassified by multiple DNN models) by explicitly optimizing the attack performance w.r.t. layer-wise features of a video DNN model. Moreover, we integrate boundary effect mitigation and universality into the optimization for learning the U3D perturbations.

Different from traditional black-box attacks that may request intensive queries over the target DNN model, U3D perturbations can be efficiently derived independent of the target DNN model. Assuming that the adversary does not need to know the target DNN model under the black-box setting (and no need to query over the target model by default), our U3D attack computes the perturbation using a surrogate DNN model (any public DNN model, *which can have very different model structure and parameters from the target model*). Such black-box attacks are realized via high transferability across multiple DNN models on different datasets (as validated in Section 4.5.3). We have also shown that our U3D attack can integrate queries over target model when necessary (turning into a hybrid black-box attack [123]).

Figure 4.1 demonstrates an example of the U3D perturbation, which is continuously generated. Compared to the state-of-the-art universal perturbations (see Section 4.5), U3D achieves higher success rates with significantly less perturbations (mostly between [0,10] in grayscale [0,255]). It is also highly efficient for attacking mul-

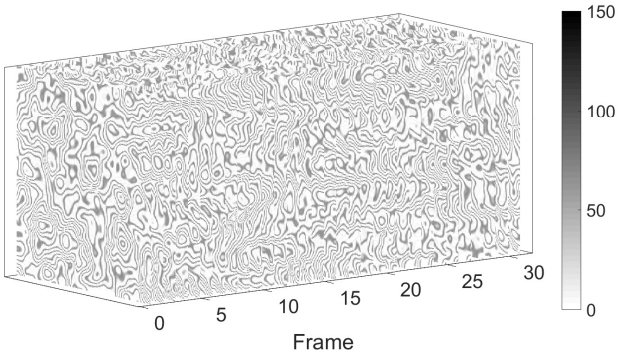


Figure 4.1. Universal 3-dimensional (U3D) Perturbation

tiple video recognition systems (e.g., classification and real-time anomaly detection). Therefore, we summarize our main contributions as below:

- To our best knowledge, we propose the first black-box attack that generates 3D perturbations to universally subvert multiple DNN-based video recognition systems.
- We construct two different types of novel U3D perturbations optimized in the feature space representation of DNNs, which can practically attack various DNN models and the related video recognition systems (e.g., classification and anomaly detection) with high transferability.
- We conduct extensive experiments to validate the U3D attack while benchmarking with the state-of-the-art attacks (e.g., C-DUP [111], V-BAD [113] and H-Opt [112]). Evaluations include success rate, transferability, universality, human-imperceptibility, performance against defenses, physical realization, and efficiency. The results have shown the superiority and practicality of U3D.
- In particular, we also evaluate the U3D against different types of state-of-the-art defense schemes. We have extensively adapted the defenses w.r.t. U3D, and studied the potential mitigation of the U3D. The high attack performance

against defenses reveals the potential severity of the adversarial attack and the vulnerabilities in the DNN-based video recognition systems. Our novel U3D attack can facilitate the development of more robust and trustworthy DNN models for video recognition.

4.2 Background

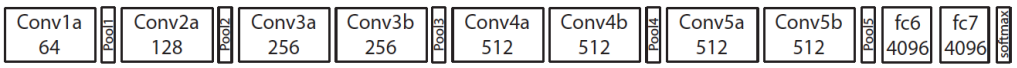
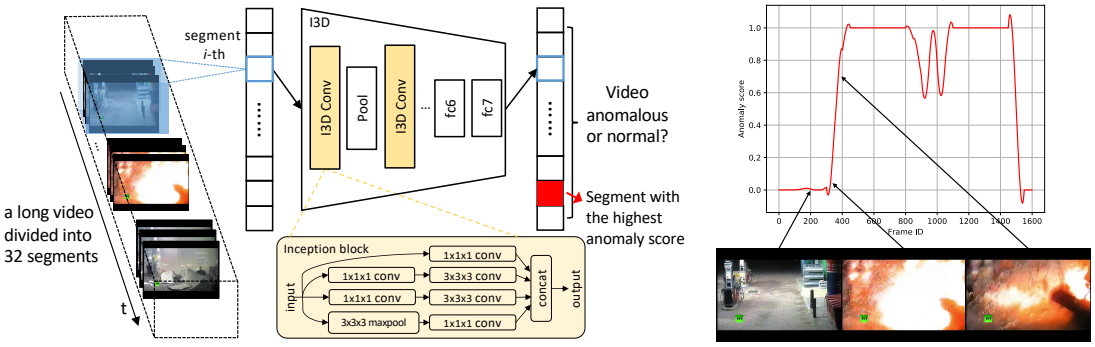


Figure 4.2. The C3D architecture [124] consists of 8 convolution, 5 max-pooling, and 2 fully connected layers, followed by a softmax output layer. All 3D convolution kernels are 3 ×3 ×3 with a stride of 1 in both spatial and temporal dimensions. The number of filters is denoted in each box [124]. The 3D pooling layers are represented as pool1 to pool5. All pooling kernels are 2 × 2 × 2, except for pool1, which is 1 × 2 × 2. Each fully connected layer has 4,096 output units.

4.2.1 DNN-based Video Recognition Systems. DNNs have been widely adopted for accurate video recognition in numerous real-world applications, e.g., anomaly detection [96], self-driving vehicles [98] and smart security cameras [125]. There have been a series of works on designing video DNNs to improve model accuracy [124, 126–128]. For instance, Donahue et al. [127] proposed the long-term recurrent convolutional networks (LRCNs) for video recognition and description via combining convolutional layers and long-range temporal recursion. Moreover, two-stream network (TSN) [128] fusing static frames and optical flows was proposed for action recognition. Later, Tran et al. [124] proposed the C3D model to significantly improve classification accuracy by focusing on spatio-temporal feature learning with 3D convolutional neural network. Figure 4.2 shows the detailed structure and the characteristics of the C3D network. Recently, more networks built on spatio-temporal convolutions (e.g., I3D [129]) have been exhibited high performance, which greatly promoted the video recognition systems. Figure 4.3(a) illustrates a video anomaly detection system with the I3D model (by integrating additional optical flow information into the spatio-

temporal features). Figure 4.3(b) illustrates the curve of anomaly scores inferred by the I3D model in time series. At first, the score is close to 0 for the normal scenery. Then, it increases as the explosion occurs to report such anomalous event (with a threshold).



(a) Anomaly detection [96] with I3D. (b) Sample Output of Anomaly

Figure 4.3. Video anomaly detection system for detecting an "Explosion" event. The score increases from 0 (normal scenery) to high (explosion).

4.2.2 Threat Model. The U3D attack is applicable to the *offline* scenario, which is identical to the attack scenario of adversarial perturbations for other types of data, e.g., images [102, 104, 105], texts [106], and audio signals [107–109]. For instance, the adversary can craft adversarial examples by adding the pre-generated U3D perturbations to static videos. Then, the perturbed videos will be misclassified to wrong labels.

Furthermore, our U3D attack can work *online* to perturb the streaming video (e.g., real-time anomaly detection in CCTV surveillance). This is also feasible since our U3D perturbations are designed to universally perturb any video at any time (from any frame in the streaming video) without the boundary effect. Thus, the U3D perturbations can be generated offline and injected into the online videos in real-time applications.

Adversary's Capabilities. The adversary can either craft adversarial examples offline

on static videos, or inject the U3D perturbations (pre-learned) into the streaming videos, similar to the attack setting in [105, 111]. Specifically, the adversary can manipulate the systems via malware, or perform man-in-the-middle (MITM) attack to intercept and perturb the streaming videos. Furthermore, the adversary could also slightly delay the streaming video when performing injections without affecting the overall quality of the streaming video.

Note that MITM adversary is unable to perform attacks by simply replacing streaming videos with pre-recorded videos or static frames while ensuring the stealthiness of the attack, since the adversary does not know what will happen in the future [111]. For instance, if the adversary wants to fool the video surveillance system in a parking lot, he/she may need to replace the video streams in long run (ideally all the time) to perform the attack. However, without prior knowledge on the future objects/events in the parking lot, it would be very hard to make the replaced video visually consistent with the real scenario (e.g., moving vehicles, humans, and weather). Then, the replaced video can be easily identified by the security personnel. Instead, U3D attack can be easier to be covertly realized (always human-imperceptible). The universal and boundary effect-free perturbation will be efficiently generated and continuously injected in real time (see our design goals in Section 4.3.1). Thus, it can universally attack video streams in long run even if video streams may differ at different times.

We experimentally study the practicality of attack vectors (e.g., man-in-the-middle attack) in a video surveillance system [130–132] and implement the real-time attack based on U3D. The results show that U3D is efficient to attack real-time video recognition systems (as detailed in Section 4.5.6).

Adversary's Knowledge (black-box). Similar to other black-box transfer-based attacks [115–117], the adversary does not necessarily know the structure and parameters

of the target DNN model. U3D aims to generate universal perturbations that can successfully subvert a variety of DNN models, each of which can be the potential target DNN model. By default, the adversary does not need to query the learning results (e.g., classification score or label) from the target model either.

To successfully perform the attack, the adversary will leverage the *high transferability* of U3D to deviate the target DNN models. Specifically, we assume that the adversary can utilize any public DNN model as the surrogate (e.g., C3D, I3D, LRCN and TSN) and some labeled videos (e.g., from any public data such as the HMDB51 dataset [133]). Such data are not necessarily included the training data of the target DNN model. *The surrogate model can be very different from the target model.* Without querying the target model, the U3D attack is even easier to realize than the conventional query-based black-box attacks [134–136].

Indeed, the U3D attack can also integrate queries over the target DNN model when necessary (see such extended attack design and evaluations in Section 4.5.4). Thus, the transfer-based back-box attack will turn into a hybrid black-box attack [123], which integrate both query-based and transfer-based attack strategies to improve the attack performance under the black-box setting. We have experimentally validated that integrating a number of queries over the target DNN model could slightly enhance the success rates.

4.3 U3D Attack Methodology

4.3.1 U3D Attack Design Goals. The goals in our U3D attack design include:

- G1: The attack should achieve high performance on the video recognition systems under the black-box setting.
- G2: The adversarial perturbations should be very small to obtain good human-

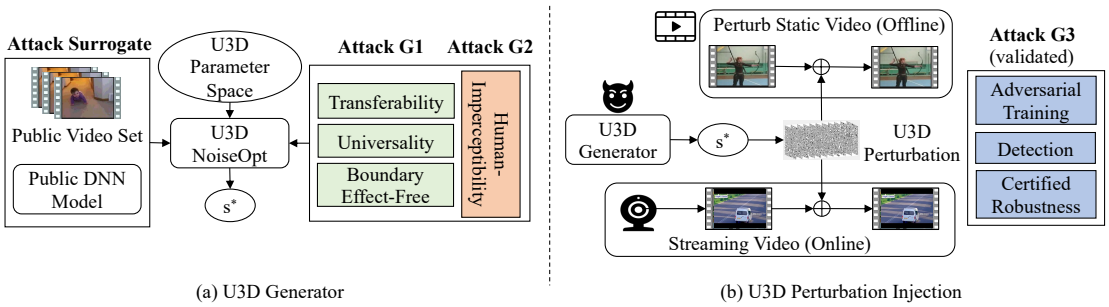


Figure 4.4. U3D attack framework (including three design goals: G1, G2 and G3). (a) `U3D_Generator` learns the near-optimal U3D parameters s^*. (b) U3D perturbations can be generated on-the-fly with s^* to perturb both static and streaming videos.

imperceptibility.

- G3: The adversarial examples are robust against existing defense schemes (cannot be easily detected or mitigated).

G1: High Attack Performance. To launch the U3D attack, the following properties are desired: (1) transferable on a wide variety of DNN models for video recognition; (2) universal on a large number of videos; (3) boundary effect-free.

Different from increasing the magnitude of the perturbations for transferability [101, 108, 137], we formulate an optimization problem with a surrogate DNN model (which can be any public DNN model) in an interpretable fashion. The objective is to maximize the distance between the clean video and perturbed video in the feature space representation (Section 4.4.2.1). First, the change of feature space representations via perturbations (especially the deep spatio-temporal feature space for videos) will non-trivially impact the classification results. This will increase the success rates of the attack. Second, the explicit attack in the feature space could craft more transferable adversarial examples since the intermediate layer features of DNNs have shown to be transferable [138]. Experimental results in Section 4.5 have demonstrated high cross-model transferability for feature space perturbations.

Moreover, the adversary does not have prior knowledge on the video (especially the streaming video), then the 3D perturbations should universally attack [103] a large number of videos (ideally, any video). We construct U3D perturbations from a relatively small set of videos to fool the target DNN model on arbitrary input videos with high success rates.

With temporal dimensions on multiple frames, the video perturbations should address the potential misalignment with the input video (boundary effect [111]), which can degrade the attack performance, especially in long run. While launching attacks, the perturbation should be effectively injected at any time in the video. To address the misalignment, we employ a transformation function to convert the perturbation temporally, and then optimize the attack on all temporal transformations (see Section 4.4.2.2), which enable the U3D perturbations to be injected at random times without the boundary effect.

G2: Human-Imperceptibility. We add a bound on the U3D perturbations with ℓ_∞-norm, which strictly restricts the pixel deviations. Later, we use MSE metrics to quantify the perturbations in the experiments. Moreover, we conduct surveys among humans to illustrate the imperceptibility of U3D.

G3: Robustness against Defenses. To show the robustness of our U3D attack, we implement attacks on the video recognition models equipped with defense schemes (*G3 is not directly designed but ensured with post-validation*). There are two rules of thumb for evaluating attacks: (1) we should carefully utilize current effective defenses to explicitly defend against the newly proposed attack, e.g., developing adaptive schemes which uncover the potential weaknesses of the attack; (2) the defenses should be in white-box setting, i.e., the defender should be aware of the attack, including the adversary's knowledge and strategy. The rules of thumb also work for evaluating

newly proposed defenses vice versa [139–141].

Specifically, we adapt three types of major defense schemes: (1) adversarial training [118, 119]; (2) detection [114, 120]; (3) certified robustness [121, 142]. We redesign the defense schemes to defend against universal perturbations or U3D per the rules of thumb. For example, based on the adversarial training (AT) [118, 119], we design the U3D-AT, which utilizes the capability of AT to defend against the best U3D (iteratively updating U3D perturbations). See details in Section 4.6.

4.3.2 U3D Attack Overview. We now overview the U3D attack in Figure 4.4. We first formulate the U3D perturbation generation (`U3D_Generator`) by synthesizing the procedural noise [143, 144] (which can be efficiently generated with low-frequency patterns) with the U3D parameters s (see Section 4.4.1). Meanwhile, the attack goals of U3D are optimized: transferability, universality, and boundary effect-free (see Section 4.4.2). Then, we apply the particle swarm optimization (PSO) to solve the problem to derive the near-optimal parameters s^* for generating U3D perturbations (see Section 4.4.3). Finally, U3D perturbations can be generated on-the-fly to be injected into the videos in the attack scenarios (either static or streaming videos).

4.3.3 U3D Attack Formulation. The DNN model can be modeled as a function $f(\cdot)$ that infers the video v with a label (e.g., the label with the top-1 probability). The attack crafts a video adversarial example v' by injecting the perturbation ξ into the original video v: $v' = v + \xi$, where the output label of v' by the DNN model $f(\cdot)$ would be $f(v') \neq f(v)$ (as a universal attack).

To pursue human-imperceptible perturbations, ℓ_∞-norm is adapted to bound the distance between the original and perturbed videos (w.r.t. U3D perturbation ξ) with a pre-specified small value ϵ: $||v' - v||_\infty = \max_i |\xi| \leq \epsilon$. Then, we formulate an optimization problem to generate U3D perturbations:

$$\arg\min_{\xi} : \Gamma(v+\xi), s.t.\ ||\xi||_\infty \leq \epsilon \tag{4.1}$$

where Γ is a loss metric function, e.g., a distance or cross-entropy metric. In Section 4.4.2, we align the objective function with the attack goals in the optimization for the U3D design.

4.4 Attack Design

4.4.1 U3D Perturbation Formalization. "Procedural noise" [143–146] refers to the algorithmically generated noise with a predefined function, which can be added to enrich visual details (e.g., texture, and shading) in computer graphics. It can be directly computed with only a few parameters, and has no noticeable direction artifacts [104, 143, 144]. These properties make it *potentially fit for inexpensively computing adversarial perturbations*. While constructing U3D perturbations, we utilize two types of common procedural noises: (1) "Perlin noise" [143,145] (a lattice gradient noise) due to its ease of use, popularity and simplicity; (2) "Gabor noise" [146] (a convolutional sparse noise) with good sparsity and accurate spectral control. We propose two types of U3D perturbations, "U3D$_p$" and "U3D$_g$", both of which universally perturb videos to subvert the DNN models.

We first formally define the U3D noise function. Denote $\mathcal{N}(x, y, t; S)$ as the U3D noise function, where (x, y, t) represents the 3D coordinates of each pixel in the video, and S is the parameter set for noise generation.

4.4.1.1 U3D$_p$ Noise. Perlin noise [143, 145] originally works as an image modeling primitive to produce natural-looking textures in realistic computer generated imagery.

Specifically, we denote every pixel in a video by its 3D coordinates (x, y, t) where (x, y) are the coordinates in frame t, and denote the Perlin noise value of the pixel (x, y, t) as $p(x, y, t)$. To model the change of visual perturbations, we define three

new parameters of wavelength λ_x, λ_y, λ_t to determine the octaves along the three dimensions x-axis, y-axis, and frame t, respectively, and define the number of octaves as Λ. The newly updated noise is computed as the sum of all the corresponding octaves for 3D coordinates:

$$\mathcal{N}(x, y, t) = \sum_{\ell=0}^{\Lambda} p(x \cdot \frac{2^\ell}{\lambda_x}, y \cdot \frac{2^\ell}{\lambda_y}, t \cdot \frac{2^\ell}{\lambda_t}) \tag{4.2}$$

Moreover, we compose the noise function with a color map function [147] to generate distinct visual perturbations in the video. Then, the noise of pixel (x, y, t) can be derived as:

$$\mathcal{N}_p(x, y, t) = cmap(\mathcal{N}(x, y, t), \phi) \tag{4.3}$$

where $cmap(p, \phi) = \sin(p \cdot 2\pi\phi)$ is a sine color map function, which ensures *the bound of noise value with the circular property.* ϕ indicates the period of the sine function, and the visualization of perturbations can be tuned with ϕ.

U3D$_p$ Parameters. Combining Equation 4.2 and 4.3, we denote the corresponding parameter set as S_p for U3D$_p$ noise:

$$S_p = \{\lambda_x, \lambda_y, \lambda_t, \Lambda, \phi\} \tag{4.4}$$

4.4.1.2 U3D$_g$ Noise. Gabor noise [144, 146] is a type of sparse convolution noise that obtains a better spectral control via the Gabor kernel, a multiplication of circular Gaussian envelope and a harmonic function [148]. We construct U3D$_g$ noise by extending 2D Gabor kernel to 3D Gabor kernel (adding the temporal dimension t):

$$g(x, y, t) = Ke^{-\pi\sigma^2(x^2+y^2+t^2)} \cos\left[2\pi F(x' + y' + t')\right] \tag{4.5}$$

where $x' = x \sin\theta \cos\omega, y' = y \sin\theta \sin\omega, t' = t \cos\theta$; K and σ are the magnitude and width of the Gaussian envelope; F and (θ, ω) are the magnitude and orientation angles of the frequency in the harmonic function. Then, we derive the noise $\mathcal{N}(x, y, t)$ with the sparse convolution and 3D Gabor kernel:

$$\mathcal{N}(x, y, t) = \sum_k g(x - x_k, y - y_k, t) \tag{4.6}$$

where the point set $\{\forall(x_k, y_k, t)\}$ are a set of sampled pixel points in the same frame t with Poisson distribution. Furthermore, to model the isotropy of the Gabor noise [144], we realize the two frequency orientations (θ, ω) as random variables (θ_i, ω_i) uniformly distributed in $[0, 2\pi]$. Then, the updated U3D$_g$ noise is given as below:

$$\mathcal{N}_g(x, y, t) = \sum_i \mathcal{N}(x, y, t; (\theta_i, \omega_i)) \tag{4.7}$$

U3D$_g$ Parameters. Similar to U3D$_p$, we denote the following parameter set as S_g for U3D$_g$ with Equation 4.5 and 4.7:

$$S_g = \{K, \sigma, F\} \tag{4.8}$$

We synthesize the procedural noise to construct the U3D perturbations, whose low-frequency patterns and low computational overhead can greatly advance the attacks. Formally, given the U3D noise function $\mathcal{N}$ and the parameters s, the generated U3D perturbation ξ of length T will be:

$$\xi = \{\mathcal{N}(t; s) | t \in [0, T - 1]\} \tag{4.9}$$

If T is less than the video length, ξ will be circular. Note that T works as a pre-specified parameter. For simplification, we use $\xi = \mathcal{N}(T; s)$ to represent Equation 4.9. Next, we will present how to calibrate U3D perturbation to achieve the design goals.

4.4.2 Calibrating U3D Perturbations.

4.4.2.1 Improving Transferability in Feature Space. U3D aims to deviate the intermediate layer's features, which could improve the transferability of the attacks. Large distance between the original and perturbed videos' features at intermediate layers of the DNN model can result in relatively high deviations in the final results. This will increase the probabilities on false learning by the unknown target DNN model and videos.

Specifically, we formally define $f_L(\cdot, d)$ as the truncated DNN model function, which outputs the intermediate feature of the input video at layer $L_d, d \in [1, M]$ of the DNN model $f(\cdot)$, M is the number of DNN layers. Then, $f_L(v, d), f_L(v', d)$ are denoted as the intermediate features of the original video v and perturbed video v', respectively. Thus, we have the ℓ_2-norm distance between the feature representations of the original video v and perturbed video $v' = v + \xi$ at layer d of the DNN as:

$$\mathcal{D}(v, v'; d) = ||P(f_L(v, d)) - P(f_L(v', d))||_2 \tag{4.10}$$

where $P(z) = sign(z) \odot |z|^\alpha$ is a power normalization function $\alpha \in [0, 1]$ and $\odot$ is the element-wise product [149].

Then, we maximize the distance $\mathcal{D}(v, v'; d)$ between the original and perturb videos over all the intermediate feature space as our attack objective function:

$$\max_{\xi} : \sum_{d \in [1,M]} \mathcal{D}(v, v + \xi; d) \tag{4.11}$$

4.4.2.2 Mitigating Boundary Effect. Recall that the boundary effect may potentially degrade the attack performance due to the misalignment between the adversarial perturbation and the input video. To tackle such issue, we introduce a temporal transformation function $\texttt{Trans}(\cdot)$ for the U3D perturbation with a shifting variable denoted as τ. Specifically, given a U3D perturbation ξ of length T, then $\texttt{Trans}(\xi; \tau)$ represents the U3D perturbation ξ temporally shifted by $\tau \in [0, T-1]$. Then, we maximize the expectation of the feature distances with all the T possible temporal shift transformation $\tau \in U[0, T-1]$ for U3D perturbation ξ (U denotes the uniform distribution):

$$\max_{\xi} : \mathop{\mathbb{E}}_{\tau \sim U[0,T-1]} [\sum_{d \in M} \mathcal{D}(v, v + \texttt{Trans}(\xi, \tau); d)] \tag{4.12}$$

To achieve such objective, we can consider all the possible transformed U3D perturbation (the size of transformation will be T) uniformly shifted with $\tau \in [0, T-1]$ (step 1 frame by frame in the video). τ will be sampled in the corresponding algorithm. Then, our U3D attack can learn a generic adversarial perturbation without the boundary effect, which can be injected into the streaming video anytime.

4.4.2.3 Improving Universality with Public Videos. Another goal is to find a universal perturbation learned from a relatively small set of videos, which can effectively perturb the unseen videos for misclassification. Denoting a set of public videos as V, the optimization integrates the universality maximization on videos in V (and ℓ_∞-norm bound) as below:

$$\max_{\xi} : \mathop{\mathbb{E}}_{v\sim V,\tau\sim U[0,T-1]}[\sum_{d\in M}\mathcal{D}(v, v+\texttt{Trans}(\xi,\tau); d)] \\ s.t.\ \ \xi = \mathcal{N}(T; s),\ ||\xi||_\infty \leq \epsilon \tag{4.13}$$

4.4.3 Optimizing and Generating U3D Perturbations. Since the U3D perturbation ξ can be efficiently generated if the U3D parameter set $\mathcal{S}$ is pre-computed, Equation 4.13 will optimize the attack w.r.t. $\mathcal{S}$. To search the optimal U3D parameter set $\mathcal{S}$), we solve it with the Particle Swarm Optimization (PSO) method [150]. Specifically, the parameter values in $\mathcal{S}$ are viewed as the particles' positions, and the set of parameter ranges can be constructed as the search space. Then, the objective function (Equation 4.13) is the fitness function $\mathcal{A}(f, V, \mathcal{N}(T; \vec{s}))$, where $\vec{s}$ is the current position for U3D parameter set S in the iterations.

In the initialization phase, m points will be randomly selected from the searching space for $\mathcal{S}$ while satisfying ℓ_∞-norm bound. Then, in the iterations, every particle will iteratively update its position by evaluating the personal and group best location (determined by the output of the fitness function). Notice that, before fed into the fitness function, the algorithm validates if the U3D perturbation ξ generated by the parameter set $\vec{s_i}^{k+1}$ satisfies ℓ_∞-norm bound ϵ or not. Finally, we can get the near-optimal parameter set s^*. Then, we generate the U3D perturbation $\xi = \mathcal{N}(T; s^*)$.

More specifically, we first define the fitness function $\mathcal{A}(f, V, \mathcal{N}(T; \vec{s_i}))$, detailed in Algorithm 11. Then we aim to find the optimal particle position (i.e., U3D parameters value) for such fitness function. Algorithm 10 demonstrates the detailed process of U3D optimization. At the beginning, a swarm of m particles denoted as $S = \{\vec{x_1}^k, \vec{x_2}^k, \ldots, \vec{x_m}^k\}$ will be initialized. For each iteration k, each particle i holds a position $\vec{x_i}^k = [x^k_{i,1}, x^k_{i,2}, \ldots, x^k_{i,d}]$, where d is the dimension of the searching parameter space and $x_{i,j}, j \in [1, d]$ indicates the parameter value of jth dimension. To update its position $\vec{x_i}^{k+1} = \vec{x_i}^k + \vec{v_i}^k$, each particle i compute with its current velocity

$\vec{v_i}^k = [v_{i,1}^k, v_{i,2}^k, \ldots, v_{i,d}^k]$ as the following equations:

$$v_{i,j}^{k+1} = W * v_{i,j}^k + c_1 * r_1(s_{i,j} - x_{i,j}^k) + c_2 * r_2(s_{g,j} - x_{i,j}^k) \tag{4.14}$$

$$x_{i,j}^{k+1} = v_{i,j}^k + x_{i,j}^k \tag{4.15}$$

where (1) $s_{i,j}$ is the value for kth dimension of the best solution searched via particle i so far; $S_i = [s_{i,j}], j \in [1, d]$ is called personal best; (2) $s_{g,j}$ is the value for kth dimension of the best solution in the Swarm S so far; $S_g = [s_{g,j}], j \in [1, d]$ is called leader. Note that every particle can use the S_i (local information) and S_g (social information) to iteratively update its velocity and position. $c_1, c_2 \in \mathbf{R}$ are weights for quantifying the impacts of the personal and social best solution correspondingly; r_1, r_2 is uniformly distributed values of range $[0, 1]$ which represents of randomness in the search. $W = \{W_s, W_f, W_e\}$ is called inertia weight, which can control the impacts of the previous velocity on the current iteration, and then influence searching ability. W will be decreased with every iteration via the following equation: $W = W - \frac{W_s - W_e}{k * W_f}$, where W is initialized as W_s and ended as W_e.

Comparison of PSO with Other Meta-Heuristic Algorithms. We have evaluated PSO by benchmarking with genetic algorithms [151], simulated annealing [152], and Tabu search [153]. PSO slightly outperforms them for U3D optimization. We use the C3D model as the public DNN model and randomly sample 500 videos from the HMDB51 dataset as the public dataset. The ϵ is set as a small bound 8. The parameter of the normalization α is set to 0.5. Table 4.1 shows the specified value ranges of the parameters for U3D_p and U3D_g, respectively. As for PSO, we set up the parameters as follows: (1) swarm size $m = 20$; (2) individual and social weight $c_1 = c_2 = 2$; (3) inertia weight $W = \{1.2, 0.5, 0.4\}$; (4) maximum iteration times $h = 40$.

Input: U3D function $\mathcal{N}(\cdot)$, DNN model f, video dataset V, ℓ_∞-norm bound ϵ, search space $\mathcal{X}$ for U3D perturbation parameter $\mathcal{S}$; PSO model: inertia weight $W = \{W_s, W_f, W_e\}$, individual/social weights c_1, c_2, swarm size m, maximum iteration number h

Output: optimal parameter set $\mathcal{S}^*$

// each node has $||(\mathcal{N}(T; \vec{s_i}))||_\infty \leq \epsilon$

1 $\mathcal{X}_{sample} \leftarrow$ randomly sample m points from $\mathcal{X}$

2 **for** *each* $\vec{s_i} \in \mathcal{X}_{sample}$ **do**

3 Call Algorithm 11: $\mathcal{A}(f, V, \mathcal{N}(T; \vec{s_i}))$

4 Set personal best of each particle $S_i \leftarrow \vec{s_i}$

5 Find the leader S_{gb}

6 Initialize $\vec{s_i}^k \leftarrow \vec{s_i}, i \in [1, m]$

7 **while** $k = 1 \leq h$ **do**

8 **for** $i \in [1, m]$ **do**

9 Update velocity and position per Equation 4.14, 4.15

10 Repeat Line 2-4

11 Update S_{gb} if leader changes

12 Update inertia weight W

13 **return** S_{gb} as S^*

Algorithm 10: `NoiseOpt`(f, V)

Table 4.1. U3D parameters setting

U3D$_p$			U3D$_g$		
$\lambda_x, \lambda_y, \lambda_t$	Λ	ϕ	K	σ	F
$[2, 180]$	$[1, 5]$	$[1, 60]$	$[1, 5]$	$[1, 20]$	$[0.25, 20]$

Then, we compare PSO with genetic algorithm (GA) [154], simulated annealing (SA) [152], and Tabu search (TS) [153] on tuning the U3D parameters. For GA, we set the number of chromosomes to be 20 (same as the number of PSO's particles) with the combination of tournament selection with a 50% uniform crossover probability [154] and mutation rate 0.5%. For SA, we set the initial temperature is 5000, and cooling factor 0.99. For TS, the tabu list size is set to 4. We implement the four methods for both $U3D_p$ and $U3D_g$ on the 500 videos, which are repeated 5 times and averaged for the final results. Table 4.2 illustrates their experimental results. We can observe that the PSO-based method is efficient, and also slightly outperforms GA, SA and TS on attack performance, e.g., PSO improves 1.7% over GA and 3.3% over SA for $U3D_p$. Besides, the MSE of both U3D perturbations are below 20 (very minor distortion out of 255^2 in the scale).

Table 4.2. PSO vs. GA, SA and TS (learning U3D parameters offline) for $U3D_p$ and $U3D_g$ (success rate "SR").

Method	$U3D_p$			$U3D_g$		
	Time (s)	SR	MSE	Time (s)	SR	MSE
PSO	847	88.7%	15.3	789	89.6%	16.0
GA	1,481	87.0%	14.6	1,164	89.4%	17.8
SA	1,976	85.4%	16.9	2,267	87.7%	13.7
TS	1,039	86.5%	17.5	822	88.1%	15.8

4.5 Experiments

4.5.1 Experimental Setup. We introduce our experimental setting, including the datasets, target models and benchmarks.

Datasets. We use three widely used datasets for video recognition to validate the

proposed U3D attack.

- HMDB51 [133] dataset includes 6,766 video clips (30 fps) in 51 different actions, e.g., fencing, climb and golf.
- UCF101 [155] dataset includes 13,320 video clips (25 fps) in 101 different actions, e.g., archery, and punch.
- UCF Crime [96] dataset includes 1,900 long surveillance videos (30 fps) collected from Youtube and Liveleak, in 13 anomalies, e.g., accident, explosion, and shooting.

The HMDB51 and UCF101 datasets are used for video classification, and the UCF Crime dataset for anomaly detection.

Target DNN Models. We evaluate the U3D attack on two common DNN models for video recognition: (1) C3D model [124]; (2) I3D model [96]. We also implement two video recognition techniques based on both C3D and I3D: (1) video classification [124]; (2) video anomaly detection [96] identifying anomalies by scoring the video segments in sequence.

Note that we choose C3D and I3D as the main evaluation models to show the attack performance due to the popularity and practicality in the video recognition systems (as depicted in Section 4.2.1). To fully evaluate the *transferability* of the U3D attack, we choose three more video classification models, including LRCN [127], DN [126] and TSN [128], and evaluate the U3D attack across five different DNN models.

Benchmarks. We use the following baseline adversarial perturbations: (1) Gaussian Noise: $\xi_g \sim \mathcal{N}(0, \sigma^2)$ and σ=0.01; (2) Uniform Noise: uniformly sampled noise

$\xi_u \sim [-\epsilon, \epsilon]$; (3) Random U3D: applying U3D without calibration by randomly choosing parameters. For the above three methods, we repeat each experiment 10 times, and return the average value; (4) The state-of-the-art video attacks, C-DUP [111] (as a *white-box* universal attack), V-BAD [113] and H-Opt [112] (both as *non-universal* black-box attacks).

Since V-BAD [113] and H-Opt [112] are non-universal, they might be incomparable with U3D and C-DUP on attacking a specific target (though their success rates are claimed to be high in such cases). It might also be unfair to compare U3D and C-DUP with V-BAD and H-Opt on transferability since the latter two are not designed towards that goal.

4.5.2 Attack Performance. We first evaluate U3D_p and U3D_g generated with a surrogate C3D model to attack unknown target models on different datasets. Specifically, we randomly select 500 videos from the HMDB51 dataset (retaining a similar distribution for classes as the full dataset) as the public video set (V), and consider the full UCF101 and UCF Crime datasets as the target set. We set $\epsilon = 8, T = 16$ and report the attack results on the target set. Note that the MSE of all the U3D perturbations are below 20 (very minor distortion out of 255^2 in the scale).

Table 4.3 lists the results for applying U3D_p and U3D_g to attack unknown models and videos (in different datasets). The U3D perturbations are injected into both UCF101 (for *video classification*) and UCF Crime (for *anomaly detection*) datasets, which are then inferred by both C3D and I3D. Such black-box attacks are realized by the *transferability* and *universality* of U3D (which will be thoroughly evaluated in Section 4.5.3). Table 4.3 also includes the attack performance of Gaussian, Uniform, Random, and C-DUP [111] (see the setting in Section 4.5.1). For both U3D and benchmarks, we apply the perturbations to full UCF101 and UCF Crime datasets.

Table 4.3. U3D vs. benchmarks (success rates; C3D/HMDB51 as surrogate; C3D/I3D and UCF101/UCF Crime as target).

Model / Noise	C3D		I3D	
	UCF101	UCF Crime	UCF101	UCF Crime
Gaussian	10.2%	15.3%	9.1%	12.6%
Uniform	5.3%	9.1%	1.7%	2.4%
Rnd. U3D	43.2%	52.6%	40.3%	51.8%
C-DUP [111]	80.2%	83.6%	54.4%	45.8%
$U3D_p$	82.6%	92.1%	80.4%	87.1%
$U3D_g$	85.4%	93.4%	82.9%	90.2%

Both $U3D_p$ and $U3D_g$ achieve high success rates on the two DNNs. For C3D, $U3D_p$ achieves 82.6% on the UCF101 dataset (video classification) and 92.1% on the UCF Crime (anomaly detection) while $U3D_g$ obtains a slightly higher success rate, i.e., 85.4% on the UCF101, and 93.4% on the UCF Crime. This can also show that our U3D perturbations effectively attack to other different DNN models on different datasets, e.g., HMDB51 and C3D $\rightarrow$ UCF Crime and I3D.

However, the benchmarks cannot achieve satisfactory attack performance. Injecting random noise (Gaussian and Uniform) to videos can only give 2.4%-15.3% success rates in all the experiments. Random U3D (random parameters without optimization) performs better but still not satisfactory (35.7%-52.6%). C-DUP [111] returns worse success rates on both C3D and I3D, even in the white-box setting. Since it is designed for attacking C3D, its performance on I3D is even worse (54.4% on UCF101 and 45.8% on UCF Crime, low transferability).

Finally, both U3D$_p$ and U3D$_g$ can perform very well (90%+) on anomaly detection regardless of the target models. We observe that anomaly detection is more vulnerable than video classification under the same perturbation, e.g., U3D$_p$ (92.1%>82.6%). The possible reason is that such DNN models have an extra computing model to output anomaly scores, which may make it more sensitive to perturbations.

4.5.3 Transferability and Universality.

Transferability. *Transferability* refers to the perturbations designed for one classifier can also attack other classifiers (cross-model) [101]. To study the transferability, we first define the *transfer rate* (TR) as the percent of the adversarial examples which deviates one model f_{srg} (e.g., a public DNN model as the surrogate model) and also deviate the target model f_{tar} (black-box). We denote $f_{srg} \rightarrow f_{tar}$ as the transferability of the attack from surrogate model to target model.

To evaluate the transferability, we choose C3D, I3D and other three more video classification models as surrogate/target models: DN [126], LRCN [127], and TSN [128], all of which are already fine-tuned on the UCF101 dataset. Then, we compute U3D$_p$ and U3D$_g$ ($\epsilon = 8$) with the surrogate model (as surrogate) and apply the U3D perturbations to craft adversarial examples on the UCF101 dataset, which are fed into the target models. We generate the U3D perturbations with 10% of the UCF101 dataset (randomly picked for each class), and select all the adversarial examples (crafted on the 90% of videos for target dataset) which can successfully fool the surrogate models. Then, we examine the percent of such adversarial examples that can fool the target model.

Table 4.4 presents the transfer rates of both U3D$_p$ and U3D$_g$. We can observe that all the attack can achieve high transfer rates (over 80%). This shows that our U3D perturbations achieve good transferability across these models. For example,

Table 4.4. Transferability: transfer rate (TR) on UCF101.

Noise	f_{srg} \ f_{tar}	C3D	I3D	DN	LRCN	TSN
$U3D_p$	C3D	–	93.4%	92.7%	85.0%	87.2%
	I3D	89.7%	–	96.3%	88.7%	85.0%
	DN	84.0%	83.2%	–	85.5%	83.4%
	LRCN	85.8%	87.2%	92.4%	–	86.1%
	TSN	85.5%	82.5%	89.3%	87.5%	–
$U3D_g$	C3D	–	87.0%	93.2%	86.3%	85.3%
	I3D	88.2%	–	97.4%	85.2%	86.0%
	DN	82.6%	81.4%	–	83.7%	85.6%
	LRCN	81.2%	83.4%	88.6%	–	84.5%
	TSN	86.2%	83.6%	90.2%	86.4%	–

$U3D_p$ can obtain 92.7% transfer rate for C3D→DN and 92.4% for LRCN→DN. We repeat the same set of experiments on the HMDB51 (Table 4.5) and UCF Crime datasets (4.6). High cross-model transferability for U3D can also be observed from such experiments.

Universality. *Universality* refers to the perturbations learnt from one dataset can perturb many different datasets (cross-data) to fool the DNN models [103]. To evaluate the universality of U3D, we first randomly pick 500 videos as the surrogate video set (denoted as X) from each of the three datasets, HMDB51, UCF101 and UCF Crime, respectively (retaining a similar class distribution as the full datasets). Then, we compute the U3D perturbations ($\epsilon = 8$) on X with the C3D model and evaluate

Table 4.5. Transferability: transfer rate (TR) on HMDB51.

Noise	f_{srg} \ f_{tar}	C3D	I3D	DN	LRCN	TSN
$U3D_p$	C3D	–	92.0%	89.5%	83.2%	84.6%
	I3D	87.6%	–	91.2%	82.5%	81.4%
	DN	82.3%	81.6%	–	84.5%	80.5%
	LRCN	85.0%	85.4%	95.3%	–	85.6%
	TSN	82.5%	85.7%	88.0%	84.2%	–
$U3D_g$	C3D	–	86.6%	96.4%	81.4%	83.1%
	I3D	92.2%	–	96.0%	88.3%	84.5%
	DN	82.0%	84.8%	–	87.5%	82.3%
	LRCN	85.6%	83.4%	88.2%	–	82.9%
	TSN	84.6%	81.2%	86.1%	84.2%	–

the attack success rates on all the three datasets (as the target dataset Y). For the same dataset case (intra-dataset attack), the target set Y excludes X to evaluate the universality. All the results are listed in Table 4.7.

We can observe that the U3D achieve 80%+ success rates for all the cases ($X \rightarrow Y$ within the same dataset or across different datasets). The diagonal results are higher than other cases, which shows that the U3D can perform well among the unseen videos in the same dataset. Moreover, for the same U3D perturbation, e.g., $U3D_g$, the success rate of UCF Crime→UCF101 is lower than that of HMDB51→UCF101 (81.6%<85.4%). Similar results are also observed from UCF Crime→HMDB51 and UCF101→HMDB51 (82.2%<85.0%). Since HMDB51 and UCF101 consist of human

Table 4.6. Transferability: transfer rate (TR) on UCF Crime.

Noise	f_{srg} \ f_{tar}	C3D	I3D	DN	LRCN	TSN
$U3D_p$	C3D	–	91.5%	94.1%	92.3%	89.0%
	I3D	90.7%	–	94.5%	90.2%	89.1%
	DN	87.2%	87.4%	–	88.2%	90.7%
	LRCN	92.8%	87.2%	92.4%	–	86.1%
	TSN	91.7%	90.2%	93.4%	91.7%	–
$U3D_g$	C3D	–	87.0%	93.2%	86.3%	85.3%
	I3D	91.3%	–	93.4%	90.2%	89.0%
	DN	89.3%	88.4%	–	93.4%	89.5%
	LRCN	93.2%	90.8%	91.2%	–	90.4%
	TSN	89.4%	88.6%	92.4%	89.5%	–

action videos while UCF Crime includes surveillance videos for anomaly detection, U3D perturbations learned on UCF Crime will exhibit less universality to HMDB51 and UCF101. Thus, selecting different surrogate videos can slightly help tune the attack performance on different target models and videos.

We repeat the same set of experiments for I3D as surrogate (Table 4.8). Note that C-DUP [111] (as a *white-box* attack only on C3D) has low transferability (in Table 4.3), and V-BAD [113] and H-Opt [112] (both as *non-universal* attacks) have low transferability.

4.5.4 Hybrid Black-Box Attack with Queries over Target Model. U3D

Input: public DNN model f, public video dataset V, current U3D noise parameters $\vec{s}$, sample times I

Output: Output value r of fitness function

Initialize $r \leftarrow 0$

$\xi \leftarrow \mathcal{N}_p$

for $v_i \in V$ **do**

    $t \leftarrow 0$

    `// Sample I times`

    for $i \in I$ **do**

        $\tau \leftarrow U[0, T-1]$

        $t \leftarrow t + \mathcal{D}(v_i, v_i + \texttt{Trans}(\xi, \tau); d)$

    $r \leftarrow r + \frac{t}{I}$

$r \leftarrow \frac{r}{|V|}$

return r

Algorithm 11: Attack Objective $\mathcal{A}(f, V, \mathcal{N}(T; \vec{s}))$

Table 4.7. Universality (success rate (SR); surrogate C3D).

Noise	X \ Y	HMDB51	UCF101	UCF Crime
U3D$_p$	HMDB51	87.3%	82.6%	92.1%
	UCF101	84.2%	88.4%	91.5%
	UCF Crime	80.1%	82.4%	96.0%
U3D$_g$	HMDB51	88.7%	85.4%	93.4%
	UCF101	85.0%	86.2%	90.2%
	UCF Crime	82.2%	81.6%	95.3%

is designed to universally attack different target models, and it has shown high transferability. If the attack performance is not very satisfactory for a new target model (though not found in our extensive experiments), we can extend the U3D to a hybrid black-box attack [123] by integrating queries over the target model $g(\cdot)$. Note that this still maintains attack universality on different target models. Thus, given the surrogate model $f(\cdot)$ (including a small set of public videos) and the target model $g(\cdot)$ available for queries, we can update the optimization for U3D by integrating the queries using input videos $v_1, \ldots, v_n$ (perturbed by ξ before querying):

$$\begin{aligned} \max_{\xi} : & \mathop{\mathbb{E}}_{v\sim V, \tau\sim U[0,T-1]} [\sum_{d\in M} \mathcal{D}(v, v + \texttt{Trans}(\xi, \tau); d)] \\ & + \omega \cdot \mathcal{Q}(g, v_1, \ldots, v_n, \xi) \\ s.t. \;\; & \xi = \mathcal{N}(T; s), \;\; \|\xi\|_\infty \leq \epsilon \end{aligned} \tag{4.16}$$

where the query oracle $\mathcal{Q}(\cdot)$ first derives the final classification results of the

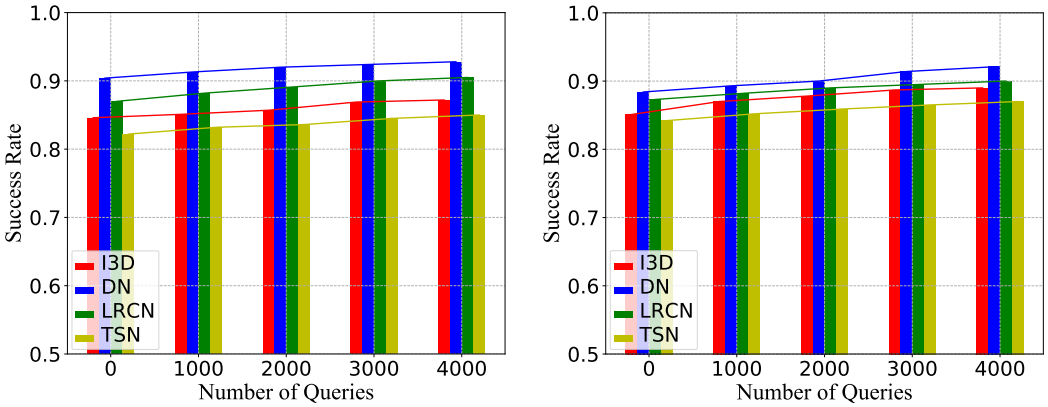
Table 4.8. Universality (success rate (SR); I3D surrogate).

Noise	X \ Y	HMDB51	UCF101	UCF Crime
$U3D_p$	HMDB51	89.6%	80.4%	87.1%
	UCF101	83.5%	86.3%	93.4%
	UCF Crime	80.1%	82.4%	96.0%
$U3D_g$	HMDB51	86.3%	82.9%	90.5%
	UCF101	82.5%	86.7%	91.7%
	UCF Crime	80.1%	84.3%	98.5%

perturbed videos $\{v_1 + \xi, \dots v_n + \xi\}$ by the target model $g(\cdot)$, and then returns the success rate for such videos. ω is hyperparameter for weighing the transferability of the surrogate model and queries (success rate) over the target model. Note that the adversary only needs to query the final classification (limited information) instead of the specific probability scores or logits information.

This optimization will search the U3D perturbations which can successfully fool the target model $g(\cdot)$ by both transferability and queries (hybrid). Similarly, after revising the fitness function with the new objective (Equation 4.16), we can apply PSO to compute the optimal U3D parameters.

To evaluate the hybrid attack, we choose the C3D as the surrogate model, and I3D, DN, LRCN, TSN as target models, respectively. Then, we follow the setting as the previous experiments on the UCF101 dataset (10% for learning U3D while 90% for target set) and U3D parameters. For the hybrid attack, we set the size of querying dataset as 50 (randomly chosen), $\epsilon = 8$ and $\omega = 10$, vary the number of queries as



(a) U3D$_p$ (b) U3D$_g$

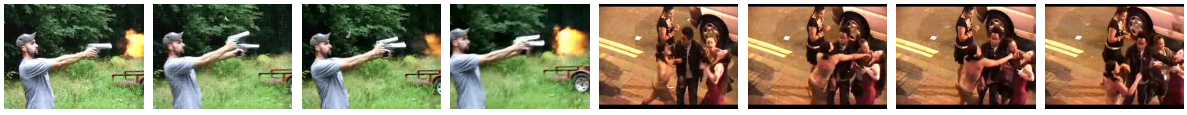
Figure 4.5. Hybrid black-box attack performance (surrogate C3D).

{0, 1000, 2000, 3000, 4000} ("0" means the original U3D attack without queries). Then, we apply PSO to optimize U3D perturbations on Equation 4.16, and report the success rate for both U3D perturbations against the four target models. Figure 4.5 shows that the success rates of both U3D perturbations slightly increase as the number of queries increases. The hybrid attack with additional queries to the target model can improve the transfer-based attack to some extent [123].

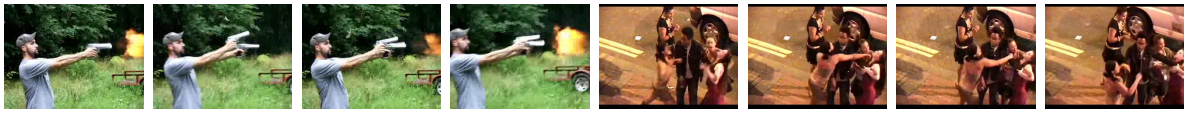
4.5.5 Visual Impact and Human-Imperceptibility.

Visual Impact. We arbitrarily select two videos, "shooting" and "fighting" to demonstrate the visual differences of the adversarial examples. Figure 4.6 presents a sequence of selected frames in two videos, and we can observe that the videos perturbed by U3D$_p$ and U3D$_g$ are much more human-imperceptible than C-DUP.

Human-Imperceptibility Study. We also conducted a human-imperceptibility study (with an IRB exempt) to validate if the perturbed videos could be visually discerned by humans compared with the original videos. We distributed the videos (original videos, adversarial examples based on U3D$_p$, U3D$_g$ and C-DUP) and an



(a) U3D$_p$



(b) U3D$_g$

(c) C-DUP [111]

Figure 4.6. Some selected frames in videos corresponding to "shooting" and "fighting": (a) perturbed by U3D$_p$, (b) perturbed by U3D$_g$ and (c) perturbed by C-DUP [111]. Both U3D$_p$ and U3D$_g$ show good human-imperceptibility compared with C-DUP.

online survey to 87 anonymous students.

Specifically, we randomly pick 870 videos from the UCF101 and UCF Crime datasets. To avoid bias on the distribution of adversarial examples, we randomly choose 435 videos to inject perturbations (218 videos by U3D$_p$ and 217 by U3D$_g$; all the adversarial examples are selected from the previous groups of experiments to successfully fool the video classifiers). The students do not know the distribution of adversarial examples. Then, 435 pairs of videos (the adversarial examples and their clean videos) are mixed with another 435 pairs of videos (the remaining 435 clean videos and their duplicates).

Next, we randomly assign 10 pairs of videos to each student. Each of the 87 students is expected to watch and annotate 10 pairs of videos ("visual difference" or "no visual difference"). We also ask the participants to identify the region of difference if they choose "visual difference". Finally, we collected 740 valid annotations on pairs of videos (74 students have submitted their results over a 1-week time frame; among

the assigned videos to these students, there exist 386 adversarial examples). We found that 96.4% (372) out of such 386 annotations are "no visual difference", while only 3.6% (14 annotations) of them are annotated with "visual difference". Among the remaining 354 pairs of original videos and their duplicates, there also exist 7 annotations submitted as "visual difference".

Meanwhile, we conduct the same group of studies for C-DUP by randomly picking another 870 different videos to inject perturbations (in order to avoid the bias on mutually linking the adversarial examples of U3D and C-DUP perturbed on the same videos).[6] As high as 85.4% (304 out of 356 valid adversarial examples of C-DUP) can be successfully identified by the same group of students. All the above results have indicated that our U3D perturbations achieve high human-imperceptibility (significantly better than C-DUP).

4.5.6 Practicality for the U3D Attack. We now discuss the possible attack vectors, and evaluate the U3D attack on a real system. Prior real-time video attack scenarios can also be used for U3D (e.g., manipulating the system via a pre-installed malware in C-DUP [111]). Besides them, we design extra ways for the real-time attack. Other adversarial attacks on videos (including future attacks) can also use our physical scenarios to inject real-time perturbations.

First, the network topology of the recent intelligent video surveillance system (VSS) [130, 131], include: (1) camera; (2) communication network; (3) server. Then, the adversary needs to inject the U3D perturbations in two cases: data-at-rest and data-in-motion [105]. The data-at-rest locally stores videos in the camera or the server. The data-in-motion transfers videos across the network or loads them into

[6]If any visual difference is identified by the student from the C-DUP perturbed video (or vice-versa), this may give him/her prior knowledge to identify visual difference from U3D (or vice-versa) since both are perturbed.

the volatile memory. Per the potential threats to VSS [131, 132], we consider the local infiltration to the systems in two scenarios: (1) malware; (2) man-in-the-middle (MITM) attack. First, malware can be locally installed via a malicious firmware update over the USB port. Moreover, the surveillance cameras could be sold through legitimate sales channels with the pre-installed malware [156]. Second, for the MITM attack, the adversary could access to the local network (e.g., by penetration) which connects to the camera and server, and behave like a normal user. Here, we take the MITM attack as an example.

Specifically, we setup a local camera-server network, where one PC works as the surveillance camera to continuously send video streams to another PC (as a server) using the real-time streaming protocol (RTSP). Then, we use the third PC as the adversary running `Ettercap` (https://www.ettercap-project.org/) with ARP poisoning to implement the man-in-the-middle attack (sniffing the network traffic). All three PCs use Ubuntu 18.04 OS, connected on a LAN. According to the recent survey on the security of IP-based video surveillance systems [132,157], a large number of unencrypted cameras (4.6 millions) are exposed to the network, e.g., using HTTP instead of HTTPS. Although the percentage of such unencrypted cameras is not disclosed, the unencrypted RTSP has been a major security vulnerability in video surveillance [132,157]. Thus, in our attack setting, we assume that the camera network is open with unencrypted RTSP. By exploiting the vulnerabilities, the adversary will target the camera-server communication without decryption and temporarily intercept the communication session by injecting the TEARDOWN request to the server. When the server tries to send a new request for the new communication session with the camera, the adversary will capture it, modify the delegated client port and forward the request to the camera. Finally, the adversary can receive video streams from the camera in real time.

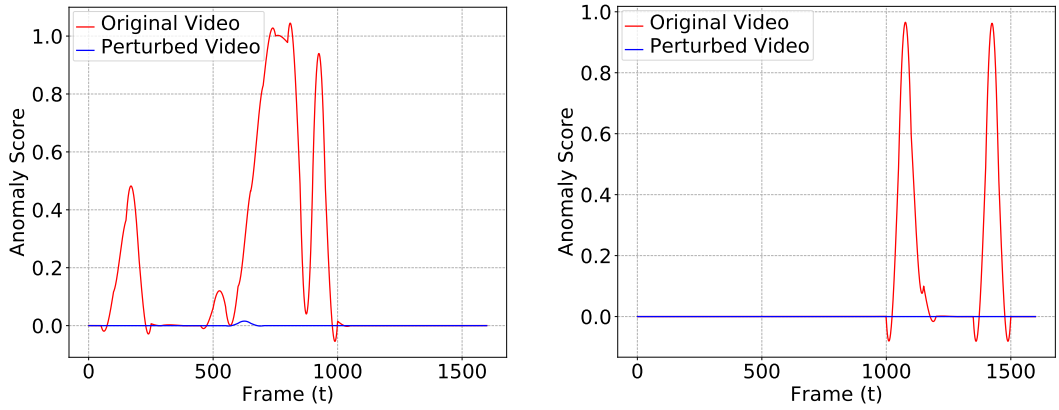
Note that we utilize the `FFmpeg` compiled with the video encoder `libx264` to execute the codec (decode and encode process) on the video from RTSP streams. Since our attack is performed through unencrypted video streams, there is no extra cost for decrypting video packets. To evaluate the computational overheads for the codec on the video streams, we set the following encoding parameters: (1) PRESET (encoding speed): "medium" by default; (2) bit rate: same as the streaming video bit rate ($\sim$ 350kbps) [158]. The average cost for the codec on the video is $\sim 0.3 < 1$ second, which will not affect the streaming video quality. It can also be accelerated by hardware, e.g., GPU. Overall, we have experimentally shown that the delay of our U3D attack (including both codec and injection) are negligible. Finally, the adversary can forward the perturbed video streams to the server for misclassifications.

Table 4.9. Amortized runtime (each frame) for attacking the streaming video on UCF101 and UCF Crime (in Seconds).

Video Name	Codec	Inject	Runtime	Video Name	Codec	Inject	Runtime
Bowling	0.010	0.004	0.014	Arson	0.010	0.004	0.014
BoxingPunchingBag	0.012	0.005	0.017	Assault	0.010	0.005	0.015
CliffDiving	0.010	0.005	0.015	Explosion	0.009	0.007	0.016
CuttingInKitchen	0.009	0.005	0.014	Fighting	0.009	0.006	0.015
HorseRace	0.010	0.004	0.014	Shooting	0.010	0.004	0.014

Evaluation. We randomly pick 10 videos (5 videos from each of UCF101 and UCF Crime) to evaluate the attack against the video classification and anomaly detection. For each video, we repeat 10 times while injecting 10 different U3D perturbations (pre-generated with the HMDB51 dataset and C3D) in the video streams. The attack success rate for classification is 88% (44/50) and for anomaly detection is 98% (49/50). Table 4.9 presents the amortized online time for processing each frame. All the runtimes are less than 1/30 (the frame rate is between 24fps and 30fps in experimental videos). Similar results on UCF Crime are also given in the table. Thus, U3D can

efficiently attack streaming videos with negligible latency.



(a) Road Accident ($U3D_p$) (b) Explosion ($U3D_g$)

Figure 4.7. Real-time attack on anomaly detection

Moreover, Figure 4.7 presents the real-time anomaly scores of two example videos (i.e., "Road Accident" and "Explosion"), where each streaming video (sent from the camera to the server) is perturbed by a U3D perturbation in real-time (w.l.o.g., $U3D_p$ for "Road Accident" and $U3D_p$ for "Explosion"). In Figure 4.7(a) ("Road Accident"), we can observe that there are three wave peaks in the original video, e.g., around frame 750, which will trigger the anomaly alarm (reporting "Road Accidents" if the score is greater than a pre-set threshold). While our U3D attack perturbs the streaming video, the anomaly scores of the perturbed video are reduced to almost zero in all the frames. The "Explosion" example (Figure 4.7(b)) also shows similar results. This illustrates that our U3D can perfectly compromise the video anomaly detection systems.

Finally, to validate the boundary effect-free property of the U3D attack on streaming videos, we conduct another group of experiments on the UCF101 and UCF Crime datasets. Note that the lengths of videos are at least 15 seconds. Then, for each input video in two datasets, we insert the U3D perturbation at 10 different times (from

Table 4.10. Success rates of U3D perturbations (boundary effect-free), injected at 10 different times for each video.

Model / Noise	C3D		I3D	
	UCF101	UCF Crime	UCF101	UCF Crime
$U3D_p$	81.2%	90.3%	80.2%	85.3%
$U3D_g$	84.5%	93.0%	82.6%	89.4%

0 to 5s with a step of 0.5s), and the classification and anomaly detection will start from the first perturbed frame to the end of the video. Table 4.10 summarizes the results for success rates in two applications. We can observe that our U3D perturbations can still achieve high success rates while the misalignment may occur, e.g., $U3D_p$ still achieves 81.2% on UCF101 against C3D, and $U3D_g$ achieves 93.0% on UCF Crime against C3D. This shows that U3D can mitigate the boundary effect well.

4.6 Evaluation against Defense Schemes

To our best knowledge, there are very few defense schemes against the adversarial attacks on videos (mostly on images). We comprehensively evaluate the performance of U3D against three major categories of state-of-the-art defense schemes, which are adapted towards video/U3D defenses. They include: (1) adversarial training [118,119]; (2) adversarial example detection [114,120]; (3) certified robustness [121,122].

Attack and Defense Setting. We use the $U3D_p$ and $U3D_g$ perturbations generated in Section 4.5.2 (surrogate C3D and HMDB51 dataset) to craft adversarial examples on a dataset (e.g., UCF101 or UCF Crime). The adversarial examples will be used to attack the target model (C3D or I3D), which will be adversarially trained, integrated into the detection schemes, or certified with randomization. In all the tables in this subsection, "Model" refers to the target model, and "Dataset" refers to the dataset

used to craft adversarial examples.

Adversarial Training. Adversarial training [118, 119, 159, 160] refers to the model training by adding adversarial examples into the training dataset. It has been empirically validated to be effective on improving the robustness against adversarial examples and maintaining accuracy on clean data.

First, due to the universality of U3D, we evaluate U3D attack on a universal adversarial training (denoted as "UAT") [119] which defends against universal perturbations. Specifically, such scheme adopts PGD-based adversarial training [118] to formulate a min-max optimization problem as below:

$$\min_{\theta} \max_{\xi} : \frac{1}{|X|} \sum_{(x_i, y_i) \in X} L(\theta; x_i + \xi, y_i) \quad s.t. \quad ||\xi||_\infty \leq \epsilon \tag{4.17}$$

where θ denotes the model parameters, $X = \{(x_i, y_i), i \in [1, |X|]\}$ is the training sample set, $L(\cdot)$ is the loss function, and the ℓ_p-norm of universal perturbation ξ is bounded by ϵ. Different from the conventional PGD-based adversarial training (computing the perturbation for each instance), the inner optimization problem seeks a universal (more precisely, batch X-universal) perturbation ξ to maximize the adversarial loss w.r.t. the sample set X. It has been shown to be effective against universal perturbations compared to PGD-based adversarial training [118]. In addition, it is more efficient to compute one universal perturbation across all the training iterations, i.e., only updating perturbation ξ once for each step [119].

Second, besides "UAT", we tailor the universal adversarial training towards U3D (denoted as "U3D-AT"), and evaluate the U3D under a *stronger defense setting* (see the white-box defense of G3 in Section 4.3.1). Specifically, the defender knows the U3D function $\mathcal{N}(\cdot)$ but does not know the specific values of the U3D parameters s.

Recall that the U3D perturbation is computed by optimizing Equation 4.13, which is formulated as a attack fitness function $\mathcal{A}(f, V, s)$. Then, we can also adapt the UAT framework by replacing the inner optimization objective with the U3D function as $\mathcal{A}(f, X, s)$:

$$\min_{\theta} \max_{s} : \mathcal{A}(f, X, s) \quad s.t. \;\; \xi = \mathcal{N}(T; s), \;\; ||\xi||_{\infty} \leq \epsilon \tag{4.18}$$

where the norm-bounded U3D perturbation ξ can be computed by the U3D function with the parameters s. Similar to UAT, we can iteratively update the best U3D perturbation among a batch of data (X) in the inner loop via `NoiseOpt`, which adapts PSO to find the optimal parameters.

For the experiments, we evaluate the defense performance of standard PGD-based adversarial training (denoted as "Normal"), universal adversarial training ("UAT") and our U3D-adaptive AT ("U3D-AT") against our U3D perturbations, respectively. We split the datasets (UCF101 and UCF Crime) into the training dataset (80%) and the test dataset (20%). We set the perturbation bound $\epsilon = 8$. For both UAT and U3D-AT, we set the batch size as 200. For UAT, we utilize FGSM to update ξ, and Momentum SGD optimizer to update model parameters as the original setting [119]. For the adversarially trained models, we evaluate Clean ACR – the predication accuracy on the clean data, besides the attack success rate (SR) for misclassification. Note that we also report the accuracy and SR of the normal models.

Table 4.11 (left) summarizes the results of the adversarial training against U3D on the UCF101 (see similar results on UCF Crime on the right.). The accuracy of both UAT and U3D-AT on the clean data declines since the training has included adversarial examples. Nevertheless, the success rates of both U3D_p and U3D_g have been reduced against both UAT and U3D-AT. The U3D-AT performs better than

the UAT, e.g., the attack SR of U3D$_p$ is 42.7%<67.4% on the C3D. This is because U3D-AT directly optimizes the defense on U3D (with the attack fitness function), which thus makes the model more robust against U3D. However, such U3D-AT is more like "white-box" defense in which the defender (model owner) already knows the adversary's strategy (e.g., U3D format and attack function). In practice, the defender usually cannot readily obtain such information.

Table 4.11. Adversarial training on UCF101

Model	Defense	Clean ACR	U3D$_p$ (SR)	U3D$_g$ (SR)
C3D	Normal	86.2%	83.7%	84.2%
	UAT	78.5%	67.4%	65.5%
	U3D-AT	77.2%	42.7%	45.3%
I3D	Normal	88.7%	82.1%	82.6%
	UAT	80.4%	70.2%	69.5%
	U3D-AT	78.6%	50.3%	47.4%

Adversarial Examples Detection. Most detection schemes [114, 120, 161–163] locally train a detector or utilize feature characteristics in adversarial examples to determine if the input is perturbed or not. For instance, a detector can be trained on both clean data and adversarial examples via adversarial training [120]. Although detection schemes have difficulties on mitigating adversarial attacks (e.g., Magnet [161] was broken by [141], and some recent defenses were broken by adaptive attacks [139]), we still evaluate our U3D against detection schemes (including that adapted to U3D). Note that the U3D attack can be both online and offline. Then, we evaluate both of them against the detection schemes (assuming that the offline detection can be executed with arbitrary runtime).

Table 4.12. Adversarial training on the UCF Crime

Model	Defense	Clean ACR	U3D$_p$ (SR)	U3D$_g$ (SR)
C3D	Normal	92.5%	91.6%	90.7%
	UAT	84.5%	74.7%	75.3%
	U3D-AT	82.4%	62.7%	65.3%
I3D	Normal	95.3%	88.4%	91.2%
	UAT	89.4%	76.2%	80.5%
	U3D-AT	86.2%	58.6%	59.5%

First, for the online detection, we choose AdvIT [114] which is effective against the existing adversarial attacks on *real-time* video recognition. It finds the inconsistency among the temporally close frames with the optimal flow information, assuming that perturbations can destroy the frame consistency to some extent. Specifically, given one target frame (to be detected), AdvIT first estimates the optimal flow between the target frame and previous k frames, and then reconstructs pseudo frames by applying the optical flow to the beginning frame. Finally, it would compute the inconsistency score c between the target frame and pseudo frames, where high inconsistency score indicates that the target frame is adversarial. To defend against the adaptive attacks, AdvIT applies Gaussian noise to fuzz the optical flow for generating the pseudo frames.

In the experiments, we randomly select 200 clean videos from the UCF101 and UCF Crime datasets (100 each), and apply both U3D$_p$ and U3D$_g$ perturbations to craft adversarial examples. We set the perturbation bound $\epsilon = 8$. For detection, we set $k = 3$ (which only slightly affects the detection rate) and utilize FlowNet [164] as the optical flow estimator in AdvIT. Then, we randomly select 5 frames in each video

as the target frames, and average the inconsistency scores (reporting detection when ≥ 1) to derive the detection results.

Table 4.13 summarizes the detection accuracy (DR) and false positive rate (FPR) of AdvIT. It shows that U3D can bypass the detection of the state-of-the-art detection scheme, even though AdvIT achieves low false positive rates. For instance, AdvIT only obtains 12% accuracy to detect $U3D_p$-based adversarial examples for the C3D. The results show that U3D is immune to the temporal consistency detection by AdvIT, since the U3D perturbations are constructed on continuous 3-dimensional noise, which can still retain the consistency in temporal space.

Table 4.13. Detection and false positive rates of AdvIT [114]

Model	Dataset	$U3D_p$		$U3D_g$	
		DR	FPR	DR	FPR
C3D	UCF101	12%	2%	18%	2%
	UCF Crime	12%	5%	19%	3%
I3D	UCF101	10%	3%	17%	3%
	UCF Crime	12%	5%	22%	3%

Furthermore, we have evaluated the Area Under Curve (AUC) values of the Receiver Operation Characteristic Curve (ROC) of AdvIT for U3D perturbations and other three benchmarks: C-DUP [111], V-BAD [113] and H-Opt [112]. The AUC metric represents the probability that the detector assigns a higher score to a random positive sample (adversarial example) than to a random negative sample (clean data) [114]. It can better measure the detection performance than the DR/FPR. Table 4.14 summarizes the results. From the table, we can observe that the AUC values of U3D are close to random guess, e.g., 54.2% ($U3D_p$) and 56.7% ($U3D_g$) on

Table 4.14. Detection AUC of AdvIT [114] against U3D, C-DUP, V-BAD, and H-Opt. C3D:1st/3rd row. I3D:2nd/4th row

Dataset	$U3D_p$	$U3D_g$	C-DUP	V-BAD	H-Opt
UCF101	54.2%	56.7%	97.2%	98.4%	99.2%
	56.4%	55.3%	98.7%	97.3%	98.6%
UCF Crime	61.2%	64.8%	97.6%	99.5%	98.3%
	55.6%	58.1%	97.4%	99.7%	99.8%

C3D and UCF101 while all the benchmarks can be almost fully detected by AdvIT (*all the AUC values are very close to 1*). This occurs since the temporal consistency cannot hold in the adversarial examples by C-DUP, V-BAD and H-Opt (perturbations are generated specific to the frames as frame-by-frame perturbations).

Second, for the offline detection, we evaluate the U3D against another recent work [120] based on the adversarial training [118]. If the universal adversarial training (UAT) can defend against U3D to some extent, we can also extend it to train a universal perturbation detector against U3D. Specifically, the *asymmetrical adversarial training* (AAT) [120] trains K detectors (for a K-class classification model) to detect adversarial examples. Given an input x, each detector $h_k, k \in [1, K]$ will output a *logit* score corresponding to the class label, which can determine if data is perturbed or not (see details in [120]). To defend against the U3D, we revise the K detectors $h_k, k \in [1, K]$ with the UAT by changing the training objective as below (denoted as "U3D-AAT"):

$$\begin{aligned} \min_{\theta}: \ & [\mathop{\mathbb{E}}_{x\sim D'_k} \max_{s} L(h_k(x+\xi),1) + \mathop{\mathbb{E}}_{x\sim D_k} L(h_k(x),0)] \\ s.t. \ & \xi = \mathcal{N}(T;s), \ ||\xi||_\infty \leq \epsilon \end{aligned} \tag{4.19}$$

The objective includes two parts: (1) the maximum loss of adversarial examples (by U3D perturbation ξ) on the out-of-class data samples D'_k; (2) the loss of intra-class natural data samples D_k. $L(\cdot)$ is a loss function, e.g., binary cross-entropy loss. For the inner optimization of the first part, we adopt similar procedures as U3D-AT to update the U3D perturbations (as depicted earlier).

Table 4.15. Detection and false positive rates of U3D-AAT.

Model	Dataset	U3D_p		U3D_g	
		DR	FPR	DR	FPR
C3D	UCF101	56.2%	6.3%	53.4%	5.9%
	HMDB51	44.5%	8.2%	47.4%	7.1%
I3D	UCF101	55.4%	4.2%	56.5%	5.1%
	HMDB51	52.6%	5.7%	54.3%	5.9%

To evaluate the performance of the detectors, we choose the action classification on the UCF101 and HMDB51 as K-Class problem ($K = 101$ and 51). Specifically, we split the training/testing datasets by 80%/20% for each category. We set the perturbation bound $\epsilon = 8$, and apply the two U3D perturbations to craft the adversarial examples, which are mixed up with the clean videos for detection (for instance, in UCF101 dataset, there are 2664 clean videos, 2664 videos perturbed by U3D_p, and 2664 videos perturbed by U3D_g). We adopt the *integrated classifier* which computes the estimated class label $c = f(x)$ with the original classifier f and computes a logit vector $h_c(x)$ using the corresponding detector h_c [120]. We report the detection accuracy (DR) and false positive rate (FPR). The results in Table 4.15 have shown that such universal adversarial detector can detect the U3D perturbations to some extent: the universal AAT detector can achieve about 50% detection rate while maintaining a low FPR

(less than 7%). Such FPR is reasonable considering there could still exist overlapped adversarial subspaces, i.e., U3D-AAT may not be trained to be perfect to learn U3D perturbations and thus separate the perturbed video and clean ones. However, training such AAT detectors should know the U3D attack (*white-box defense*), and it is only limited to defend against offline attacks due to the computational costs.

Certified Robustness. Recently, certified schemes [121, 122, 165–167] have been proposed to defend against norm-bounded adversarial perturbations with theoretical guarantees. We evaluate the U3D attack against two representative certified schemes: PixelDP [121] and randomized smoothing [122].

First, PixelDP [121] adopts the Gaussian mechanism of differential privacy to slightly randomize the image pixels. After injecting Gaussian noise, the small change of image pixels (adversarial perturbation) will not affect the classification results with some probabilistic bound (thus provide robustness guarantee for DNN models). It will be extended from protecting image DNN models to video DNN models.

To evaluate the U3D attack against PixelDP, we modify the video DNN models by placing the noise layer in the first convolutional layer under the same Gaussian mechanism setting [121] w.r.t. an ℓ_2 attack bound $L = 0.1$ (such setting ensures a high accuracy in [121]). We split training/test as 80%/20% for retraining the model. Note that PixelDP admits that the certified effectiveness against ℓ_∞ attacks is substantially weaker via empirical evaluations (which conforms to the performance of other certified schemes such as randomized smoothing). Then, we generate U3D perturbations bounded by ℓ_2 norm value of 0.5 (which indeed generates very minor perturbations in case of very high video dimensions).

We report the classification accuracy of PixelDP on clean videos, and the success rates of the U3D attack in Table 4.16. The accuracy of PixelDP drastically

Table 4.16. Accuracy (ACR) and success rate (SR) of PixelDP [121].

Model	Dataset	Clean ACR	$U3D_p$ (SR)	$U3D_g$ (SR)
C3D	UCF101	63.2%	83.4%	85.3%
	UCF Crime	65.9%	86.2%	89.7%
I3D	UCF101	65.8%	82.3%	79.4%
	UCF Crime	67.4%	84.7%	85.2%

declines after injecting Gaussian noises (vs. the baseline models), e.g., 86.2%→63.2% on C3D. Meanwhile, the U3D attack can still achieve high success rates in all the cases. This shows that PixelDP cannot defend against U3D since PixelDP only ensures a weak bound with the Gaussian mechanism of differential privacy.

Second, we also evaluate the certified robustness via randomized smoothing [122]. It provides a tight guarantee (based on the Neyman-Pearson Lemma) for any random classifier by smoothing inputs with an additive isotropic Gaussian noise. However, it only certifies ℓ_2 radius of the perturbation bound. The certified schemes via smoothing against ℓ_∞ have been shown to be ineffective as the input dimensionality d increases. The certified radius is bounded by $O(\frac{1}{\sqrt{d}})$ as $p > 2$ [165, 166].

To evaluate our U3D attack against such certified scheme, we first generate the optimal U3D perturbations ξ by changing perturbation bound ℓ_∞ in `NoiseOpt` to ℓ_2.[7] Specifically, we evaluate the accuracy of the smoothing classifier on the perturbed videos against U3D, which is the percentage of the perturbed videos to be

[7]Since randomized smoothing cannot certify defense against ℓ_∞ bounded attack for high dimensional inputs (e.g., videos) [165, 166], the U3D perturbations using ℓ_2 bound instead of ℓ_∞ are still effective against such scheme.

correctly classified (we also evaluate the accuracy on the clean videos as benchmarks). Furthermore, we also derive certificated radius R for the videos, which indicates that the classification results can be certified against any perturbation with ℓ_2-norm no greater than R (see [121]).

Next, we set the number of Monte Carlo samples as $n = 100/1000$ and failure rate $\alpha = 0.001$. The failure rate indicates that the robust classifier can have 1-α confidence to return the classification result. We set the Gaussian variance $\sigma = 0.25$ (same as [121]), and the radius bound for U3D perturbations as $\epsilon = 0.5$ (which generates minor perturbations in case of high video dimensions). We report the accuracy and average certified radius in Table 4.17 (see similar results in Table 4.18). The results show that the randomized smoothing cannot defend against the U3D attack under ℓ_2-norm perturbations (can only certify very small radius), and the robust classifier only achieves less than 70% accuracy on the clean video samples.

Table 4.17. Accuracy and radius of Randomized Smoothing [122] on UCF101

Model	n	Clean	$U3D_p$		$U3D_g$	
		ACR	ACR	Radius	ACR	Radius
C3D	100	67.4%	14.5%	0.23	15.2%	0.19
	1000	68.2%	16.2%	0.24	18.4%	0.25
I3D	100	71.5%	21.7%	0.32	19.8%	0.28
	1000	72.2%	25.2%	0.37	21.2%	0.26

4.7 Mitigation of U3D Perturbations

The experiments show that the adversarial training (AT) is still the state-of-the-art on improving the model robustness regardless of overheads. The universal

Table 4.18. Accuracy and radius of Randomized Smoothing [122] on UCF101

Model	n	Clean	$U3D_p$		$U3D_g$	
		ACR	ACR	Radius	ACR	Radius
C3D	100	70.6%	26.3%	0.26	25.1%	0.22
	1000	71.2%	30.2%	0.21	32.4%	0.23
I3D	100	74.6%	28.3%	0.23	25.6%	0.20
	1000	75.1%	29.2%	0.25	26.4%	0.18

AT and AT adapted to U3D (U3D-AT) have shown some effectiveness on reducing the attack success rates (though the accuracy on clean videos has been reduced). To further improve the performance of AT against U3D, we can enhance the search in a larger U3D perturbation space. Also, we can integrate the adaptive inference method, e.g., applying stochastic interpolation to reduce the effect of U3D [168], and certified robustness [122].

For detection methods, the properties of the procedural noise (e.g., low frequency texture structure) can be utilized. For instance, since the background scenes in most surveillance videos captured by static cameras do not change, the defender can extract the static frame of the background and compare it with the perturbed video frame(s) to check the possible perturbation. An alternative way is to check the moving objects or humans in the videos. Since the U3D perturbations applied to the same object in different frames are likely to be different due to the changed coordinates, the deviations between the perturbed object in different frames might be identified. This needs other object detection/tracking algorithms, which may only be suitable for offline analysis due to high overheads.

Furthermore, although certified robustness cannot defend against the U3D, it is promising since it provides theoretical guarantee against norm-bound perturbations. One potential method to improve is the integration of randomized smoothing with UAT, which could potentially make the trained model robust against more unknown perturbations and thus improve the robust accuracy. However, this also poses challenges on expensive training (not model-agnostic either). We should also address the high dimensionality of videos since the certified guarantee can be jeopardized drastically on high dimensional data under ℓ_∞ bound. Thus, we can execute transformation to reduce the dimension of input data (e.g., by autoencoder) while certifying the robustness after transformation. We will explore these in the future.

Last but not least, we can also mitigate the online U3D attacks by enhancing the security of video recognition systems, e.g., upgrade to encrypted communication channels or add watermarking to the video streams (to detect injections).

4.8 Related Work

Security in machine learning, especially the vulnerabilities of AI systems to the adversarial inputs, has been intensively studied in both security and machine learning communities. Since adversarial examples were introduced [101, 159, 169], there have been numerous works on attacking image classifiers. For instance, FGSM [159], PGD [118], UAP [103] and many others [102, 137, 170], work well in the white-box setting. For black-box attacks, researchers have proposed two main types of methods: transfer-based [115–117] and query-based attacks [134–136, 171]. Recently, a hybrid attack [123] combines both of them to improve the attack performance. Moreover, adversarial attacks emerge in voice recognition [107, 108], malware classification [172], text understanding [106], etc.

Recent research has extended adversarial attacks from attacking DNNs on 2-D

images to 3-D videos [110–113]. Wei et al. [112] proposed a heuristic algorithm based on the query-based optimization attack [171] to search the saliency region in the video frames for perturbation. V-BAD [113] utilizes natural evaluation strategy (NES) [136] to query the target model for estimating gradient, and then craft adversarial examples via PGD. Both methods compute the perturbation for each frame, which requires heavy computational overheads. They cannot attack real-time videos (due to lack of universality either). C-DUP [111] applies GAN to generate universal perturbations offline and attack real-time video classification. However, it is a white-box attack, which is also limited to only the C3D model. More importantly, due to lack of consistency in the perturbations across frames, all these three attacks can be directly mitigated by AdvIT [114] with high accuracy.

To defend the model against adversarial attacks, a wide range of defense schemes [114, 118–120, 122, 142] have been proposed, which aim to either improve the robustness of model or detect adversarial examples. To our best knowledge, existing defense schemes (e.g., [118, 121]) mainly work on images, and have not empirically studied videos. Instead, we have thoroughly evaluated our U3D attack by redesigning current defense schemes in Section 4.6, which show some effectiveness against the U3D attack on videos. We anticipate that our U3D can motivate to build more robust defense schemes for DNN-based video recognition.

CHAPTER 5

STEALTHY POISONING ATTACK ON VIDEO CLASSIFICATION

5.1 Introduction

Deep neural networks (DNNs) have been extensively studied in different domains, especially video recognition, such as self-driving [142], action recognition [124] and anomaly detection [96]. However, DNNs have been proven to be vulnerable to adversarial attacks. Evasion attacks [169] were first proposed to craft adversarial examples to deviate the learning models [104, 105, 137, 159, 173, 174]. [8]

Different from the aforementioned evasion attacks during inference phase, data poisoning attacks [176–181] target the training phase of machine learning models, where the adversaries aim to inject poisoned data instances into the training dataset and thus degrade the performance of the model trained with such poisoned dataset. For example, a classic form of data poisoning attacks [177, 180] aims to enforce the trained model to misclassify a particular set of inputs. Recently, another form of poisoning attacks [176, 181–184] can pose a more sophisticated threat to the DNN models, i.e., the attacker can inject the poisoned data generated with a small trigger pattern and then set up a link between the trigger with a target label (installing backdoor). Thus, the trained DNN model on the poisoned data will consistently misclassify the data involving the trigger to the target label while still making correct classification on the clean data. For example, a sticker on the road sign can effectively change the classification result from "stop sign" to "speed limit" [182]. However, the sticker can be easily detected since it is highly human-perceptible. Turner et al. [181] proposes a clean-label poisoning attack with Generative Adversarial Network (GAN) without changing the label of poisoned data, which improves the stealthiness of the

[8] This work has been published on IEEE TDSC [175].

attack. That is, such clean-label poisoning attacks will not degrade the test accuracy given the normal data, which can be harder to be detected by evaluating the overall performance of the trained model on a clean holdout test dataset.

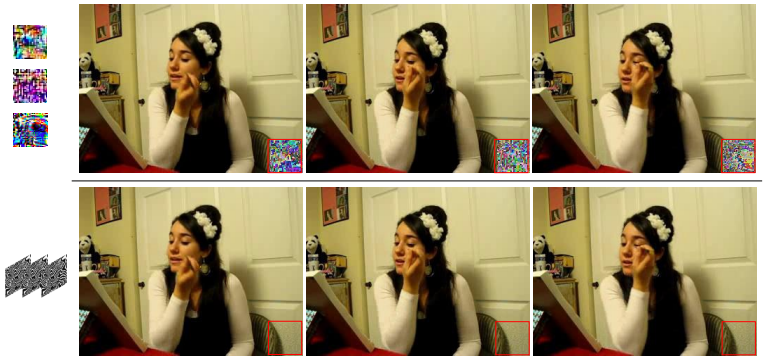


Figure 5.1. A poisoned video example "Apply EyeMakeup" with the poisoning trigger (the squares of pixels leftside): recent work [184] (top) vs. ours (bottom). Our 3D poisoning trigger is more human-imperceptible as nature-like textures compared to [184]'s trigger of highly-deviated pixels.

While most existing data poisoning attacks focus on images [181–183], there are very limited works on DNN-based video models. It is worth noting that Zhao et al. [184] first explores the poisoning attack on the video models by extending a conventional image attack [181] to achieve high performance, which still has the major flaws on poisoning video models as following: (i) the patched frame-by-frame poisoning triggers [184] could jeopardize the temporal consistency in videos such that the poisoning attack might be easily detected, which can degrade the attack *stealthiness* and thus cause attack failure in the testing. We have experimentally shown such poisoned instances with trigger can be accurately detected by the state-of-the-art detection scheme on temporal consistency, AdvIT [114]; (ii) most poisoning attacks rely on feature collision [183, 184] with input-specific data samples by one-to-one mapping (similar to the targeted evasion attack [137]), which could lack *generalization* to the unseen data even if injecting multiple poisoned data [184]; (iii) under the black-box setting, poisoning attacks may not work well by attacking the substitute

Table 5.1. Comparison of our work and existing clean-label poisoning attacks.

	Turner et al. [181]	Zhu et al. [180]	Hidden [183]	Zhao et. al [184]	Ours
Trigger	Adv. Perturbation	-	Rand. Pixels	Deviated Pixels	3D Procedural
Backdoor	✓	✗	✓	✓	✓
Video Dom.	✗	✗	✗	✓	✓
Stealthiness	⋆⋆	⋆	⋆⋆	⋆	⋆⋆⋆
Attack Effec.	⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆⋆

model since the target model can have very different classification boundaries (low *transferability*). These could greatly degrade the attack performance.

To address such limitations, we propose a novel stealthy and effective poisoning attack framework against the video recognition DNN models. Specifically, we first design a 3-dimensional (3D) poisoning trigger with temporal consistency based on a computer graphic primitive for *stealthiness*, which obtains good human-imperceptibility as natural-like textures. Figure 5.1 demonstrates an example of our 3D poisoning trigger compared with the state-of-the-art [184] on poisoning videos. Second, we craft the poisoned videos with the integration of an ensemble attack oracle (as the attack optimization objective), which formulates a convex polytope to cover the targeted videos in feature representation space (to provide more attack generalization and flexibility). Third, our proposed attack can craft more transferable poisoned videos by explicitly optimizing the attack in the intermediate layer feature representation of a video DNN model, which works in the black-box setting. Therefore, our main contributions are summarized as below:

- To our best knowledge, we are the first to reveal the limitations of state-of-the-art video poisoning attack in both stealthiness and attack performance.

- We design novel 3D poisoning triggers with a classic computer graphic primitive to ensure the attack stealthiness, which can be easily generated (by a few parameters) and human-imperceptible (nature-like patterns or textures, see Figure 5.1).

- Based on the 3D poisoning trigger, we propose a general attack framework, which can efficiently craft poisoned videos by formulating an ensemble attack oracle as objective. We further optimize the attack in aspect of attack generalizability and transferability.

- We conduct extensive experiments to validate the attack effectiveness and stealthiness with the benchmark of the previous attack methods. Besides, we have experimentally shown that the proposed attack can bypass various state-of-the-art defense schemes. We also show that our 3D poisoning attack can be readily downgraded to image domain.

5.2 Background

In this section, we first review the related literature of poisoning attack, which also includes the existing defense mechanisms. We also briefly present the taxonomy for DNN-based video recognition models.

5.2.1 Poisoning Attacks. Poisoning attack injects poisoned instances (generated with some specific triggers) into the training dataset [177, 180–184], which can install the particular trigger as backdoor into the DNN. Thus, at the inference phase, the trained DNN model will misclassify the test instances with the presence of such trigger pattern. There are mainly two types of poisoning attacks: (i) poison-label attack can change both the training instances and their corresponding labels; (ii) clean-label attack changes the training instances without changing the labels. Poison-label attack can be mitigated by data filter since the poisoned data (mislabeled data) visually look

different from the clean data but belonging to the same label. For example, Gu et al. [182] first proposes the poisoning attack on the deep learning application, BadNets, which injects patterns (e.g., stickers) into the poisoned data and also changes the corresponding label to the target label (as the poison-label attack). However, the patched triggers (e.g., stickers) can be easily detected via filtering or humans.

To improve the stealthiness of poisoning attack, Turner et al. [181] proposes clean-label attacks without changing the poisoned labels, by utilizing GAN to craft the poisoned instances for feature collision [177]. Saha et al. [183] presents a universal optimization method to generate multiple poisoned instances for one specific source instance, which could achieve relatively high success rates but still lack generalization. Zhu et al. [180] studies the transferability of poisoning attack and generates more transferable poisoned data based on convex polytope. It is limited to attacking the images without the backdoor trigger. Besides, such attack cannot be directly applied to videos since it would be computationally impractical to directly craft the poisoned videos due to the two-fold optimization with additional constraints. Zhao et al. [184] first studies the poisoning attack in the video domain, which aims to jointly craft universal triggers and poisoned videos with adversarial perturbations. Although this method has shown to be effective, it has some major flaws, e.g., temporal inconsistency aroused by the trigger, and low transferability as depicted before.

To address the above limitations, we propose a novel attack scheme for attacking video recognition models. Table 5.1 summarizes the main difference between our proposed attack and the state-of-the-art attacks. Our attack outperforms them on both stealthiness and attack effectiveness while attacking video DNN models (see the design goal in Section 5.4 and experimental results in Section 5.6).

5.2.2 Defenses against Poisoning Attacks. There have been several works which defend against the data poisoning attacks. For instance, Steinhardt et al. [185]

proposed a certified defense scheme by constructing approximate upper bounds on the loss across the poisoning attacks. Tran et al. [186] proposed a spectral signature detection method for detecting poisoned instances in the training dataset. They observed that the poisoned data could be different from the clean data in the latent DNN space, which can be used for removing the poisoned data as outliers from the training data. Liu et al. [187] proposed a fine-pruning method to prune the abnormal units to prevent the poisoning attack. Another approach is the neural network cleanse [188], which checks if the trained model is poisoned via reverse engineering the poisoning triggers with the gradient information. Then, neural cleanse uses an input filter to filter the poisoned data using a simple technique called median absolute deviation. We have experimentally evaluated the resistance of our proposed attack against such defense schemes. The experimental results show that our attack can bypass these defense schemes.

5.2.3 DNN-based Video Recognition. Well-designed DNN models, e.g., C3D [124], I3D [129], TSM [189] and X3D [190] have been widely adopted for efficient and accurate video recognition, such as action classification [191] and anomaly detection [96] in surveillance systems. Starting from the C3D model, the 3D convolutional networks on learning spatio-temporal features have significantly improved the performance of video recognition. Moreover, I3D improves C3D via inflating the 2D convolution filters (in conventional image networks) into the 3D convolution. We will evaluate our attack on such two most representative video DNN models on two large-scale video datasets, UCF101 [192] and HMDB51 [133] for video classification (see details in Section 5.6.1).

5.3 Attack Preliminaries

In this section, we first introduce the threat model, including the attack scenario, the adversary's knowledge/capability. We then formulate 3D poisoning attacks with video models.

5.3.1 Threat Model. We consider the *clean-label* poisoning attack [181, 183, 184] in the video domain. That is, the attacker will generate the poisoned videos, which visually look as the original clean data (thus keeping the clean label to bypass the detection of the data filtering/humans). It should be noted the attacker cannot control the labeling process (different from the poison-label setting). To improve the attack performance, we inject a set of poisoned videos [183, 184] (still a very small portion of training dataset, e.g., 0.5%). Besides, since the generation of the poisoned video is offline, we do not consider the extra computational costs of generating poisoning videos (as pre-attack phase).

For the victim's model, we consider the transfer learning setting [180, 184, 193], i.e., given a pre-trained DNN models as feature extractor (yet kept *frozen*), we can finetune a linear classifier based on to the specific applications/datasets. Such transfer learning-based approach have been shown to be practical and effective considering the relatively small computation costs in various domains. For example, we can utilize a pre-trained I3D model on kinetics-400 dataset to extract the video features and train (fine-tune) a SVM classifier on UCF101 dataset for action classification [129].

Attacker's Knowledge. We consider both white-box and black-box setting. For white-box setting, the attacker only knows the victim's model architecture (white-box) [183]. For the black-box, the attacker will not have access to model's architecture and parameters as the black-box evasion attacks [173]. Then, the attacker can utilize a substitute model to craft poisoned videos to attack the victim's model (via transferability). In both white-box and black-box setting, we assume that the attacker knows the training dataset to train victim's model (thus can generate poisoned data).

Attacker's Capability. As depicted above, we assume that the attacker can successfully inject a small number of well-crafted poisoned data into the victim's training

dataset, which follows the setting of previous works [181,183]. This is reasonable since the victim could obtain the training dataset by crawling from the online resources with web crawler. That is, the attacker only needs to put the generated poisoned data on the internet as public resources, which could be very likely collected by the victim. The attacker cannot control the training process of victim's model.

5.3.2 Attack Formulation. Denote the target video by v_{tar}, and the source video by v_{src}. Given a poisoning trigger $\mathcal{P}_n$ and a binary mask M (the location of patch is 1 while non-patched locations are 0), the attacker can generate a patched source video v'_{src} by patching the poisoning trigger $\mathcal{P}_n$ to the source video v_{src}:

$$v'_{src} = v_{src} \odot (1 - M) + \mathcal{P}_n \odot M \tag{5.1}$$

where $\odot$ denotes the Hadamard multiplication. We assume that the patch location on all the frames in one video are fixed, and can also be changed by modifying the binary mask M. It should be noted that we have verified the location of trigger will not arouse the attack results significantly. We can always adjust the patch location accordingly to obtain more visual imperceptibility (e.g., in the background).

To enable a successful poisoning attack, we will generate a poisoned video v_{poi} which visually looks like the target video v_{tar} such that it can be labeled with the target label. Meanwhile, the poisoned video v_{poi} should be similar to the patched source video v'_{src} in the feature representation of a DNN model (to cause feature collision) [181,183]. Thus, a video instance (belonging to the source class) patched with the trigger can be misclassified into the target class. Formally, the attacker can craft the poisoned video as the following objective function:

$$v_{poi} = \arg\min_{v} ||\mathcal{F}(v) - \mathcal{F}(v'_{src})||_2^2 + \lambda D(v, v_{tar}) \tag{5.2}$$

where $\mathcal{F}(\cdot)$ outputs the video features extracted by a DNN model as feature extractor (e.g., in C3D model [124], the output of the fc7 layer is a 4096-dimensional feature vector). $D(\cdot)$ is a distance function (e.g., ∞-norm distance) to quantify the distance between v_{poi} and v_{tar} (the maximum pixel change). The attack optimization consists of two terms:

1. The first term makes the feature representation of the poisoned video $\mathcal{F}(v)$ close to the patched source video $\mathcal{F}(v'_{src})$.

2. The second term ensures that v_{poi} looks like the target video v_{tar}, which is upper bounded by ϵ.

We utilize the hyperparameter $\lambda > 0$ to weigh the two terms in the optimization. Conventionally, given one specific pair of source and target videos, the attacker can generate the poisoned video by solving Eq. 5.2 using the projected gradient descent (PGD) algorithm [174]. Also, multiple poisoned videos will be crafted to increase the success rate [183, 184].

5.4 Attack Design Goals & Insights

In this section, we will illustrate the major limitations of current poisoning attacks and then briefly introduce our design idea to address such limitations, respectively. Our attack design aims to improve from the following two aspects: 1) stealthiness; 2) attack performance.

G1: Stealthiness. In general, the stealthiness issues of poisoning attack with trigger mainly consist two aspects: 1) the poisoned videos to be injected into the training set (training phase); 2) the patched video with poisoning trigger at the inference phase. Recall that current state-of-the-art poisoning attacks are in clean-label setting, i.e., keeping the original labels of poisoned instances [181, 183]. This can be achieved by

bounding $D(v_{poi}, v_{tar})$ (in Eq. 5.2). However, such clean-label setting can only solve the former stealthiness issue with poisoned videos, which aims to bypass the data filtering or humans. In other words, the latter stealthiness issue of patched videos in the inference phase still exists, especially in video domain [184].

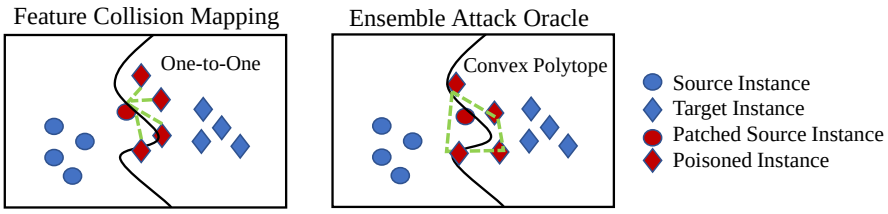


Figure 5.2. Feature Collision Mapping vs. Ensemble Attack Oracle

More specifically, the generated poisoning triggers usually consist of irregular pixels in the RGB space, which could improve attack effectiveness to some extent, however, may also result in temporal inconsistency across different video frames, which can be accurately detected by the state-of-the-art detection scheme, e.g., AdvIT [114] based on video consistency. Besides, as shown in Figure 5.1, the highly-deviated pixels in the triggers could be also discerned by humans. Both of these could directly cause the failure of the poisoning attack during the inference phase.

To address this, we construct a novel 3D poisoning trigger based on a computer graphic primitive *Procedural Noise* [144, 194], which obtains no noticeable directional artifacts and thus potentially fit for stealthiness of the poisoning attack (detailed in Section 5.5.1). We have experimentally shown that our 3D poisoning trigger can bypass the detection scheme, e.g., AdvIT while comparing with the state-of-the-art attacks [183, 184]. Also, we validate that our poisoning attack obtains good human imperceptibility by both quantitative measurement and human survey.

G2: High Attack Performance. As depicted earlier, we need to improve poisoning attack on both *generalization* and *transferability*. On the one hand, conventional

poisoning attacks rely on the feature collision with the specific source/target instances (one-to-one mapping) [183, 184], where minimizing the distance in the feature space could cause source instances to be trapped into the boundary belonging to target label (successful attack). However, such one-to-one mapping for feature collision can be restrictive like the targeted evasion attack [137], which could be still hard to attack unseen data instances even they usually inject multiple poisoned instances (*lacking generalization*). On the other hand, sometimes the attacker may not know the victim's model (in black-box setting), then the feature collision attack may not work on the substitute model since the models can be very different, e.g., feature extractor. That is, for feature collision mapping, the small distance for one pair of source/target instances on one model's feature extractor may change to larger in case of another model (*low transferability*).

Instead, we define an attack primitive, namely, *Ensemble Attack Oracle* (Definition 5) with an ensemble of a set of crafted poisoned videos, which aims to construct some adversarial subspaces as *convex polytope* [195, 196] in feature space to entrap the source video (for a successful attack) [167, 180, 197]. Different from one-to-one mapping in feature collision (some isolated adversarial points), we can formulate a convex polytope with a set of poisoned videos, which can tolerate more generalization errors and also loose the generation of the poisoned videos. Therefore, such adversarial subspaces by ensemble attack oracle can lead to more transferable attack [180, 196]. Figure 5.2 demonstrates the comparison of our ensemble attack oracle with feature collision.

Furthermore, we improve the ensemble attack optimization with two empirical yet effective calibrations. We first leverage Empirical Risk Minimization (ERM) to obtain more generalization. Then for the transferability, we utilize the intermediate layer's features instead of the final output feature of the video model as the feature

representation, since the explicit attacks on the intermediate layers have been shown to be more transferable [138, 198]. We have experimentally validated the attack effectiveness of our attack.

5.5 Attack Framework Design

In this section, we elaborate the attack design for **G1** and **G2** (Section 5.4), respectively. We overview the main steps of the proposed attack framework (shown in Figure 5.3). There are four main steps: 1. the attacker crafts 3D poisoning trigger (for stealthiness); 2. with the optimization of both generalization and transferability, the attacker formulates an ensemble attack oracle to generate a set of poisoned videos (for attack effectiveness); 3. after the attacker injects the poisoned data to the training dataset, the victim will train the DNN model with the poisoned dataset; 4. during the test phase, the attacker can patch the 3D trigger on the test video (to activate the poisoning attack), which can be misclassified to the target label, e.g., "BrushTeeth" to "EyeMakeup".

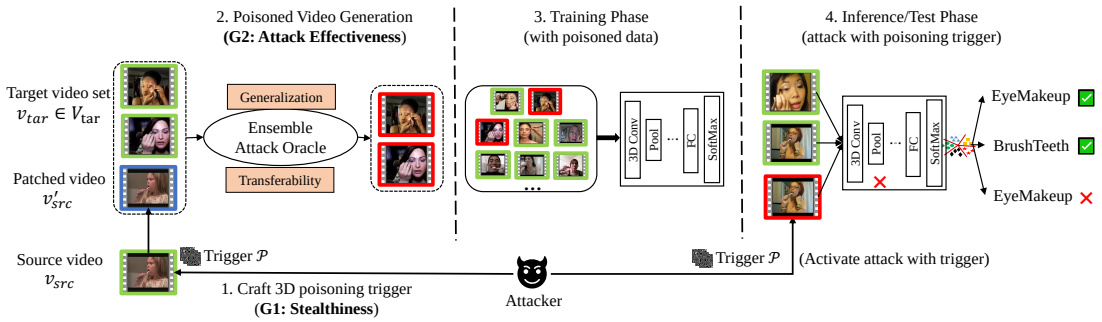


Figure 5.3. Overview of 3D poisoning attack framework

5.5.1 3D Poisoning Trigger Generation. Procedural noise [144, 194] refers to the algorithmically generated visual patterns by some predefined functions, which have been widely used in film production and video games to enrich the visual details, e.g., texture and shading. It is inherently continuous and parameterized to compute [144].

Also, the noise has no noticeable directional artifacts. All these attributes enable the procedural noise (as computer graphic primitive) to be potentially fit for generating human-imperceptible poisoning trigger (*stealthiness*).

To craft the 3D poisoning trigger, we utilize one common type of procedural noise, i.e., Perlin noise [194] due to its ease of generation and popularity. Perlin noise was first proposed by Perlin as an image modeling primitive to produce the natural-like textures. As a lattice gradient noise, the noise value is determined by computing a set of 12 pseudorandom gradient vectors at the midpoints of 12 edges of a lattice cube, and then utilizing a quintic polynomial equation, e.g., $q(t) = 6t^5 - 15t^4 + 10t^3$ to interpolate the pre-defined vectors [194]. It can be computed efficiently with a few parameters. Thus, the Perlin noise can be readily extended to construct the 3D poisoning trigger.

More formally, we denote every pixel of 3D poisoning trigger by its 3D coordinates (x, y, t), where $(x, y), x, y \in [0, d-1]$ are the coordinates in frame t (the trigger is a square of $d \times d$). Denote the Perlin noise value of each pixel (x, y, t) by $s(x, y, t)$. To enrich the visual details (e.g., natural-looking texture for *stealthiness*), we can aggregate a set of octaves (the number of octaves denoted as Λ). Besides, we define two new parameters of wavelength λ_s and λ_t to determine the attribute of octaves along the spatial (location) and temporal (frame), respectively. Then the noise value at 3D coordinates (x, y, t) can be updated as:

$$\mathcal{P}(x, y, t) = \sum_{\ell=0}^{\Lambda} s(x \cdot \frac{2^\ell}{\lambda_s}, y \cdot \frac{2^\ell}{\lambda_s}, t \cdot \frac{2^\ell}{\lambda_t}) \tag{5.3}$$

To further improve the stealthiness by enabling various visual perturbations with different color spaces in the video, we extend Eq. 5.3 with a color mapping function [147]. Then, the noise value of (x, y, t) for the 3D poisoning trigger can be

generated as:

$$\mathcal{P}_n(x, y, t) = K * cmap(\mathcal{P}(x, y, t), \phi) \tag{5.4}$$

where $cmap(b, \phi) = \sin(b \cdot 2\pi\phi)$ is a sine color map function, which bounds the noise with the circular property. K is the upper bound of the 3D trigger (in ℓ_∞-norm).

With such function, our attack can craft the poisoning trigger for patching the video on-the-fly (*video-agnostic*), i.e., we can manipulate the visual texture of the trigger pattern by adjusting the function parameters. For instance, we can first determine the location of the poisoning trigger, e.g., bottom right with trigger size $d = 30$. Since the video classification usually analyzes each video clip with 16 consecutive video frames, we can compute 3D poisoning trigger referring to Eq. 5.4, $t \in [0, 15]$. Also, we can control the style of trigger pattern by adjusting the parameter of the color map function. Once we obtain the poisoning trigger, we can craft the poisoned videos as depicted below. We have experimentally validated that our 3D poisoning trigger ensures good *stealthiness* and human-imperceptibility with both quantitative and human survey study (Section 5.6.4).

5.5.2 Poisoned Video Generation. Following **G2**, we construct an *Ensemble Attack Oracle* [167, 180, 197] to improve attack effectiveness as the following.

Definition 5 (Ensemble Attack Oracle)**.** *Given a patched source video v'_{src} and a set of N poisoned videos to be crafted $V_{poi} = \{v^i_{poi}, i \in [1, N]\}$, then an ensemble attack oracle, denoted as $\mathcal{A}(V_{poi}, v'_{src}, \mathcal{F}(\cdot))$, is to compute the feature representation distance between the linear combination of the poisoned videos set V_{poi} and v'_{src}:*

$$\mathcal{A}(V_{poi}, v'_{src}, \mathcal{F}(\cdot)) = || \sum_{i}^{N} w_i \mathcal{F}(v^i_{poi}) - \mathcal{F}(v'_{src})||_2^2$$

$$s.t. \quad \sum_{i}^{N} w_i = 1, w_i > 0, v^i_{poi} \in V_{poi}$$

Input: Target video set V_{tar}, Source video set V_{src}, Feature Layer function $\mathcal{F}_k(\cdot), k \in [1, L]$, 3D poisoning trigger $\mathcal{P}_n$

Output: N poisoned videos $V_{poi} = \{v^i_{poi}, i \in [1, N]\}$

Initialize N target videos $v^i_{tar} \in V_{tar}$ to be poisoned

$V_{poi} = \{v^i_{poi} \leftarrow v^i_{tar}, i \in [1, N]\}$

Initialize $w_i \leftarrow \frac{1}{N}, i \in [1, N]$

while *not converged* **do**

    Randomly sample $v_{src} \leftarrow^s V_{src}$

    $v'_{src} = v_{src} \odot (1 - M) + \mathcal{P}_n \odot M$

    // Given V_{poi}, update w_i

    for $k \rightarrow 1$ *to* L **do**

        $C \leftarrow [\mathcal{F}_k(v^1_{poi}), \mathcal{F}_k(v^2_{poi}), \cdots, \mathcal{F}_k(v^N_{poi})]$

        $\tau \leftarrow \frac{1}{||C^\top C||_2}$

        update $w_i \leftarrow w_i - \tau C^\top (C w_i - \mathcal{F}_k(v'_{src}))$

    // Given w_i, update V_{poi}

    for $i \rightarrow 1$ *to* N **do**

        Graident step on v^i_{poi}

        Clip v^i_{poi} to be bounded via $||v^i_{poi} - v^i_{tar}||_\infty \leq \epsilon$

return N poisoned videos

Algorithm 12: Poisoned Video Generation

With the ensemble attack oracle, we build a relaxed connection from the poisoned videos to patched source video in the feature space. That is, the convex polytope space constructed by a set of poisoned videos can obtain more generalization than the one-to-one mapping for feature collision [183, 184]. Take Figure 5.2 as an example, for feature collision-based attack, we craft the poisoned video one by one to approach the source videos at the boundary, which could change the classification boundary and thus cause misclassifcation. We can observe that there are four poisoning videos approaching the patched source video on the lefthand side. In general, we can inject more poisoned videos to arouse more change of the boundary (and thus increase the attack success rate). On the righthand side, the four poisoned videos would formalize a convex polytope space with the ensemble attack oracle, where the source videos can be easier to be entrapped for more attack effectiveness.

More formally, we have the following proposition to show attack correctness of such attack oracle:

Proposition 1. *If $\mathcal{A}(V_{poi}, v'_{src}) = 0$ holds, and given $\forall v^i_{poi} \in V_{poi}$ to be labeled with the target class c and successfully injected into the training set, then v'_{src} will be misclassified into the target class c by victim's model (as successful attack).*

Proof. We denote the video linear classifier (after feature extractor $\mathcal{F}(\cdot)$) as $g(\cdot)$. Since all the poisoned videos are labeled to class c, then $\forall v^i_{poi} \in V_{poi}$, we have

$$Pr[g(\mathcal{F}(v^i_{poi})) = c] > Pr[g(\mathcal{F}(v^i_{poi})) = c'] \tag{5.5}$$

where $c' \neq c$ is other labels. Given $\mathcal{A}(V_{poi}, v'_{src}) = 0$, i.e.,

$$\mathcal{F}(v'_{src}) = \sum_{i}^{N} w_i \mathcal{F}(v^i_{poi}) \tag{5.6}$$

we thus get:

$$\begin{aligned} Pr[g(\mathcal{F}(v'_{src})) = c] =& Pr[g(\sum_{i}^{N} w_i \mathcal{F}(v^i_{poi})) = c] \\ = \sum_{i}^{N} w_i Pr[g(\mathcal{F}(v^i_{poi})) = c] >& \sum_{i}^{N} w_i Pr[g(\mathcal{F}(v^i_{poi})) = c'] \\ =& Pr[g(\mathcal{F}(v'_{src})) = c'] \end{aligned} \tag{5.7}$$

□

According to Proposition 1, we can craft a set of poisoned videos to enable source videos to be entrapped in the convex polytope in feature space. Note that such ensemble oracle can also provide more transferable attack due to the larger adversarial subspaces (convex polytope). Then we reformulate the attack optimization function by minimizing $\mathcal{A}$ (enable the source video to be covered by convex polytope):

$$\begin{aligned} \min_{V_{poi}} \quad & \mathcal{A}(V_{poi}, v'_{src}, \mathcal{F}(\cdot)) \\ s.t. \quad & \forall v^i_{poi} \in V_{poi}, D(v^i_{poi}, v_{tar}) \leq \epsilon \end{aligned} \tag{5.8}$$

Moreover, we can further improve our poisoning attack with the following calibration:

(i) Attack Generalization. A simple approach to improve the attack generalization is to attack a set of sampled data instances (aka. universal attack [103]). Thus, to further improve the attack generalization on unseen source videos (not in the training set), we update Eq. 5.8 with the expectation on a pre-selected patched source video set V'_{src} by normalizing the distance (to avoid bias).

$$\min_{V_{poi}} \mathop{\mathbb{E}}_{v'_{src} \sim V'_{src}} \frac{\mathcal{A}(V_{poi}, v'_{src}, \mathcal{F}(\cdot))}{||\mathcal{F}(v'_{src})||_2^2} \quad s.t. \quad \forall i \in N, D(v^i_{poi}, v_{tar}) \leq \epsilon \tag{5.9}$$

(ii) Transferability in Intermediate Layers. As depicted before, we utilize the feature representations of intermediate layers to improve attack transferability. Then, we update Eq. 5.9 with all the feature representations of the intermediate layers across the entire model as below:

$$\min_{V_{poi}} \mathop{\mathbb{E}}_{v'_{src} \sim V'_{src}} \sum_{k=1}^{L} [\frac{\mathcal{A}(V_{poi}, v'_{src}, \mathcal{F}_k)}{||\mathcal{F}_k(v'_{src})||_2^2}] \quad s.t. \quad \forall i \in N, D(v^i_{poi}, v_{tar}) \leq \epsilon \tag{5.10}$$

where L is the total number of layers and $\mathcal{F}_k, k \in [1, L]$ is the k-th layer function to output feature representations.

Since the above objective function (Eq. 5.10) includes one ensemble attack oracle (the linear combination of the poisoned videos' features), we utilize an efficient optimization method to iteratively update both linear coefficients $W = \{w_i\}, i \in [1, N]$ and poisoned videos $V_{poi} = \{v^i_{poi}\}, i \in [1, N]$.

Specifically, we will fix one as the constraint while optimizing the other one. Given the set of poisoned videos V_{poi}, we use forward-backward splitting [199] (which is more efficient than back-propagation with neural model) to compute the optimal coefficients $W = \{w_i\}, i \in [1, N]$; then fixing coefficients W, we update the poisoned videos for one gradient step (due to computational efficiency). Note that we choose Adam optimizer [200] to update the poisoned videos since it converges more reliably. To find the optimal poisoned videos and coefficients, we will repeat the two sub-steps until convergence. Algorithm 12 depicts the details.

5.6 Experiments

In this section, we evaluate our 3D poisoning attack on different video datasets and DNN models with various baselines. We first introduce the experimental setup, including the datasets, models, baselines and the attack methodology. Then, we demonstrate the experimental results in aspects of attack performance and stealthiness (corresponding to our design goal **G1**/**G2**). Besides, we conduct the extensive ablation studies to study the effect of 3D poisoning trigger on the whole attack. We also experimentally shows that the proposed attack can resist defense schemes. Finally, we demonstrate that our 3D poisoning attack can be extended to the image domain (2D).

5.6.1 Experimental Setup. We evaluate the attack on two commonly used real datasets for video recognition:

- The UCF101 [192] dataset has 13,320 video clips in 101 different action categories, e.g., archery, fencing, and punch.
- The HMDB51 [133] dataset contains 6,766 video clips which are categorized into 51 different actions, e.g., fencing, hit, gun shooting, and sword exercises.

For each dataset, we choose 80% of each category as the training dataset, from which we choose the target category to generate poisoned videos. Then, the remaining 20% videos are used for the test dataset. Note that we keep the test videos clean to evaluate model accuracy under different model setting (poisoned or clean). For *stealthiness*, a successful poisoning attack is also expected to maintain the original model accuracy (inference) after training on clean/poisoned training dataset, besides obtaining human-imperceptible perturbations.

Target Models. We mainly evaluate our attack on two state-of-the-art DNNs for video recognition, C3D [124] and I3D [129]. For both C3D and I3D, we first train

the models on kinetic-400 dataset [201] as pre-trained models (working as feature extractor). Then we jointly fine-tune the last layer of models and a SVM classifier on UCF101 and HMDB51 datasets for video classification, respectively. Note that our target models only consider the RGB inputs (modifying the RGB values at the pixel level).

Baselines. Recall that there are very few works on the poisoning attack in the video domain, we utilize the most recent clean-label video attack [184] (denoted as "Baseline1"). We also extend a recent state-of-the-art image poisoning attack [183] to the video domain as the baseline (denoted as "Baseline2"). In addition, we downgrade our proposed 3D poisoning attack to 2D image and compare with Baseline2 [183]. The experimental results (Section 5.6.6) show that our attack can also effectively attack in image domain.

Attack Methodology. For both UCF101 [192] and HMDB51 dataset [133], we split the dataset into 80% training set and 20% test dataset (remain intact to evaluate the model accuracy). Take the first group of experiments (attack effectiveness) as an example, we randomly choose 50 pairs of source and target categories from the UCF101 dataset. For every source/target pair, we randomly select 20% videos of source category as the source video set V_{src}, to which we aim to attack, i.e., the source video patched with the 3D trigger during the test phase will be misclassified into the target class. We also randomly select 20% (as poisoning percentage, $\sim 0.2\%$ out of the entire training set) videos from target category as target video set V_{tar}. Then, we generate the poisoned videos following Algorithm 1 (unless explicitly specified, the parameters will keep the same). The poisoning trigger size is 30×30 out of the 320×240 video frame. we set the upper bound ϵ is 8. We use Adam [200] with a relatively large learning rate of 0.05, and perform at most 3000 iterations on crafting poisoned videos for each experiment.

5.6.2 Attack Performance. To fully evaluate attack performance of the poisoning attack, our evaluation include the following three aspects:

1. The impact on model performance with clean data.
2. The effectiveness (attack success rate) with various models/datasets/attack parameters.
3. The comparison with baselines on attack success rate/transferability.

1) Impact on model performance. As depicted above, the poisoning attack should not impact the normal performance of victim's model (with poisoned training data) too much to keep stealthy. We evaluate the accuracy of the retrained model training with poisoned video dataset and normal training model with the clean dataset. we report both accuracy on the clean test dataset (excluded from the training videos), with UCF101 and HMDB51 dataset, respectively.

Table 5.2. Test accuracy of the clean and poisoned models.

Model / Dataset	C3D		I3D	
	Clean	Poisoned	Clean	Poisoned
UCF101	82.7%	81.5%	87.5%	86.3%
HMDB51	52.3%	51.1%	63.7%	62.4%

Table 5.2 summarizes the results for both C3D and I3D. We can observe that the poisoned video can maintain almost same accuracy compared with the original model, which shows that our 3D poisoning attack will not arouse too much change (slight drop) on the model prediction (only fool the model while presenting with poisoning trigger).

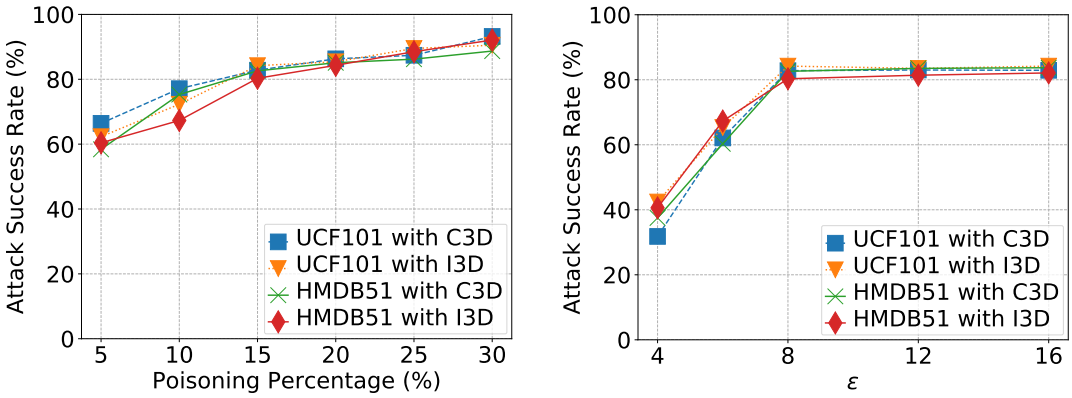
2) Attack effectiveness. We first evaluate the attack performance for specific pairs of source and target video categories with the fixed poisoning trigger size and poisoning

Table 5.3. Attack performance against the C3D and I3D models. $\epsilon = 8$ and poisoning percentage: 20%.

Src./ Tar.	Brush/ EyeMake	Biking/ HairCut	ClifDive/ Rowing	Fencing/ JumpJack	Hammer/ Archery	LongJump/ Skiing	Knitting/ Punch	Punch/ Typing	Skiing/ Tachi
C3D	90.7%	86.2%	89.2%	89.9%	96.1%	93.4%	87.0%	94.7%	96.2%
I3D	89.1%	88.3%	93.1%	92.5%	88.3%	86.7%	93.1%	88.5%	92.0%

Table 5.4. Attack performance of our attack vs. the baseline attacks [184] and [183], denoted as "Baseline1" and "Baseline2". Target category: "Apply EyeMakeup". $\epsilon = 8$ and poisoning percentage: 30%.

Model/ Dataset	Method	Brush Teeth	Biking	CleanAnd Jerk	Frisbee Catch	Horse Race	Long Jump	Playing Dhol	Punch	Skiing	Taichi
I3D/ UCF101	Baseline1	71.0%	76.2%	87.5%	88.0%	70.2%	74.9%	91.3%	82.5%	81.7%	86.0%
	Baseline2	80.5%	83.0%	86.2%	85.0%	76.2%	78.5%	84.2%	86.0%	87.4%	88.0%
	Ours	95.0%	90.4%	93.6%	91.7%	89.5%	94.0%	92.3%	96.5%	94.4%	93.8%



(a) (b)

Figure 5.4. Attack success rate vs. poisoning percentage (a) and perturbation bound ϵ (b) on the UCF101 and HMDB51.

percentage. The poisoning trigger size is 20×20 out of the 320×240 video frame. We set trigger's magnitude $K = 10$. We randomly select 20% videos from the source category as the source video set V_{src}. We also randomly select 20% (as poisoning

percentage, $\sim 0.2\%$ out of the entire training set) videos from target category as target video set V_{tar}. The upper bound $\epsilon = 8$. Table 5.3 summarizes the results of our 3D poisoning attack applied to 9 randomly selected pairs of source/target video categories in the UCF101 dataset against the C3D/I3D models. We can observe that our attack achieves high success rates on both C3D and I3D models, even with a small poisoning percentage, which shows both effectiveness and efficiency of our attack (note that small poisoning percentage reflects high efficiency).

We also evaluate the attack performance with the varying poisoning percentage and perturbation bound. As shown in Figure 5.4(a), the attack success rate also increases as the poisoning percentage grows. Our poisoning attack still achieves high success rates (>80%) even though the poisoning percentage is only 15%. This is consistent with the former results. From Figure 5.4(b), we observe that the attack rate at first increases and then does not change as perturbation bound increases from 8 to 16. This is because the craft poisoned video will be easier with a high perturbation bound. Note the perturbations with poisoned video are still small (8 out of 255).

3) Comparison with Baselines. Table 5.4 demonstrates the results of our 3D poisoning attack applied to the UCF101 dataset (against the I3D model) comparing with the two baselines [183, 184], denoted as "Baseline1" and "Baseline2". We set "Apply EyeMakeup" as the target category, and the source categories (e.g., "biking") as [184]. The trigger size is 20×20 and the poisoning percentage is 30%.

As shown in Table 5.4, our attack achieves high success rates (>89%). For example, our attack can achieve 95.0% success rate on the source category of "Brush Teeth" and 90.4% on the "Biking" while Baseline1 only achieves 71.0%(<95.0%) and 76.2%(<90.4%) on such two categories, respectively (the third and forth columns). Moreover, comparing the remaining results, we can observe that our 3D poisoning attack can perform much better than both baselines. Such results are reasonable since

our attack obtains good attack generalization for crafting poisoned videos.

Attack Transferability. For poisoning attack, we refer to the tranferability of poisoned data to be applied to another model. Then we evaluate the transferability of our attack compared with baselines (the same notations as above). Specifically, we choose one model (e.g., C3D) as substitute to generate poisoned videos, and we evaluate attack success rate on another model (e.g., I3D) trained with the poisoned videos, and vice versa. Figure 5.5 summarizes the overall results. The results show that our poisoning attack obtains high transferability across different models while the baselines lack such transferability (no more than 12% success rate). For example, our attack can achieve 50.5% success rate while Baseline1 only 8.6% on UCF101 dataset. Such results are reasonable. Considering the conventional poisoning attacks focus on the feature collision [183, 184] with fixed feature extractor function, the poisoning attack only obtain less transferability (the feature extractor of different models can be different, i.e., one successful crafted poisoned video for one feature extractor may not work for another). On the contrary, our attack can craft the poisoned videos (ensured by Eq. 5.10) which obtain good generalization and transferability. This also conforms with the previous results.

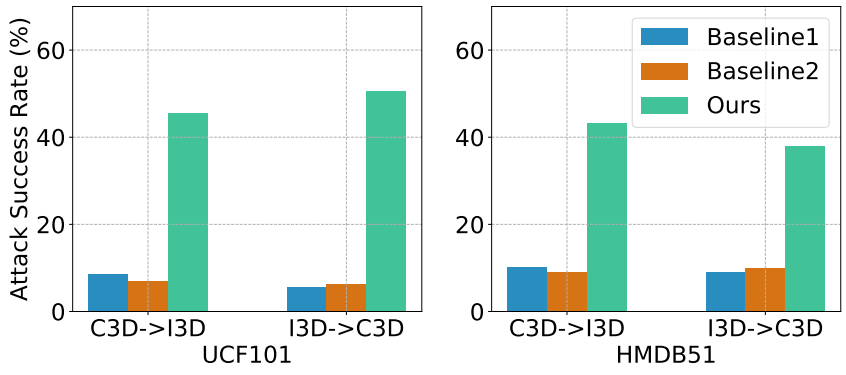


Figure 5.5. Attack transferability of our attack vs. baselines.

Computational Overheads. Table 5.5 demonstrates the average running time for crafting 10 poisoned videos for 8 randomly selected pairs of source/target category in the UCF101 dataset. From the table, we can see that the average running time for one videos is around 1 minute at most. Considering our poisoning attack only injects very small number of poisoned videos, the computation overhead for crafting poisoned video is tolerable.

Table 5.5. Average runtime for crafting poisoned videos (sec).

Biking/ HairCut	CliffDiving/ Rowin	Fencing/ JumpingJack	Hammering/ Archery	LongJump/ SKiing	Knitting/ Punch	Punch/ Typing	Skiing/ Tachi
35	39	28	62	47	35	40	32

5.6.3 Attack Stealthiness. Recall that we reveal stealthiness issue of current poisoning attacks at inference phase can be caused by the highly-deviated poisoning trigger. That is, the videos patched with poisoned trigger can be easily identified by human (visual impact) or detection schemes. Thus we evaluate the stealthiness of our attack on the following aspects. For visual impact, we conduct both quantitative and human study. We adopt the state-of-the-art detection scheme for detecting the poisoning trigger.

1. Quantitative perceptual metric, i.e., SSIM [202].
2. Human-imperceptibility survey study.
3. Video poisoning detection, i.e., AdvIT [114].

1) SSIM. Structural Similarity (SSIM) is a perceptual metric to quantify the visibility of errors between a distorted image and the original image based on the degradation of structural information [202]. The range of SSIM is $(0, 1]$. A higher SSIM value indicates a better quality of the distorted image. Then we can utilize SSIM to quantify

the visual impact of poisoning trigger. We choose the average SSIM for all the frames of one video as the SSIM of the video. Recall that our poisoning trigger is upper bounded by K (Equation 4.3). We set K as $\{5, 8, 10, 12\}$. Next, we randomly choose 100 poisoned video with one 3D trigger from each category, and average the SSIM as the final result.

Table 5.6. Average SSIM of 100 poisoned videos with varying K.

K	5	8	10	12
SSIM	0.997	0.994	0.986	0.984

In Table 5.6, the SSIM of the videos is very close to 1, which shows that the 3D poisoning trigger rarely affects the visual information. Thus, the attacker can simply adjust the parameters of the poisoning trigger function (e.g., K) with no significant visual changes in the poisoning attack. Note we also conduct extensive ablation study of 3D poisoning trigger in Section 5.6.4.

2) Human study. We conducted a human survey study to evaluate whether our poisoning attack could cause visual effect to humans (with the IRB exempt approval).

For the setup of study, we first generate the videos (including original videos, patched videos with trigger) by our attack. Specifically, we randomly pick 500 videos from the UCF101 and HMDB51 datasets. To avoid bias on the distribution of data samples, we first randomly choose 250 videos to generate 250 pairs of videos (the poisoned videos and original clean videos), and the remaining for 250 pairs of clean videos and their duplicates. Then we distribute an online survey to 50 anonymous students (not record any personal information, e.g., major, age or gender), which ask each participant to annotate 10 pairs of videos as ("visual difference" or "no visual difference"). Finally, we received 490 valid annotations of video pairs (49 students have submitted their results), including 244 poisoned pairs. Figure 5.6 demonstrates

the results (left-side). We found that 97.5% (238) out of such 244 poisoned videos are annotated as "no visual difference", while only 2.5% are identified (as "visual difference"). There also exist 8 annotations identified as "visual difference" in the remaining 246 pairs of original videos and their duplicates.

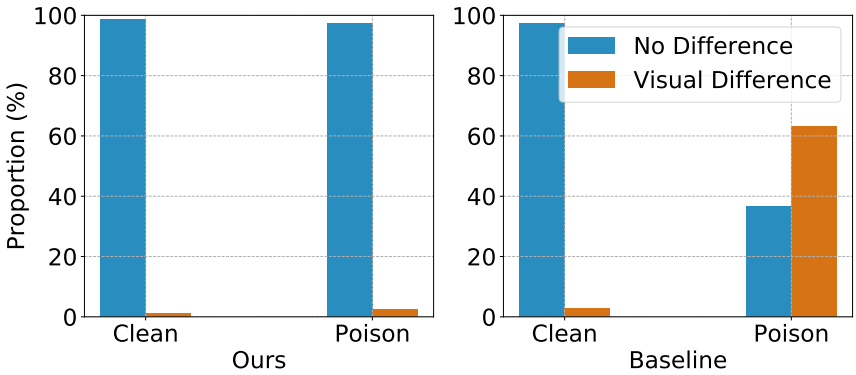


Figure 5.6. Results of human survey on our attack vs. baseline [184].

We also repeat the same group of study for baseline attack [184] but selecting *different* videos from the dataset, which aims to avoid the connection with the previous study for our attack. That is, the previous annotation of our attack will enable the participants to have prior knowledge and then make biased annotation on the same pair of videos for baseline attack (vice versa). From Figure 5.6, we observe that 63.8% (157 out of 246 valid poisoned videos) are identified by the same group of participants. All the above results have indicated that our attack achieve high human-imperceptibility (significantly better than the baseline [184]).

Figure 5.10 gives two example pairs of source and target videos in categories "PlayingDhol" and "Apply EyeMakeup". Due to strictly bound perturbations, the poisoned target video is visually similar to the target video. The patched source video with 3D poisoning trigger is also very similar to the clean source video.

3) Poisoning detection. Recall that we observe the poisoning triggers can directly

cause the stealthiness issue by temporal video frame. We adopt a state-of-the-art detection scheme AdvIT [114] to detect the video patched with poisoning trigger (thus validate the limitation of previous attack [184]). AdvIT is originally designed to identify adversarial perturbation in the videos. Based on the assumption that perturbations can destroy the video frame consistency, AdvIT can find the temporal inconsistency among video frames by the optical flow information.

We identify the poisoning triggers of highly-deviated pixels [184] could be potentially destroy temporal inconsistency of video frames as adversarial perturbations. Then we adopt AdvIT to detect the poisoning trigger in the videos. Specifically, AdvIT first utilizes a DNN-based optical flow estimator, i.e., FlowNet [203], which can compute the optical flow information of suspicious video (usually a few video frames since the poisoning trigger is patched on the whole video). Then such optical flow information can be used to reconstruct some pseudo frames. We can output a inconsistency score between the suspicious video frames and pseudo video frames. Since the perturbations/triggers usually consists of deviated pixels, the optical flow information along with video frame will be destroyed. That is, the higher inconsistency score, the higher possibility poisoning trigger's existence. Note that the trigger is usually fixed, e.g., bottom right. We can always separate the video with different regions for more precise detection.

Table 5.7. SSIM and Detection AUC of AdvIT.

Dataset / Trigger	UCF101		HMDB51	
	SSIM	AUC	SSIM	AUC
Baseline1	0.804	98.5%	0.822	99.3%
Baseline2	0.841	99.2%	0.865	98.4%
Ours	0.956	61.3%	0.973	58.6%

In the experiments, we choose other two types of poisoning trigger from the

baselines: i) randomly generated static trigger [183]; ii) universal adversarial trigger from video poisoning attack [184] for comparison. We fix the trigger size as 20×20 and patch location is bottom right (as fixed in [184]). we randomly select 400 clean videos from the UCF101 and HMDB51 (200 each dataset), and apply both our 3D poisoning trigger and two baselines' trigger to generate patched videos. We set the upper bound of poisoning trigger to be $K = 8$. We report the Area Under Curve (AUC) values of AdvIT for detecting trigger and the average SSIM values of the corresponding videos for detection in Table 5.7 .

From the table, we can observe that the SSIM of our poisoned videos is close to 1, which shows that our 3D poisoning trigger rarely affects the visual information. Besides, the AUC values of ours are close to random guess (e.g., 61.3% for UCF101 dataset) while all other two baselines can be almost fully detected by AdvIT (the AUC values are close to 1). This is reasonable since the temporal consistency could be destroyed with the baseline's (highly deviated pixels).

5.6.4 Understandings of 3D Poisoning Trigger. We also perform ablation studies with 3D poisoning trigger in aspects of the stealthiness and attack performance with various trigger size, upper bound and patched location. Specifically, for every experiment, we will vary one parameter independently while fixing others and report the corresponding results. Figure 5.7 first visualizes the 3D poisoning trigger patched on 16 consecutive frames of the video.

Table 5.8 shows the attack performance of various trigger parameters on UCF101. We observe that both upper bound K and trigger location do not influence our attack performance much. Then we can adjust the trigger location to match with the background/objects (to improve stealthiness). Moreover, we see that the increase of trigger size can slightly improve the attack performance (as a larger trigger patch can help construct adversarial subspaces and attack easier to some extent), which also

degrades stealthiness.

Table 5.8. Attack performance *vs.* varying trigger parameters.

Trigger Size			K		
10	20	30	8	10	12
82.3%	82.7%	83.0%	82.7%	82.7%	82.6%

To evaluate the effect of patched trigger location on the attack performance, we choose 5 different locations, i.e., top/bottom + left/right and center on the video frames. We perform the same attack evaluation as previous experiments and report attack success rate. Trigger size is 20. Poisoning percentage is 20% and upper bound is 8. Table 5.9 shows that trigger location cannot impact the attack performance too much. This is reasonable since the poisoning trigger will not directly be used for crafting poisoned videos to cause feature collision (as a backdoor in the victim's model). Besides the temporal consistency, we can further utilize the natural-like texture of our proposed trigger to increase the stealthiness, i.e., match the trigger with the background or the objects. The SSIM values of our poisoned videos also validate this point.

Table 5.9. Attack rate (AR) *vs.* varying trigger locations.

Location	Top Left	Top Right	Bottom Left	Bottom Right	Center
AR	82.6%	83.1%	83.0%	82.7%	82.9%

5.6.5 Resistance of Attack against Defenses. Besides adopting video detection scheme (Section 5.6.3), we also conduct extensive experiments to evaluate the resistance of our attack by adopting several state-of-the-art defense schemes: i) Fine-Pruning [187]; ii) Neural Cleanse [188]; iii) Spectral Signature [186], respectively. Additionally, we also design an adaptive defense scheme to fully evaluate the proposed poisoning attack.

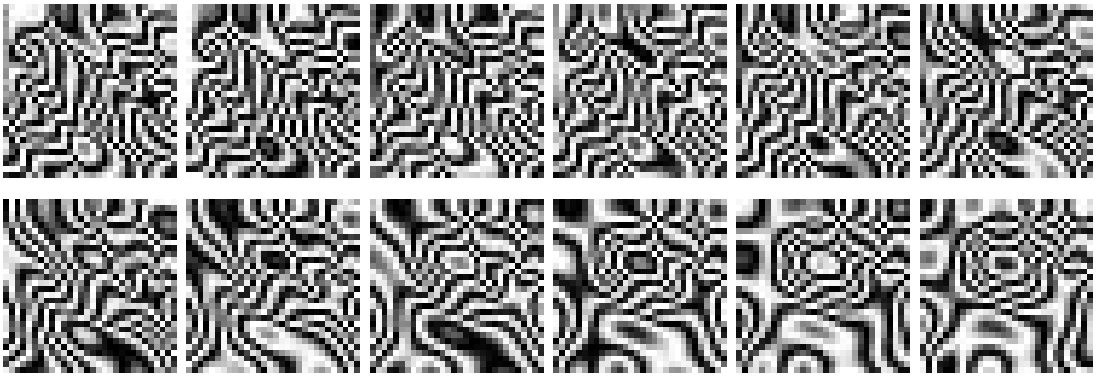


Figure 5.7. One example visualization of 3D poisoning trigger (the first 12 consecutive frames). Trigger size: 30×30.

Fine Pruning. We evaluate the resistance of all three attacks against the state-of-the-art Fine-Pruning [187]. We set the poisoning percentage to be 30%. The trigger size is 20 and upper bound is 8. We train C3D with the poisoned UCF101 dataset compared with other two baselines. For pruning, we prune the last convolutional layer of C3D model (i.e., Conv5b 512) to evaluate the corresponding accuracy and attack success rate. As shown in Figure 5.8(a), the attack success rates of both baselines drop drastically when 30% neuron are removed, e.g., Baseline1 from 84.2% to 30.4%. While our poisoning attack can still maintain 80% attack rate, which shows that our attack is more resistant to the neural pruning.

Neural Cleanse. Neural Cleanse [188] can detect whether a trained model is poisoned or not, where it assumes the training instance would require minor modifications by the attacker. The tested model by Neural Cleanse will output an anomaly index (score) and a score higher than 2 indicates the poisoned model with backdoor trigger. We set the poisoning percentage to be 30%. The trigger size is 20 and upper bound is 8. We train both C3D and I3D model with the UCF101 dataset, respectively. Then we apply Neural Cleanse to detect both C3D and I3D model trained with the UCF101 dataset (by our attack). From Figure 5.8(b), we can observe that Neural Cleanse fails to detect the poisoned model for both cases, i.e., anomaly index <2 (>2 indicates

detected poisoned model) [188].

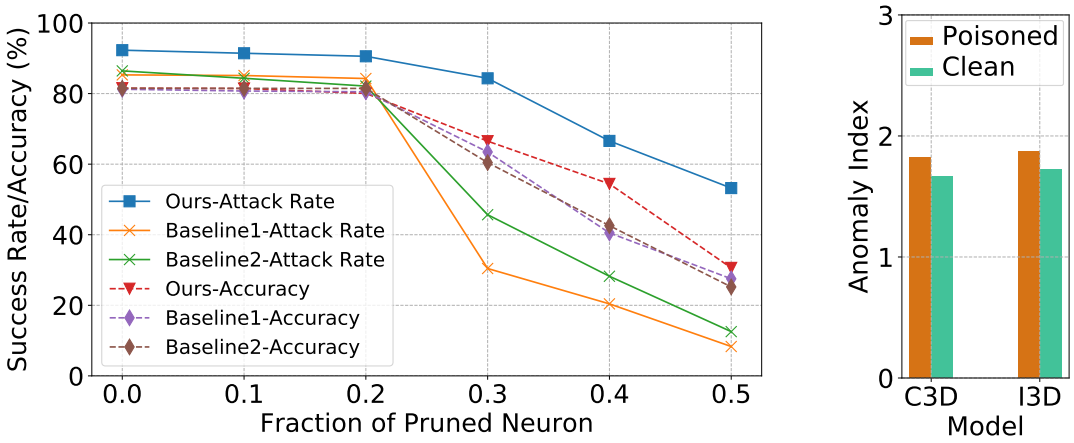


Figure 5.8. Attack results against defenses. Left: Fine-pruning [187]. Right: Neural Cleanse [188]

Spectral Signature. We also apply one state-of-the-art detection scheme Spectral Signature [186], to detect the poisoned data in the training dataset, of which the intuition is that the poisoned data can be outliers in some latent spaces (thus can be removed). It uses statistical methods, e.g., SVD to detect the posioned samples as outliers. For experimental setup, we evaluate this scheme with the C3D model on the UCF101 and HMDB51 dataset, respectively. We set the poisoning percentage of the training dataset as 30% as a higher ratio. The trigger size is 20 and upper bound is 8. Then, we apply the detection on the 1000 videos in UCF101 dataset (consisting of 800 clean target videos and 200 generated poisoned videos) and 500 videos in HMDB51 dataset (400 clean target videos and 100 poisoned videos). Figure 5.9 demonstrates the detection results. From the figure (lefthand), taking UCF101 as an example, we observe that the detection method can only identify a small percentage (∼11%) of poisoned videos while also reporting false positive rate (∼9%) from the clean videos. The result of HMDB51 shows similar results. The above results indicate that such detection cannot mitigate our attack. Also, the attack success rate only downgrades

about 4% even we remove the poisoned data as experiments and retrain the model. Note that the two baselines also report the similar results for this detection.

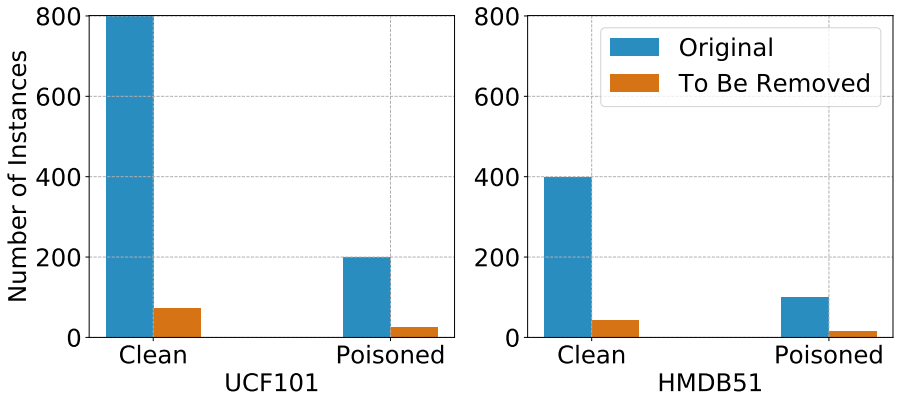
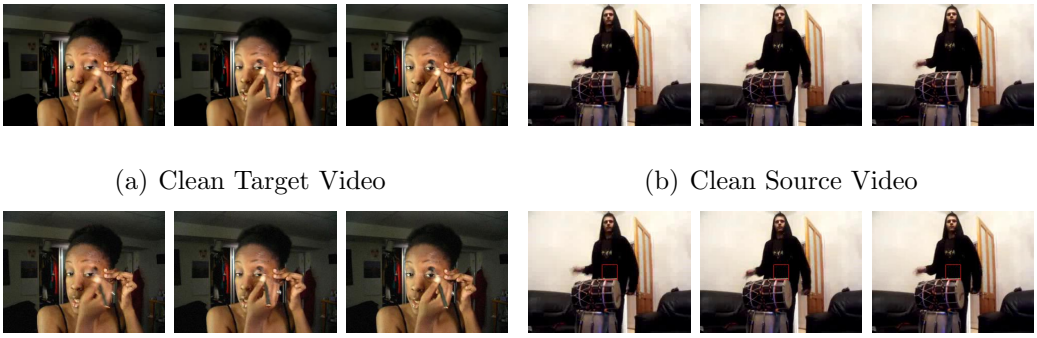


Figure 5.9. Detection results of Spectral Detection [186].

Adaptive Defense. To fully evaluate our proposed attack, we also adopt current defense method as adaptive defense to tailor with the attack properties [139]. That is, we facilitate the defender with the knowledge for the 3D poisoning attack, e.g., the computer graphic primitive procedural noise is utilized for constructing poisoning triggers for our 3D poisoning attack. For the defense method, we improve Spectral Signature [186] by applying procedural noise-based poisoning triggers to its learning scheme.

Specifically, the defender will generate poisoned video samples with the procedural noise as a part of the training set for the detector. Thus it would potentially increase the detection performance considering the detector could achieve more generalization with the newly added poisoned videos. Since the defender does not necessarily know the poisoning trigger parameter, we assume that the poisoning triggers are randomly generated and patched on the videos. We follow the same setting as the detection experiments above. We report the final detection results in Table 5.10 for both UCF101 (first row) and HMDB51 (second row) dataset, respectively. From the



(a) Clean Target Video (b) Clean Source Video

(c) Poisoned Target Video (d) Patched Source Video

Figure 5.10. Visualization of selected frames of clean target video (a), clean source (b), poisoned target video (c), and patched source video (d) of one specific pair, i.e., "Apply EyeMakeup" and "PlayingDhol". With strictly bounded perturbation, the poisoned target video (c) is visually no difference compared with the clean target video (a), but close (in feature space) to the patched source video (d) with 3D poisoning trigger, where the trigger (in "red frame") is human-imperceptible.

table, we can observe that adaptive defense achieves a higher detection rate and also a lower false positive rate on both datasets, e.g., 27% > 11% and 5.6%>9%. Such results show that the adaptive defense can defend our attack to some extent. However, our proposed attack can also change its attack strategy, such as adjusting trigger generation function with another procedural noise to bypass the detection, which would require more robust and adaptive defense schemes.

Table 5.10. Detection results of adaptive defense on UCF101 and HMDB51

Clean	Poisoned	Removed Clean	Removed Poisoned
800	200	47/5.6%	53 /27%
400	100	15/3.8%	32 /32%

5.6.6 Application on Image Poisoning Attack. Considering that the image can be viewed as a one-frame video, we can simply extend our 3D poisoning attack to images, i.e., downgrading the 3D poisoning generation to the 2-dimension by setting the time dimension to be 1. Then, we implement our poisoning attack on the CIFAR10 dataset [204] by benchmarking with the recent image poisoning work,

"Baseline2" [183]. Under same experimental setting of the baseline attack (see details in [183]), we choose 10 randomly selected pairs of image categories, such as bird-dog, dog-plane and cat-truck (specific information of image categories pairs refers to Table 7 in [183]). The model is a simplified AlexNet which has four convolutional layers (64, 192, 384, and 256) kernels and two fully connected layers (512 and 10) neurons. The size of poisoning trigger 8×8 and the bound of trigger is 16. The size of images evaluation dataset for each category is 1000. We average all the success rates of 10 randomly selected pairs via our attack compared with the baseline attack.

We present the attack results for four representative pairs of image categories in Table 5.11. We observe that our downgraded 3D poisoning attack can still achieve high success rate on the image compared with the baseline. Such results have shown the flexibility and effectiveness of our attack. Again, our poisoning attack can also ensure the stealthiness of poisoning trigger in the inference phase, whereas the baseline only focuses on hiding the poisoning trigger prior to training and still reveals the trigger pattern for testing.

Table 5.11. Comparison of attack results on the CIFAR10. Baseline attack [183].

Source/Target	bird/dog	dog/ship	frog/plane	cat/truck
Baseline	94.3%	87.6%	90.1%	93.0%
Ours	92.7%	90.4%	90.8%	94.4%

5.7 Discussion

We will discuss the potential mitigation of our 3D poisoning attacks and advanced attacks to motivate more robust defense schemes as the following.

Potential Mitigation. Considering the poisoning attack is a data-intensive attack, The potential defense schemes can be studied in the following aspects: 1) the detection of input videos with poisoning trigger during the inference phase, e.g., utilizing the

property of trigger in the video domain; 2) the data filtering/detection of the poisoned training data (training phase), e.g., using the adversarial outliers of poisoned training data; 3) the certified robustness [122, 167, 205] against poisoned training data. The first two aspects aim to detect or mitigate the poisoning attack empirically with state-of-the-art schemes, while the last one is theoretical defense scheme against norm-bounded adversarial attack. Considering that the poisoning attack could depend on some intrinsic attributes, e.g., attack by the feature collision of feature representations, we may extend such certified robust scheme to defend against poisoning attacks.

Detection. Recall that we have designed a detection scheme adopting from AdvIT [114], it could effectively detect the poisoned instances with poisoning trigger of the baselines. Thus, to mitigate the risks of the proposed attack, we may utilize the knowledge of the procedural noise as the main defense primitive. That is, we could utilize the procedural noise as the defender's knowledge to revise/adopt the current poisoning or adversarial detection schemes adaptively, such as Spectral Signature [186], AdvIT [114]. We have shown an adaptive defense method based on spectral signature, which can defend against our attack to some extent. Alternatively, we can revise the AdvIT to train a detector by adding procedural noise to increase the detection accuracy, e.g., to enable the detector to memorize the change of optical flow aroused by the procedural-based trigger. We can also leverage ensemble-based [206, 207] method to improve the performance of the detector. For instance, we can choose multiple video models as base models to train multiple detectors and then get an average score for detecting the poisoned videos. Finally, we could leverage a reference database to classify the poisoned videos by k-NN. However, it could bring extra both storage and computational overheads.

Certified scheme. Certified robustness [122, 167] schemes have been shown to defend against adversarial attacks with additive ℓ_p bounded perturbations theoretically. More

specifically, the certified scheme, e.g., random smoothing [122], can provide consistent predictions with guarantee for some norm-bounded input sets around one data instance, i.e., ℓ_p ball. That is, such ℓ_p ball could provide a "safe" space to resist such adversarial perturbed inputs. Similarly, we can enforce the trained classifier to form a "anti"-convex polytope [167] against such convex polytope-based poisoning attacks. That is, we can utilize the randomized smoothing method provided by certified schemes to trap the poisoned training data with a larger convex polytope. Thus after training, the model can still classify the poisoned video into the correct label instead of wrong label with high confidence. However, it should be noted the curse of high-dimensionality [165] still exists for certified robustness scheme, e.g., randomized smoothing, especially in video domain. We will work in this direction.

Advanced Attacks. We propose a *general* attack framework based on 3D poisoning trigger, which can improve the stealthiness of poisoning attack in the video domain (new modeling of poisoning trigger). Besides, our framework also integrates new attack ensemble to improve attack performance in both generalization and transferability. This can bring more flexibility. For example, there will be some new or unknown video models (black-box), we can attack such models with the transferability. Also, our 3D trigger attack framework can also readily integrate other new attack optimizations or powerful attacks from adversarial attack domain in the future to obtain more attack performance. Note our poisoning attack can achieve good human-imperceptibility (according to the quantitative or human servery results), we can also further integrate our attack into the physical-world attacks [182, 208] based on the natural-like texture or style of poisoning trigger. For example, we can utilize visual light technology, such as smart LED [209], which could help to realize 3D poisoning trigger by programmable building blocks. This can pose a practical threat in the physical world.

CHAPTER 6
PRIVACY EVALUATION OF LANGUAGE MODELS

6.1 Introduction

With the development of deep learning technologies, a large number of applications in various domains (e.g., image classification and NLP) have been greatly promoted with significantly improved performance. However, this also arouses serious privacy concerns since a large portion of the training data are usually collected from individuals. For instance, the diagnosis systems in hospitals or healthcare institutions will be trained on the patients' private data, such as medical history [210], and radiology medical images [211]. In addition, it has been reported that the input keyboard prediction model can be trained with the users' data on mobile devices [212], and the assisted composing function for emails/texts can be trained with users' personal messages [213].

The privacy-enhancing technologies (PETs) [214–217] have been widely studied to ensure the data privacy in the machine learning, which mainly include two foundations of theory as following. First, the cryptographic protocols [216, 218] can help to securely train the model with the private data (in encrypted format), and the privacy of data depends on the hard mathematical problems [219]. Although the cryptographic protocol-based schemes provide good data privacy, these also arouse high computational overheads due to the computation on encrypted data and other complicated building blocks.

Second, differential privacy (DP) [220, 221] provides a lightweight way to protect the data against the adversaries with arbitrary information during the training, which can obtain quantifiable privacy guarantees. For example, the widely used DP-SGD [222, 223] ensures the privacy of training data sample by clipping the gradients

and adding DP noise (e.g., Gaussian mechanism) with the model updates. The introduction of DP noise enables the limited effect of one individual data on the trained model (and thus achieving the privacy guarantee). Additionally, another category of work is to add DP noise into the dataset following the method of DP synthetic data release and then train a model on such private data [224, 225]. Yet, the differential privacy-based learning schemes could cause great accuracy loss.

Despite the above demerits, both types of methods can ensure provable privacy guarantees for the training data. This also raises the question: *are there any private learning schemes which can preserve both accuracy and efficiency?* To this end, there are several techniques [226, 227] which privately train the model via the so-called instance encoding scheme, by encoding the local data into a somewhat "encrypted" (encoded) data with a *mixup* scheme [228], and directly training the model on the encoded data. Data privacy is claimed to be well preserved through the encoding method while only causing minor accuracy loss with the merit of the *mixup* scheme.

In aspect of the privacy issues with instance encoding, we conduct comprehensive and empirical studies for such method in the language domain. The main contributions are highlighted below.

- We first show that the instance encoding cannot provide sufficient privacy protection as the conventional cryptographic techniques against well-designed attacks. We design a reconstruction attack to recover the original data from the privately encoded data to demonstrate the limitation of instance encoding (Section 6.4). [9]

- We then improve the TextHide with differential privacy and prove the improved scheme ensures theoretical privacy guarantee under the differential privacy

[9]This work is published in EMNLP [229].

framework (Section 6.5). [10]

6.2 TextHide

The TextHide [226] aims to protect the private text data under the federated learning setting. First, the input text is pre-processed with a BERT transformer encoder to output the corresponding text representation. Then, for "encryption", TextHide will apply the instance encoding to mix up the original text representation with some randomly selected text (representations), which will be fed into the training model of various downstream language understanding tasks, e.g., classification, and question answering. Formally, given the input text x_i with the label y_i, we denote the text representation as $e_i = \phi(x_i)$, where $\phi(\cdot)$ is a pre-tuned BERT model. The private instance encoded data $\widetilde{e}_i$ can be generated as below:

$$\widetilde{e}_i = \sigma \circ \sum_{j=1}^{K} \lambda_j e_j \tag{6.1}$$

where λ_j is chosen uniformly at random such that $\sum_j^K \lambda_j = 1$, the sign-flipping mask $\sigma \in \{-1, 1\}^d$ is also chosen uniformly at random, and d denotes the dimension of the encoding vector. $\circ$ represents the Hardamard (element-wise) multiplication, and K is the number of combined mix encoding data (as the security parameter). Therefore, the label (one-hot vector) $\widetilde{y}_i$ of the $\widetilde{e}_i$ is updated as: $\widetilde{y}_i = \sum_{j=1}^K \lambda_j y_j$, which is the element-wise addition across y_j. Then, for the training with one data batch $\mathcal{B}$, each data $(x_i, y_i) \in \mathcal{B}$ will be privately encoded as Equation 6.1, where the K data for mixup are randomly sampled from the batch $\mathcal{B}$. TextHide also specifies another parameter m as the size of the mask pool to facilitate the security of instance encoding against the reconstruction attacks. These formalize the (m, K)-TextHide

[10] This preliminary work is published in NAACL [230].

(Algorithm 1 in [226]), which can be integrated into the language training process to ensure text privacy. For instance, $(m = 0, K = 1)$ is the baseline training setting without protection. A larger K will sacrifice some accuracy while improving the privacy (higher costs on recovering the original data), which reflects the trade-off between privacy and accuracy for private training.

Furthermore, TextHide can utilize another dataset X_{public} (usually a large public corpus, e.g., Wikipedia) for mixup, where such mixup works similar to a *random oracle* in the cryptography domain.[11] Specifically, TextHide will mix up about one half $\lfloor K/2 \rfloor$ public data with the private original data, then Equation 6.1 is updated as:

$$\widetilde{e}_i = \sigma \circ (\sum_{j=1}^{\lfloor K/2 \rfloor} \lambda_j e_j + \sum_{j=\lfloor K/2 \rfloor + 1}^{K} \lambda_j e_j^p) \tag{6.2}$$

where $e_j^p = \phi(x_j^p), x_j^p \in X_{public}$ (randomly sampled). As a consequence, the mixed label $\widetilde{y}$ is computed by normalization with the labels of the private data (public data usually do not have labels):

$$\widetilde{y}_i = \frac{\sum_{j=1}^{[K/2]} \lambda_j y_j}{\sum_{j=1}^{[K/2]} \lambda_j} \tag{6.3}$$

In practice, given the original training dataset (denoted as X), each data $(x_i, y_i) \in X$ will be encoded for n times (usually equal to the number of training epochs).

6.3 Related Work

6.3.1 Privacy Attacks in ML. Privacy attacks against machine learning mainly

[11]The privacy notion provided by mixup in TextHide is based on a *k-vector subset sum* [231] oracle, which would require $O(n^{k/2})$ efforts to break.

consist of two categories: 1) membership inference attacks (MIA) [232–234]; 2) data reconstruction or extraction attacks. On the one hand, membership inference attacks (MIA) [232,235,236] have worked as state-of-the-art attack scheme due to its simpleness and effectiveness, where an attacker can determine whether a data point was used to train the ML model or not. Such MIAs have been commonly used for auditing training dataset privacy [237].

On the other hand, as a stronger attack primitive, data reconstruction attacks [238–241] usually refer to the attacks that could utilize auxiliary information (e.g., background knowledge) and counter measures to reconstruct or extract the original private data. For example, model inversion attacks [235] or data extraction by memorization [242] could extract private information of training dataset by querying the target model without access to dataset. Another example is that the attacker can utilize gradients to recover data [240, 243].

Our attack on TextHide works closely as the reconstruction attack [241, 244], which aims to reconstruct the original data/information from the protected data (privately encoded data). Note that Carlini et al. [241] attacks the instance encoding on images while we extend this method to the language understanding domain.

6.3.2 Privacy-Enhancing Technologies (PETs). As data privacy risks become an emerging issue, there have been a number of research works, namely, privacy-enhancing technologies (PETs) focusing on the data protection in the machine learning [215, 218], including the two main directions as following: 1) designing secure computation protocols with cryptographic building blocks to secure the data-in-use [216, 218, 245], which could achieve "perfect" secrecy but bring both extra computational and communication costs; 2) improving the privacy of machine learning algorithm with differential privacy [223, 224]. For example, a Naïve Bayes classifier can be trained by applying Laplace noise on the dataset by computing proper sensi-

tivity [224], which will be further utilized to add Laplace noise to satisfy DP notion. Another popular but different scheme, DP-SGD [223] applies the Gaussian noise into the gradients of a single data sample during the model training, which aims to bound the influence of such one individual data sample under the paradigm of differential privacy. It is worth noting that there have been recent works in NLP [246–249], which aim to empirically train/fine-tune language models to satisfy DP notion.

Both categories of privacy-enhancing schemes above can provide provable privacy guarantee for the training data. However, the instance encoding scheme may not obtain such privacy guarantee. As mentioned earlier, the instance encoding scheme [226, 227] was proposed to protect the training data's privacy by mixing up input data [228]. The paper claims that such scheme can preserve data privacy while maintaining good data utility. However, recent data reconstruction attacks [241] have shown that instance encoding lacks provable privacy guarantee. That is, the "indistinguishability" definition of privately encoded data is rather spurious, which does not comply with the concept of indistinguishability in either cryptography or DP. For example, the security of asymmetric encryption scheme could be theoretically proven by a security game (defined as IND-CPA [250]) where no adversary can win the game with significantly greater probability than an adversary with random guessing. Similarly, differential privacy [220, 223] also presents the individual data with deniability that attacker cannot differentiate it with some probability bound. Considering that TextHide fails to provide such privacy guarantee, it can be broken by the carefully designed attacks and leak the private data [229, 241].

6.4 Empirical Study 1: Privacy Attack Evaluation

6.4.1 Attack Setting. We assume that the attacker have full knowledge of the public dataset X_{public} and the embedding model for downstream ML tasks. Besides, we assume that the attacker can obtain the private dataset (but unaware of the specific

data for the training). Note that we need to consider the worst case (attacker) to evaluate the vulnerabilities of the privacy-enhancing schemes. That is, the strong knowledge (e.g., embedding model and private training dataset) can be accessed by a skilled attacker armed with any background knowledge. For instance, such private training dataset can be machine-generated. Specifically, if the dataset involves personal conversations, then the attacker can utilize some language models to generate a large set of commonly-used dialogs as the private training dataset. The attacker can also leverage some advanced inference attacks, e.g., side-channel or public essays to derive some sentences.

6.4.2 Attack Goal. Given a privately encoded dataset $\widetilde{\mathcal{E}}$ (including the mixed label $\widetilde{y}$), the attacker aims to reconstruct the original data vector $e \in \mathcal{E}$, where $\mathcal{E}$ is the set of the original data vectors. W.l.o.g., we consider the basic mixup case that the two original data vectors are used for private encoding, i.e., for one encoded data $\widetilde{e}_i$, it will be constructed on two original data e_{j_1} and e_{j_2}. Then, we denote a mapping function for the attack as $\mathcal{A}_m : \widetilde{e}_i \in \widetilde{\mathcal{E}} \rightarrow \{e_{j_1}, e_{j_2}\} \in \mathcal{E} \times \mathcal{E}$. Thus, given $\mathcal{A}_m(\widetilde{e}_i) = \{e_{j_1}, e_{j_2}\}$, the attacker seeks to derive such mapping function. Note that our attack focuses on reconstructing the text representation vectors (processed by the language understanding model, e.g., BERT) and then we can utilize the model inversion attack [240] to recover the raw text, i.e., $x_i = \phi^{-1}(e_i)$.

6.4.3 Attack Methodology. Our proposed attack consists of three main steps:

1. Removing the sign-flipping mask σ. We first nullify the sign-flipping step for encoding by taking the absolute value of the encoded data $\widetilde{e} \in \widetilde{\mathcal{E}}$ as:

$$\widetilde{\mathcal{E}} \leftarrow \{abs(\widetilde{e}), \widetilde{e} \in \widetilde{\mathcal{E}}\}. \tag{6.4}$$

2. Revealing the mapping function $\mathcal{A}_m$ to map the encoded data vector $\widetilde{\mathcal{E}}$ to the original data vector via clustering (Section 6.4.3.1).

3. Reconstructing the original text representation vector e_i (by computing the λ_i) given the mapping function $\mathcal{A}_m$ (Section 6.4.3.2).

6.4.3.1 Revealing Mapping Function. The main procedure of this step is clustering the encoded text vectors and mapping the clusters back to the original text vectors. Given a set of original data instances $|X|$ and every data instance will be encoded n times. Since each encoded text vector $\widetilde{e_i}$ is corresponding to the two original data (i.e., $\mathcal{A}_m(\widetilde{e_i}) = \{e_{j_1}, e_{j_2}\}$), the clustering result would expect to be $|X|$ clusters of size $2 * n$ encoded data vectors (the size of encoded data $\widetilde{\mathcal{E}}$ is $|X| * n$).

1) Compute Similarity Score. For the cluster of $\widetilde{\mathcal{E}}$, we first compute a similarity score $s \in [0, 1]$ among the two privately encoded data $\widetilde{e_i}$ and $\widetilde{e_j}$: if $\mathcal{A}_m(\widetilde{e_i}) \cap \mathcal{A}_m(\widetilde{e_j}) \neq \emptyset$, $s = 1$ (or close to 1), otherwise 0 (or close to 0). To compute the similarity score s, we train a neural network model $f(\cdot)$ by inputting two privately encoded vectors $(\widetilde{e_i}, \widetilde{e_j})$, and $f(\widetilde{e_i}, \widetilde{e_j}) = \{0, 1\}$. The two vectors will be stacked together (e.g., for $d \times 1$ encoded vector, the input will be $d \times 2$).

Specifically, we utilize a vanilla MLP model trained with Adam (learning rate 0.01) on the cross-entropy loss. We use the MNLI dataset (around 393k examples with all labels removed) [251] as the public dataset , and Corpus of Linguistic Acceptability (CoLA) [252], and Stanford Sentiment Treebank (SST-2) [253] as the private dataset. Then, we construct a large-scale training data pairs encoded with the above datasets by TextHide, which are labeled accordingly (1 if encoded with the same original text data; otherwise 0). The final model can achieve 94% accuracy.

Notice that reconstructing model $f(\cdot)$ by computing the similarity scores between two privately encoded data is based on a key hypothesis: *given any instance encoding scheme which achieves a high accuracy, the privacy guarantee would be somewhat weak (since the original information should be preserved with high accuracy).*

In other words, if TextHide ensures high accuracy in the downstream tasks (e.g., sentence classification), then the instance encoded data can also be "learned" to recover the original text data (model $f(\cdot)$ can be viewed as a downstream task in NLP). We identify this as an intrinsic vulnerability of such instance encoding schemes, which can be exploited to launch the reconstruction attack.

2) Clustering. Given the similarity model, we can compute the similarity scores on all pairs of the encoded data $(\widetilde{e}_i, \widetilde{e}_j)$ ($|\widetilde{\mathcal{E}}|^2$ pairs in total). This procedure can be computationally efficient. To find $|X|$ clusters (exclusive), denoted the cluster set as $\{C_p, p \in [1, |X|]\}$ w.r.t. $|X|$ original text vectors, we formulate the objective function as:

$$\max \sum_{p=1}^{|X|} \sum_{\widetilde{e}_i, \widetilde{e}_j \in C_p} f(\widetilde{e}_i, \widetilde{e}_j) \tag{6.5}$$

Ideally, the size of each cluster should be exactly $2n$, and any two encoded data $(\widetilde{e}_i, \widetilde{e}_j)$ in every cluster C_p should satisfy $f(\widetilde{e}_i, \widetilde{e}_j) = 1$ (or close to 1). Following K-NN, we can design a greedy method to iteratively update $|X|$ clusters by selecting the encoded data which has the maximum average similarity score of all the data in the cluster. Furthermore, we can audit each cluster by checking the similarity scores among the encoded data and finally partition $\widetilde{\mathcal{E}}$ into $|X|$ clusters.

6.4.3.2 Reconstructing Original Text Vectors. After deriving the mapping function from the encoded data to the original data, we can reconstruct the original data. Roughly we can sum up the absolute values of all the encoded vectors mapping to one given original data vector e and average it: $e' = \frac{1}{n} \sum abs(\widetilde{e}_i)$. The vector e' is approximately close to the original e based on two aspects: 1) the sign-flipping mask σ is removed by taking the absolute values; 2) the values of other irrelevant mixup text vectors can be "cancelled out" by the averaging (could also result in some noises

added into the vector). Thus, we need to ensure that the recovered result is close to the original result with tolerable noises.

We first recover the values of the mix-up coefficients λ via the mix-up labels. Specifically, we can get the list of λ with the mix-up labels since TextHide utilizes one-hot vector labels. For example, given one TextHide label $(0.4, 0, 0, 0.6)$, we can directly derive λ_i, λ_j as $0.4, 0.6$ (Figure 1 in [226]). Then, the attacker can directly retrieve the values of λ. Note that there exists one special case: the mixed two data could belong to the same class (the mixed label will only have one non-zero entry), and thus we can consider $\lambda_i = \lambda_j$.

After we compute the value of λ, we can reconstruct the original vector e by trying to inverse the mixup operation (Equation 6.2). Specifically, we denote Λ as an $|\mathcal{E}| \times |X|$ matrix. For each row of Λ, there are two non-zero entries i, j corresponding to the two mixup values λ_i and λ_j (other entries are 0). Denote the original text vectors as $\mathcal{X} = [e_1, \cdots, e_{|X|}]^T$ (with dimension $|X| \times d$), and the privately encoded vectors as $\mathcal{Y} = [\widetilde{e}_1, \cdots, \widetilde{e}_{|\mathcal{E}|}]^T$ (with dimension $|\mathcal{E}| \times d$). Then, Equation 6.2 can be updated as:

$$\Lambda \cdot \mathcal{X} = \mathcal{Y} + \epsilon \tag{6.6}$$

where ϵ denotes the potential introduced noises ($\mathcal{X}$ may not be exactly the original one). To compute $\mathcal{X}$, we can directly solve the above equation:

$$\mathcal{X} = \Lambda^{-1} \cdot \mathcal{Y} + \Lambda^{-1} \cdot \epsilon \tag{6.7}$$

Since the noise could subject to Gaussian distribution, the component $\Lambda^{-1} \cdot \epsilon \approx 0$ (the mean value would be close to 0, then we can average it). Furthermore, we can

formulate another optimization to minimize the "extra" noise ϵ:

$$\min_{\mathcal{X}} ||\epsilon||_2^2 \quad s.t. \;\; \epsilon = \mathcal{Y} - \lambda \cdot \mathcal{X} \tag{6.8}$$

Thus, with the minimization of the noise, we can accurately derive $\mathcal{X}$ (close to the true value). It is worth noting that $\mathcal{X}$ includes the sign-flipping mask σ. Recall that we nullify the mask σ by taking the absolute value, then Equation 6.8 can be updated:

$$\min_{\mathcal{X}} ||\epsilon||_2^2 \quad s.t. \;\; \epsilon = abs(\mathcal{Y}) - \lambda \cdot abs(\mathcal{X}) \tag{6.9}$$

where *abs* is the element-wise absolute value function of the matrix $\mathcal{X}$ or $\mathcal{Y}$. To solve Equation 6.9, we can utilize the gradient descent to search the value of $\mathcal{X}$, and thus compute the ϵ based a fit solution of $\mathcal{X}$ (w.r.t. the objective function $||\epsilon||_2^2$). Note that there may exist several values of ϵ to satisfy the constraints, then we can heuristically search the value of ϵ entry by entry to get the smallest $||\epsilon||_2^2$. Since the attackers have the full knowledge of the pre-trained language model $\phi(\cdot)$, we can directly utilize model inversion attacks [235] to recover the original text.

6.4.4 Results and Analysis. We utilize the pre-trained BERT_{base} model by [254][12] as the language model to generate the text representations (the dimensionality d is 768). We evaluate our attack on two datasets for sentence classification: 1) Corpus of Linguistic Acceptability (CoLA) [252]; 2) Stanford Sentiment Treebank (SST-2) [253] (the private datasets). For the "public dataset", we use MNLI daset [226]. We utilize the open source code of TextHide (`https://github.com/Hazelsuko07/TextHide`) to construct the private dataset. We vary the parameter $k \in [1, 2, 4, 6]$ (the

[12] `https://github.com/google-research/bert`

K	1	2	4	6
CoLA	100%	88%	91%	93%
SST-2	100%	92%	95%	88%

Table 6.1. Attack success rate on the two datasets.

number of data for mixup). We keep the size of mask pool $m = 1$. Also, we evaluate the attack performance on varying the size of mask pool $m = [1, 16, 64, 256, 1024, 4096]$. For each dataset, we randomly select 100 data points and generate 5000 encoded data via TextHide. In our attack, we will try to reconstruct the original data from such 5000 encoded data by instance encoding. We report the attack success rate (the percentage of reconstructed data out of the original data). Note that our attack is independent of datasets/applications and hyper-parameter free.

Table 6.1 illustrates the attack results (the percentage of recovering original data) on the two datasets. We can observe that our proposed attack can almost recover the text vectors (high success rate). Moreover, while TextHide claims that the privacy will increase as K increases (while losing accuracy), the results show that the value of K does not impact privacy much. Similarly, Figure 6.1 shows that the mask cannot ensure privacy (but only increasing computational costs instead). Above all, the text vectors cannot be simply viewed as "real-number" vectors since they may still contain semantic meanings (features), which may help the attacker break the security oracle more efficiently.

6.5 Empirical Study 2: Protection by Differential Privacy

A well-designed privacy-enhancing scheme must ensure provable privacy guarantee, and show its performance on data protection. Since TextHide is based on such mixup encoding method, it would be possible to apply differential privacy [220] to the

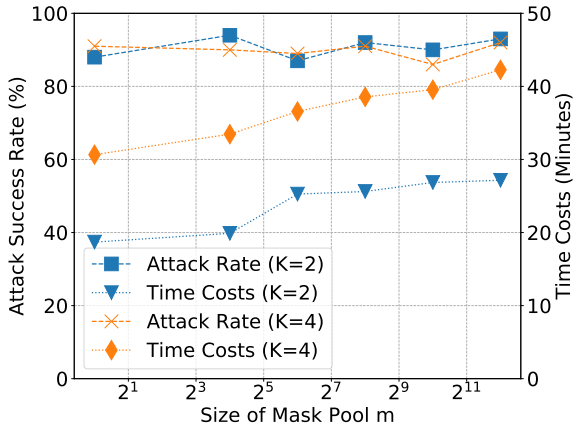


Figure 6.1. Attack success rate vs. size of mask pool m

mixup encoding and thus to show similar indistinguishability of the privately encoded instances. This can defend against our reconstruction attacks to some extent (at least reducing the information disclosure). We then demonstrate the privacy practice on the instance encoding with the differential privacy as the following.

6.5.1 Differential Privacy. As one main category of privacy-enhancing technologies, differential privacy (DP) [220, 221] has been widely used as a de facto standard notion in protecting individual's data privacy for data collection and analysis [255], especially in machine learning applications [223, 224]. The principle of the differential privacy [220, 221] is that an individual's data point x in one dataset D will not arouse significant change to the outcome of a randomized mechanism or algorithm applied to the D. Thus, the attacker cannot make difference with such a specific data point x by observing the outputs of D by the randomized mechanism, which thus provides deniability for the existence of x (ensuring data privacy).

Formally, to define individual's privacy, we first define the neighboring datasets, i.e., D, $D' \in \mathcal{D}$ are the neighbors if they only differs in one data point, denoted as $D \sim D'$. Then we define the DP notation as following:

Definition 6 (Differential Privacy [220, 221]). *For any two neighboring datasets, $D, D' \in \mathcal{D}$, a randomized mechanism $\mathcal{M}$ is said to be (ϵ, δ)-differentially private if it satisfies the following equation:*

$$Pr(\mathcal{M}(D) \in \mathcal{O}) \leq e^{\epsilon}\ Pr(\mathcal{M}(D') \in \mathcal{O}) + \delta \tag{6.10}$$

where $\mathcal{O}$ denote all the events in the output space of $\mathcal{M}$. If $\delta = 0$, $\mathcal{M}$ is ϵ-differentially private.

In this work, we will utilize the Laplace and Gaussian mechanisms to guarantee (ϵ, δ)-DP. The Laplace mechanism [220] adds the noise from Laplace distribution with mean zero and scale parameter b, denoted as $\text{Lap}(b)$ with density function $\frac{1}{2b}\exp^{\frac{-|x|}{b}}$. Formally, we have the following theorem:

Theorem 5 (Laplace Mechanism [220, 221]). *Given any function $f : \mathcal{D} \to \mathbb{R}^d$, the Laplace mechanism is defined as $\mathcal{M}_L(D, f, \epsilon) = f(D) + N$, where N is the random noise drawn from Laplace distribution $Lap(\frac{\Delta f}{\epsilon})$, and Δf is ℓ_1 sensitivity. Laplace mechanism satisfies $(\epsilon, 0)$-DP.*

Theorem 6 (Gaussian Mechanism [221, 256]). *Given any function $f : \mathcal{D} \to \mathbb{R}^d$, the Gaussian mechanism is defined as $\mathcal{M}_G(D, f, \epsilon) = f(D) + N$, where N is the random noise drawn from Gaussian Distribution $\mathcal{N}(0, \sigma^2 I_d)$ with $\sigma \geq \Delta f\sqrt{2\ln(1.25/\delta)}/\epsilon$. Δf is the ℓ_2 sensitivity of function f, i.e., $\ell_2 = \sup_{D \sim D'} ||f(D) - f(D'||_2$. Guassian mechanism satisfies (ϵ, δ)-DP.*

6.5.2 DP Instance Encoding. Given a training batch of data samples of size M $\mathcal{B} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_i, y_i)\}, i \in [1, M]$, which is randomly sampled from the training set. TextHide will first encode every sample into a feature vector of dimension size d by a pretrained feature extractor $\phi(\cdot)$, i.e., $v_i = \phi(x_i)$. Then we can get the corresponding batch of encoded feature vectors $\mathcal{B}_e = \{(v_1, y_1), (v_2, y_2), \cdots, (v_N, y_N)\}$.

For original instance encoding, TextHide would mixup such set of size k vectors to generate private encoded vectors as training data per Equation 6.1. To address the privacy issue, we apply the differential private mechanism to such mixup process. Algorithm 13 demonstrates the details.

Input: Batch of encoded vectors $\mathcal{B}_e$,
Number of mixed data samples k,
clip bound for encoder vectors C
DP Noise $\mathcal{M}$: Laplace, Gaussian

Output: Differentially private encoded vector set $\mathcal{B}_{dp}$ of size $|\mathcal{B}_{dp}|$

Initialize DP mechanism $\mathcal{M} = \{\mathcal{M}_L, \mathcal{M}_G\}$
Randomly sample K mixup coefficients: $\Sigma_i^K \lambda_i = 1, \lambda_i \in \mathcal{N}(0, I)$
`// Instance Encoding by mixup`
Randomly sample K data samples from $\mathcal{B}_e$
for $i \to 1$ *to* $|\mathcal{B}_e|$ **do**
    `// Clip Input Vector`
    $v_i \leftarrow v_i \cdot \min(1, \frac{C}{||v_i||_2})$
if $\mathcal{M}_G$ **then**
    $N \leftarrow^s \mathcal{N}(0, \sigma^2 I_d)$
else
    $N \leftarrow^s \frac{\epsilon}{4C} \exp^{\frac{-\epsilon|x|}{2C}}$
for $j \to 1$ *to* $|\mathcal{B}_{dp}|$ **do**
    $\widetilde{v}_j \leftarrow \sum_{i=1}^K \lambda_i v_i + N$
    $\widetilde{y}_j \leftarrow \sum_{i=1}^K \lambda_i y_i$
return $|\mathcal{B}_{dp}|$ private encoded data vectors

Algorithm 13: DP Instance Encoding

Theorem 7. *The DP Instance Encoding revised with Laplace noise satisfies* $(\epsilon, 0)$*-DP,*

where the added noise N_L is draw from Laplace distribution as following:

$$N_L = \frac{\epsilon}{4C} \exp^{\frac{-\epsilon|x|}{2C}} \tag{6.11}$$

Proof. The proof complies with the original proof of Laplace mechanism [220, 221]. The instance encoding scheme with clipping works as the function f. The ℓ_1 sensitivity here is $2C$ since the maximum ℓ_1 norm difference of two vectors are $2C$ (viewed as a hyper-sphere of radius C). Then replacing Δf with $2C$ in Laplace distribution, we get the Equation 6.11. It has shown that adding Laplace noise sampled from Eq. 6.11 satisfies ϵ-DP [220], i.e., the DP instance encoding with $\mathcal{M}_L$ satisfies $(\epsilon, 0)$-DP. □

Theorem 8. *The DP Instance Encoding revised with Gaussian noise satisfies (ϵ, δ)-DP.*

Proof. Similar to the previous proof for Laplace, we choose the Gaussian distribution $\mathcal{N}(0, \sigma^2)$ with mean zero and standard deviation $\sigma^2 = (\frac{1+\sqrt{2\log(1/\delta)}}{\epsilon})^2 C^2$, where the ℓ_2 sensitivity is C. Note that the input vectors are multi-dimensional, and the noise added will be drawn independently from $\mathcal{M}_G$. Then we can derive that DP instance encoding with $\mathcal{M}_G$ satisfies (ϵ, δ)-DP. □

6.5.3 Experimental Evaluation. For experiments, we would like to evaluate both utility and privacy of the proposed scheme as the following: 1) utility of the private instance encoding scheme, i.e., the performance (accuracy) of model trained on the private dataset; 2) privacy guarantee of the scheme against reconstruction attacks, i.e., the attack success rate (the percentage of reconstructed private vectors).

Dataset. We consider the sentence classification task with two popular datasets: 1) Corpus of Linguistic Acceptability (CoLA) [252] (about 8500 training samples) for acceptability; 2) Stanford Sentiment Treebank (SST-2) [253] (about 67000 samples) for sentiment analysis.

Model Implementation. We use the pre-trained BERT model [254] as the language feature extractor to generate the text representation vectors (the dimensionality d is 768). Note that TextHide will encode such representation vectors into the training vectors for downstream tasks. For downstream task training, we follow TextHide to choose a multilayer perceptron of hidden-layer size $(768, 768, 768)$ since we take TextHide as baseline.

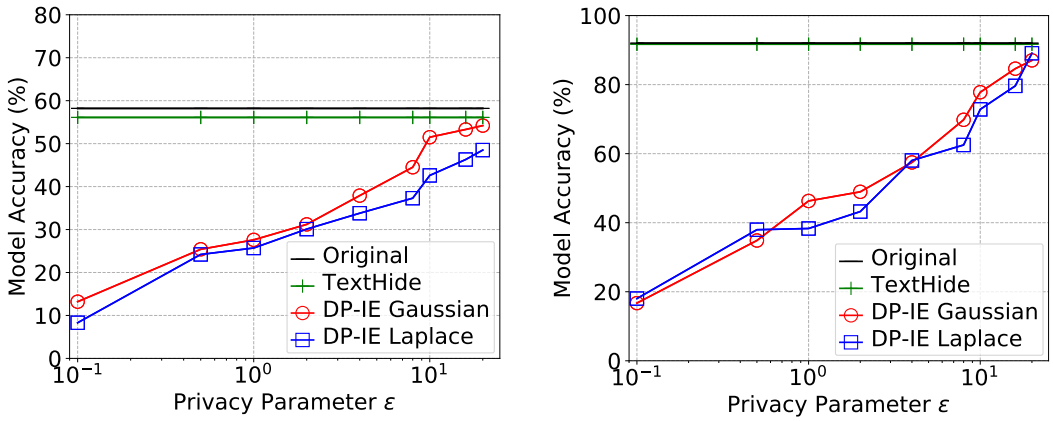
Utility Evaluation. We will apply our scheme (including Gaussian and Laplace mechanism, denoted as "DP-IE Gaussian" and "DP-IE Laplace", respectively) and TextHide to the two datasets during training, and then report the model accuracy, respectively. In addition, we will also demonstrate the accuracy of the raw dataset (without any privacy protection scheme) for better utility comparison.

Privacy Evaluation. To fully evaluate the proposed DP instance encoding scheme, we also utilize a privacy reconstruction attack [229] on instance encoding scheme. Specifically, we first construct a set of private vectors generated by our proposed scheme and TextHide (as baseline), respectively. We report the final attack success rate (the percentage of reconstructed data vectors out of the original set) by implementing reconstruction attack on the generated vectors above.

6.5.4 Utility Evaluation. For our proposed scheme, we set the privacy parameter $\epsilon = \{0.1, 1, 2, 4, 8, 10, 15, 20\}$. For Gaussian mechanism, we set δ to be 10^{-5}. Then we evaluate the model accuracy with varied ϵ for both Laplace and Gaussian mechanism on the two datasets as depicted above. For TextHide, we select $(m = 16, k = 4)$ as its own privacy parameters. We also evaluate the base case (without any privacy-protection scheme). We report the final model accuracy (the testing performance of trained model on the private dataset).

Figure 6.2 demonstrates the results. From the figure, we can observe that the

model accuracy increases as the private parameter ϵ increases for both Gaussian and Laplace. This is reasonable since the privacy parameter ϵ of the DP schemes works as the privacy budget to determine the privacy-protection level for the dataset. That is, the larger the privacy budget, the smaller the noise added to the original data vectors (the privacy-protection would be weaker). As a result, the utility of the training set would not be affected too much. In addition, we can also observe that the model accuracy can approach the base case as ϵ increases, which will cause the compromise of privacy to some extent (as shown in the privacy evaluation).

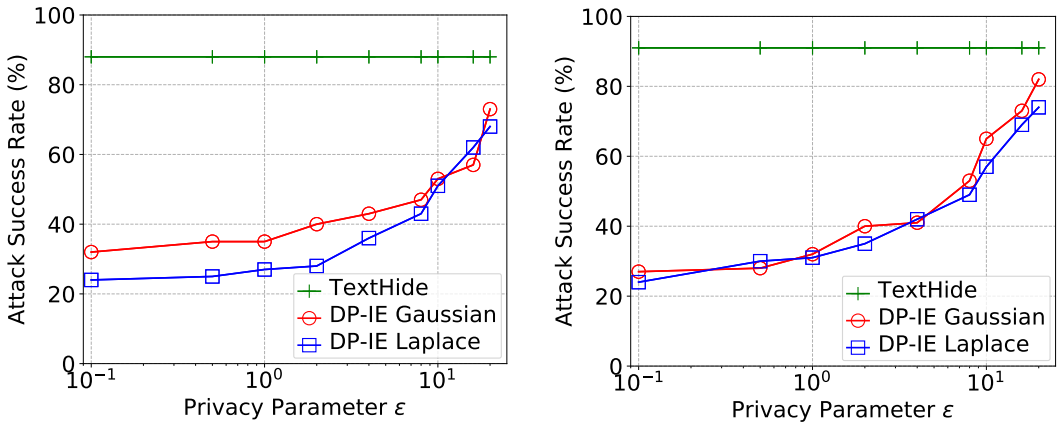


(a) CoLA (b) SST-2

Figure 6.2. Accuracy (learning utility) on the two datasets with DP-IE schemes

6.5.5 Privacy Attack Evaluation. We follow the attack model setting [229, 241] that the attacker could obtain the background knowledge of the private dataset but be unaware of the specific data for training, which would utilize any auxiliary information to reconstruct the vectors (as a strong attack). We reproduce the attack scheme following the attack proposed in [229]. More specifically, we randomly select 100 data points and generate 5000 encoded data by our DP schemes for each dataset, respectively. We measure the attack results with varying values of the privacy parameter $\epsilon = \{0.1, 1, 2, 4, 8, 10, 15, 20\}$ (referring to different levels for privacy-protection). For

example, $\epsilon = 0.1$ is the strong protection and 20 is a weak protection. We repeat the same process for TextHide using the same privacy parameter as the previous utility evaluation.



(a) CoLA (b) SST-2

Figure 6.3. Attack success rate on the two datasets with DP-IE schemes

Figure 6.3 demonstrates the final attack results. First, we can observe that the TextHide cannot ensure data privacy against privacy attacks, i.e., the privacy attack can recover around 85% of the original data vectors for both CoLA and SST-2 dataset. This also conforms to the previous works. Second, the results show that our proposed DP scheme can defend against such privacy attack from reconstructing the data. Take Figure 6.3(a) as an example, the overall attack success rate is lower than the baseline's. Besides, the attack success rate increases as the privacy parameter ϵ increases, which indicates that a higher privacy budget will lead weaker protection by differential privacy. Such results also validate the previous DP theorems. Again, it should be noted that DP cannot prevent leakage of the dataset completely. Instead, we would like to achieve a proper utility-privacy trade-off while applying differential privacy to the machine learning applications. For example, some privacy-sensitive applications, e.g., on-device input prediction, could require strong privacy guarantee

while tolerating a fair utility loss. We can also improve our instance encoding scheme with other techniques, e.g., Federated Learning [257] or optimize the privacy budget to get a better utility accordingly.

CHAPTER 7

PRIVACY-PRESERVING CLOUD-BASED DNN INFERENCE

7.1 Introduction

Deep neural network (DNN) models have been frequently deployed in a wide variety of real world applications, such as image classification [258], video recognition [259] and voice assistant (e.g., Apple Siri and Google Assistant). Meanwhile, cloud computing technologies (e.g., Microsoft Azure Machine Learning, Google Inference API, and Amazon AWS Machine Learning) have promoted the deep learning as a service (DLaaS) to make DNNs widely accessible. Users can outsource their own data for inferences based on the pre-trained DNN models provided by the cloud service provider.

However, severe privacy concerns may arise in such applications. First, if the data of the clients are explicitly disclosed to the cloud, sensitive personal information included in the outsourced data would be leaked. Second, if the fine-tuned DNN models are shared for inferences [260], the parameters might be reconstructed by untrusted parties [261]. To address such privacy concerns, several recent works [262–265] have proposed cryptographic protocols to ensure privacy in inferences via garbled circuits [22] and/or homomorphic encryption [219]), which rely on expensive cryptographic primitives. Then, such protocols may result in fairly high computation and communication overheads. Since the volume of the outsourced data grows rapidly and the DNN models usually require high computational resources in the cloud, such techniques may not be suitable for practical deployment due to limited scalability. Thus, we are seeking an efficient scheme to securely implement the DNN inferences in the cloud.

Specifically[13], we propose a privacy-preserving cloud-based DNN inference framework ("PROUD") by co-designing the cryptographic primitives, deep learning, and cloud computing technologies. We mainly take advantage of a novel matrix permutation with ciphertext packing and parallelization to improve the computational efficiency of linear layers. With the privacy guarantee provided via homomorphic encryption, PROUD supports all types of non-linear activation functions by leveraging an interactive paradigm. Above all, PROUD integrates the cloud container technology to further improve the performance via parallel execution, which can also be readily adapted for various DNNs via configuring container images.

7.2 System Overview

Figure 7.1 illustrates the framework of the proposed system for the users (clients) and the cloud service provider (cloud server). The client locally holds the private data, which will be encrypted with the client's public key and sent to the cloud server. Then, the cloud server initializes container instances (pre-compiled with secure protocols, i.e., MatF and NlnF) to execute the DNN inference with the encrypted input. Finally, the client will decrypt and receive the classification result.

Automated Backend Execution. The backend system can automatically deploy the cryptographic protocol for the secure data inference in the cloud. Specifically, once the server receives encrypted data from the client, it will compose the configuration file to initialize a bunch of container instances via a pre-compiled image (with the source codes), where the secure protocols (i.e., MatF and NlnF) will start to be executed for DNN inference until the final result is returned. The automation of the backend ensures that the secure protocols can be delivered efficiently, and enables the full system to be capable of processing a large number of clients (if necessary).

[13]This work has been published on IEEE ICASSP [266].

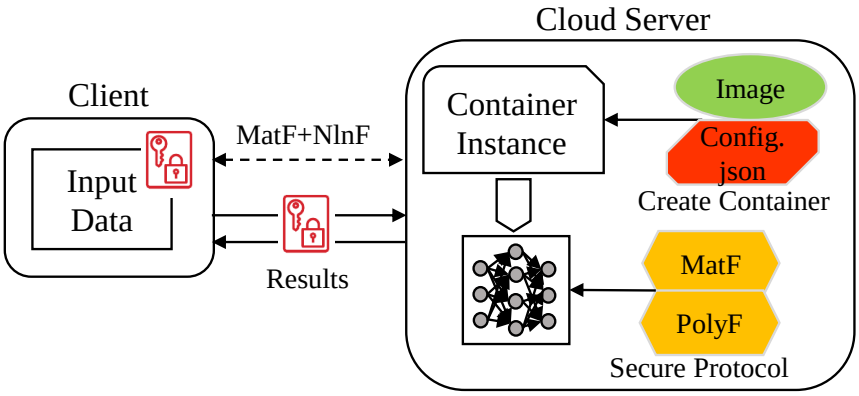


Figure 7.1. The PROUD Framework

7.3 Protocol Design

7.3.1 Problem Formulation. The PROUD will securely compute the DNN model with encrypted inputs in the cloud. We first denote an ℓ-layer DNN model as $\mathcal{M} = \{L_i, i \in [1, \ell]\}$, and the input video as $\mathcal{V}$. The inference model $\mathcal{M}$ can be viewed as a complex function $f(\cdot)$ integrating linear functions (corresponding to linear layers, e.g., convolution layers and fully-connected layers) and non-linear functions (activation functions, e.g., Sigmoid and ReLu). Denoting the inference result as $\mathcal{S}$, we have:

$$\mathcal{S} = f(\mathcal{V}) = L_\ell(L_{\ell-1}(\cdots L_2(L_1(\mathcal{V}))\cdots)) \tag{7.1}$$

Threat Model. We consider *semi-honest* model where both parties are honest to execute the protocol but are curious to learn private information. PROUD can preserve privacy for both parties against possible leakage: (1) client's private input videos are not leaked to the cloud service provider; (2) cloud service provider's DNN model (e.g., linear/non-linear weight parameters, and bias values) is not revealed to the client in the computation. We also assume that all the communications are executed in a secure and authenticated channel.

7.3.2 Protocol Overview. Algorithm 14 illustrates the protocol for PROUD.

In the initialization phase, the client generates a key pair and encrypts the private data $\mathcal{V}$ (Line 1) while the server prepares the computation for the DNN functions (Equation 7.1) with two subprotocols: (1) MatF for the linear functions; (2) NlnF for the non-linear activation functions (Line 2). With such two subprotocols, PROUD will be jointly executed by both the client and server. Specifically, the server can perform computation of the linear layers directly on the encrypted data received from the client using the subprotocol MatF (Line 5). For the non-linear layers, the output data will be sent back to the client for computation by the subprotocol NlnF (Line 6), and then the client will re-encode and encrypt the data to be sent to the server for next layer's computation. Once completing the computations of all the layers in the DNN model, the client will receive the ciphertext and decrypt it to get the classification result. The details of two subprotocols will be illustrated in Section 7.3.3 and 7.3.4, respectively.

Input: Input Data $\mathcal{V}$, $\mathcal{M}$

Output: Classification Result $\mathcal{S}$

1 Client: Encode and encrypt $\mathcal{V}$ to get τ_0

2 Server: (MatF, NlnF) $\leftarrow \mathcal{M}$

3 **for** $i \in [1, \ell]$ **do**

4 **switch** L_i **do**

5 **Case** *Linear*: $\tau_i \leftarrow \text{MatF}(\tau_{i-1})$

6 **Case** *Non-Linear*: $\tau_i \leftarrow \text{NlnF}(\tau_{i-1})$

7 Client: Decrypts τ_ℓ to get $\mathcal{S}$

Algorithm 14: PROUD Protocol

7.3.3 MatF Protocol. To ensure privacy for the linear layers, a naive method is to apply homomorphic encryption (HE) to the arithmetic operations of encrypted matrices (e.g., fully-connected layer), which might be inefficient since the input data tensors

are usually high-dimensional. To mitigate such issue, our PROUD system utilizes a novel matrix permutation method [260] to efficiently perform matrix computations with ciphertext packing and parallelization [267], where the matrix multiplication equals the sum of the component-wise products for some specific permutations of the matrices themselves.

Given the input matrix V, the linear layer (matrix) W and bias parameter B, PROUD will securely compute the function of a linear layer as: $W * V + B$ (w.l.o.g., we consider the fully-connected layer with bias while W and V are two square matrices with size $n \times n$). We illustrate an example of the square matrix as A (of size $n \times n$). To compute the multiplication, the server will first find n permutations of the matrix A via the following symmetric permutations:

$$\sigma(A)_{i,j} = A_{i,i+j}, \tau(A)_{i,j} = A_{i+j,j} \tag{7.2}$$

$$\phi(A)_{i,j} = A_{i,j+1}, \psi(A)_{i,j} = A_{i+1,j} \tag{7.3}$$

Note that ϕ, ψ are the column and row shifting operations. Then, we can compute the product for W and V as below:

$$W * V = \sum_{k=0}^{n-1} W_k \odot V_k \tag{7.4}$$

where $W_k = \phi^k(\sigma(W))$, $V_k = \psi^k(\tau(B))$, $\odot$ indicates the component-wise product and k is the number of perturbations, e.g., ψ^k will perform k times $\psi(\cdot)$ permutation on the matrix. We denote the function $permut(\cdot)$ to compute the n permutation matrices of one matrix.

Ciphertext Packing and Parallelization. To improve the efficiency, we also leverage the vectorable homomorphic encryption (aka. "Ciphertext Packing") [260,262], which transforms a matrix of size $d \times d$ to a single vector (plaintext) via an encoding

map function, denoted as *Encode*. In particular, the *Decode* function transforms the vector plaintext back to the matrix form. For simplicity of notations, we denote the encryption, evaluation, and decryption functions under an HE scheme as $Enc(), Eval()$ and $Dec()$, respectively.

Then, the component-wise product (Equation 7.4) of the ciphertexts V_k and W_k, denote as $Enc(pk, O_k)$, can be securely computed with the multiplicative property of the HE:

$$Eval(pk, Encode(W_k^{(l,m)}), Enc(Encode(V_k^{(l,m)})), *) \tag{7.5}$$

where $l, m \in [1, n]$ are the entry indices of the matrices W and V, and pk is the public key. Then, the sum of all the n component-wise products of the matrices W_k and V_k can be computed using the additive property of HE. Finally, the bias parameter B can be computed using the additive property of HE. The protocol is detailed in Algorithm 15.

Given a large number of plaintexts to be encrypted by ciphertext packing, we further expedite the matrix computation with the parallelization [260]. To this end, we modify the encoding map function to "1-to-1 map" such that an n-dimensional vector can be transformed into a g-tuple of square matrices of order d, where $g = n/d^2$. This parallelization technique can also be realized with the parallel computation in the cloud framework (using a bunch of containers), which results in a reduced computational complexity $O(d/g)$ per matrix.

7.3.4 NlnF Protocol. The NlnF protocol securely computes the non-linear layers of DNNs. Most of the existing works depend on either garbled circuits [262] or replacing square function [260], which may arouse high computational overheads or reduce the accuracy. In our protocol, the computation of the non-linear function

Input: Input V, Weighted Matrix W, Bias B

Output: $O = Enc(pk, W * V + B)$

1 $\{V_k\}_{k=0}^{n-1} \leftarrow Enc(pk, Encode(permut(V)))$

2 $\{W_k\}_{k=0}^{n-1} \leftarrow Encode(permut(W))$

3 **for** $k \in [0, n-1]$ **do**

4 $\quad O_k \leftarrow Eval(pk, W_k^{(l,m)}, V_k^{(l,m)}), *)$

5 $Enc(pk, O) \leftarrow Eval(pk, \{O_k, k \in [0, n-1]\}, +)$

6 **return** $Eval(pk, Enc(pk, O), B, +)$

Algorithm 15: MatF

(e.g., ReLu) is executed at the client side with the input of decrypted data to preserve privacy. Algorithm 16 shows that the client will first decrypt the received output of MatF from the server with its private key. Then, the client will compute the output of the non-linear function ϕ and return the output to the server for the computation of next network layer. During the execution of this protocol, the client does not leak any private information to the server and the server does not expose sensitive weight parameters to the client.

Input: Input V (from MatF), Activation Function $\phi(\cdot)$

Output: O

1 = Server: sends V to the client

2 Client: $r \leftarrow Decode(Dec(sk, V))$

3 **return** $O \leftarrow \phi(r)$

Algorithm 16: NlnF

Security and Practicality. For the linear computations (MatF), the server will not know the plaintext since all the computations are performed on the ciphertexts ("no leakage" can be theoretically proven). For the non-linear computations, the client receives some encrypted intermediate results from the server, and decrypts them to get

some trivial intermediate data (which does not result in privacy leakage). Such trivial non-private data release is traded for a light-weight cryptographic protocol, which is far more efficient than other cryptographic protocols built on secure polynomial approximation and/or garbled circuits. Since the protocol is composed independently, many neural network based applications (e.g., image classification [258] and natural language processing [268]) or video learning models (e.g., C3D [259] and I3D [269]) can be readily integrated into our system. The pre-trained DNNs can be adapted with appropriate extensions, and integrated into the PROUD protocol (for feature extraction and/or inferences on the encrypted data). Moreover, the PROUD system can be easily integrated into the practical cloud platform (e.g., AWS) since the PROUD is a cloud-based prototype of system.

7.4 Experiments

Experimental Setup. Our system is implemented on the NSF CloudLab platform[14] in which one machine works as the client and the other one works as the server. Both machines have eight 64-bit ARMv8 cores with 2.4GHZ, 64GB memory installed with Ubuntu 16.04. We implement the homomorphic encryption in HEANN [267] (which realizes the optimal computation over real numbers) for secure matrix operations. We leverage Docker to develop the prototype for PROUD: the image of the container (all the source codes) is pre-compiled with the specific functions (i.e., MatF and NlnF) in Python.

We evaluate our framework on the two datasets: (1) MNIST dataset [270] includes 70K handwritten images of size 28×28 under the gray level 0-255; (2) IDC dataset[15] for invasive ductal carcinoma (IDC) classification (IDC-negative or positive),

[14]https://www.cloudlab.us/

[15]http://www.andrewjanowczyk.com/use-case-6-invasive-ductal-carcinoma-idc-segmentation/

Table 7.1. Benchmarking on MNIST dataset

Framework	Accuracy (%)	Latency (s)	Comm. (MB)
CryptoNets	96.09	1080.3	595.5
GAZELLE	99.05	8.05	100.65
BAYHENN	98.93	2.34	20.81
DELPHI	96.2	0.84	0.81
PROUD	99.01	0.62	1.03

which contains about 28K patches of 50×50 pixels. We employ the LeNet5 [270] as the test network model. In addition, we compare the performance of PROUD with four representative schemes (CryptoNets [263], GAZELLE [262], BAYHENN [264] and DELPHI [271]) on the MNIST and IDC dataset for image classification.

Results. All the results on the two datasets are shown in Table 7.1 and 7.2, respectively. From the Table 7.1, we can observe that our PROUD results in the least average latency (e.g., 13 times faster than GAZELLE) and communication overheads for digit classification, compared with other three existing schemes. PROUD significantly outperforms other schemes considering we adopt a highly light-weight matrix computation scheme compared with the existing schemes (including garbled circuits and heavily encrypting matrices). As for the classification accuracy, PROUD works almost identical as GAZELLE (in which the optimal approximation of non-linear function achieves the negligible loss using the original activation function). It is worth noting that CryptoNets performs the worst, since it replaces all the activation functions with the square functions, and all the pooling functions with sum pooling, which also greatly increase the computational overhead and arouse the high communication bandwidth (the larger ciphertext size). BAYHENN uses a different Bayesian inference model with

Table 7.2. Benchmarking on IDC dataset

Framework	Accuracy (%)	Latency (s)	Comm. (MB)
CryptoNets	81.25	1942.6	1621.3
GAZELLE	83.74	24.64	263.4
BAYHENN	83.26	9.36	67.32
DELPHI	82.72	2.47	2.95
PROUD	84.01	1.12	3.27

some randomness for DNN, which decreases the classification accuracy to some extent. Also, considering that the DELPHI's computation overheads are mainly in the offline preparation (heavy cryptographic computations), the online computation overhead is reduced. Table 7.2 shows similar results for IDC classification. All of these results illustrate that our proposed framework can significantly improve the computational efficiency of secure DNN inference compared with other SOTA techniques.

We also illustrate the results of latency and communication bandwidth result for each step of PROUD processing one image instance in Table 7.3. Specifically, the client takes about 23.4 ms, including the runtime for encoding and encrypting the image. Then, the server initializes the DNN model by taking 107.2 ms (note that this step can be processed simultaneously as the first step at the client's). Moreover, we also observe that DNN computation in the server dominates the latency. Regarding the communication overheads, it mainly occurs when the client sends the encrypted images to the server (0.58MB). Moreover, there arouses communication consumption during the DNN inference since NlnF protocol follows an interactive paradigm.

7.5 Related Work

Table 7.3. Performance of PROUD on MNIST dataset

	Phase	Latency (ms)	Comm. (MB)
Client	Encode + Encry.	23.4	0.58
Server	Set Model	107.2	-
Server	DNN Computation	410.8	0.34
Client	Decry.+ Decode	2.7	0.03
	Total	544.1	0.95

The generic secure computation techniques (e.g., secure two-party computation [22, 272], fully homomorphic encryption [273] and secret sharing [274]) can be directly used to tackle the privacy concerns in DNN inferences. However, such cryptographic primitives would request high computation and communication overheads. For instance, the size of garbled circuits in the MPC protocols will exponentially grow as the number of parties increases. They also require multiple rounds of communications among the parties. Recently, although there are multiple works that improve the efficiency of FHE [275–277], the high computational costs still make them impractical for performing inferences.

Therefore, it seems to be necessary to design specific protocols for secure learning. There have been several works on designing specific secure protocols for DNN models [262–264, 278]. SecureML [278] is one of the first systems which focuses on machine learning on encrypted data with NN model. However, it mainly depends on the generic two-party protocols with very poor performance. Jiang et al. [260] proposed an efficient secure matrix computation protocol to improve the performance for the computation with neural networks. However, it only supports limited activation functions (e.g., only the square function in the case study). GAZELLE [262] composes

the protocol on the heavy garbled circuits while BAYHENN [264] leverages inefficient ciphertext packing of matrix for linear computations. Although DELPHI improves the bandwidth of online protocol, it still depends on the off-line computation and neural architecture search (NAS).

CHAPTER 8
SECURE OUTSOURCING COMPUTATION ON THE CLOUD

8.1 Introduction

With the significant development of cloud computing, an increasing number of data-related services such as data analysis and storage have been prevalently outsourced to the cloud [279]. In practice, outsourcing data analyses to service providers (e.g., the cloud) would request the data owner to share their original data. This may result in immense privacy concerns of the data owners with severe consequences for the enterprises [280]. As data leaking incidents become even more severe, a GDPR article [281] states that enterprises cannot share their sensitive data without sufficient protection, while acquiring the third-party services.

To date, different types of encryption algorithms may be applied to protect the outsourced data. First, encrypting the datasets using a traditional algorithm like AES [282] or 3DES [283] may prevent the external service providers from conducting useful data analysis, whereas Homomorphic encryption (including fully) [24, 76, 284] might be too expensive and inflexible for different analyses. The recent property preserving encryption schemes [285–287] have enabled service providers to perform efficient and accurate data analyses on the encrypted data, which are deterministic ciphertexts to retain a certain property of their plaintexts. Examples include hashing where the ciphertexts reveal the equality of messages, order preserving encryption (OPE) [287,288], where the ciphertexts retain the ordering of data, and prefix preserving encryption (PrefixPE) [286], where the ciphertexts share the same length of prefixes as shared between the plaintexts. However, most existing property preserving encryption schemes [285–287] as mentioned above have the following two major limitations.

First, property preserving encryption is typically limited to specific data or

applications. For instance, OPE is mostly applied in range queries based analysis on numerical data. While achieving more prefix-based utility (e.g., network trace analysis [289]), PrefixPE (e.g., CryptoPAn [286]) is only applicable to IP addresses, which is an important limitation since PrefixPE may potentially benefit a wide variety of outsourced data analyses on different datasets. As discussed in Section 8.2.2, besides IP addresses, some other data types (e.g., geo-location data, DNA sequences, market basket items, and timestamps) can be encoded with meaningful prefixes to retain very high utility in the encrypted data. For instance, in a dataset collected from location based services (LBS), two places which are close in the plaintexts (e.g., central park and the empire state building in New York) can be converted to ciphertexts (sharing the same length of prefix to maintain the same spatial distance) as two places in another city. Thus, it is highly desirable for a service provider to analyze the prefix preserving encrypted data. [16]

Second, it is well known that most property preserving encryption techniques are vulnerable to various forms of inference attacks [292–295], which attempt to link the encrypted data to original data with background knowledge or auxiliary data. As discussed in Section 8.3.1, due to the deterministic ciphertexts, PrefixPE (e.g., CryptoPAn [286]) is also vulnerable to the emerging inference attacks [292]. We have also conducted experiments on the inferences attacks in Section 8.7.1 to validate this limitation.

In this paper, we first propose a novel scheme to encode a variety of data types into bit strings (*prefix-aware encoding*), and then propose a framework for outsourcing different types of prefix preserving encrypted datasets by generalizing a multi-view approach [289] to significantly reduce the information leakage against inference attacks. Specifically, the proposed prefix-aware encoding converts various types of data into

[16]This work has been published on IEEE TKDE [290] and ICDE [291].

the prefix-aware data (viz. bit strings with prefix-based utility), and then the prefix preserving encryption, i.e., CryptoPAn [286] (originally on 32-bit IPv4 addresses or 128-bit IPv6 addresses) can be generalized to encrypt *any type of data that can be encoded into the bit strings with utility resulted from the preserved prefixes. Essentially, if any data is naturally hierarchical (e.g., IP addresses as bit strings) or can be indexed by a prefix-aware tree (e.g., location data, and market basket data. See Section* 8.2.2), the prefixes in the encoded bit strings could be preserved to ensure utility when directly analyzing the outsourced data. For instance, the distance between any two locations can be fully preserved in the outsourced data.

Furthermore, to address the inference attacks, the multi-view outsourcing framework will generate multiple *indistinguishable* data views in which one real data view fully preserves the utility (ensuring 100% accuracy while performing data analyses on the prefix preserved data), and its corresponding analysis result can also be privately retrieved.

Contributions. The primary contributions focus on the generalization of the CryptoPAn to broader data types and applications from two aspects. First, we revise the CryptoPAn by extending the input size of block cipher function (cryptographic building block) to encrypt any length of bit strings rather than fixed 32-bit IPv4 addresses or 128-bit IPv6 addresses. Second, we generalize the multi-view outsourcing framework [289] with the following major improvements: (1) encrypting multiple types of data with the new prefix-aware encoding scheme and the above generalized CryptoPAn (for any length of data), (2) incorporating more and stronger inference attacks [292, 296–298] into the threat model, and (3) generating a minimum number of data views given formally defined privacy leakage bound (say Γ-Leakage). Moreover, other contributions are summarized as below.

- To our best knowledge, we propose the first general purpose prefix encryption scheme, which can potentially be applied to a wide variety of data types and applications such as geo-locations, DNA sequences, market basket datasets, and timestamps.

- The generalized multi-view outsourcing framework provides an additional layer of protection that can make the vulnerable PrefixPE scheme (i.e., CryptoPAn) sufficiently secure against various inference attacks (e.g., [292, 296–298]).

- Besides privacy and utility guarantees, our approach offers negligible communication overheads, and the computational costs can be easily adjusted based on any bounded leakage w.r.t. inference attacks.

- We empirically evaluate the performance of numerous inference attacks [292, 296–298] on the PrefixPE encrypted data using real datasets (the *check-in location* dataset and the *network traffic* dataset) and our generalized multi-view outsourcing framework. The experimental results demonstrate that our proposed framework preserves both privacy (with bounded leakage and indistinguishability of data views) and utility (with 100% analysis accuracy).

8.2 PrefixPE and Prefix-aware Encoding

8.2.1 Generalized CryptoPAn. As a PrefixPE scheme, CryptoPAn [286] was originally designed to generate deterministic ciphertexts for IP addresses (32-bit ciphertexts for IPv4 and 128-bit ciphertexts for IPv6). We first generalize CryptoPAn for any n-bit data as below:

Definition 7 (Generalized Prefix Preserving Encryption [286])**.** *Given two n-bit strings $a = a_1a_2a_3...a_n$ and $b = b_1b_2b_3...b_n$, if a and b share a k-bit ($0 \leq k \leq n$) prefix, we have $a_1a_2...a_k = b_1b_2...b_k$ and $a_{k+1} \neq b_{k+1}$. An encryption function $f(\cdot) : \{0,1\}^n \rightarrow \{0,1\}^n$ is said to be prefix-preserving, if $f(a)$ and $f(b)$ also share a k-bit prefix.*

To encrypt each bit, CryptoPAn applies a cryptographic function (including padding, a block cipher function such as Rijndael [282] with a 256/128-bit key K, and the least significant bit function) to the bits, and then XOR with the current bit to ensure the prefix preserving property [286].

Theorem 9. *CryptoPAn has the following two properties [286]: (1) associative property: given an n-bit string* $a \in \{0,1\}^n$,

$$1 \leq i, j \leq n : f^j(f^i(a, K)) = f^{(i+j)}(a, K) \tag{8.1}$$

and (2) inverse property: given two n-bit strings a and b,

$$\text{if } f(a, K) = b, f^{-1}(b, K) = a \tag{8.2}$$

Similar to $f(\cdot)$*, for each bit in b, the inverse CryptoPAn* $f^{-1}(\cdot)$ *applies the same cryptographic function to the bit's corresponding prefix in a (which was computed while applying* $f^{-1}(\cdot)$ *to the previous bits) and XOR with the current bit.*

Theorem 9 can be directly proven with the extension from 32-bit IP addresses in [289] to generalized CryptoPAn.

8.2.2 Prefix-aware Encoding. We can directly apply CryptoPAn to encrypt the sensitive IP addresses [286, 289] for outsourcing the network traffic since IP addresses automatically hold the prefix property (32-bit IPv4 addresses or 128-bit IPv6 addresses). The utility of preserving prefixes in the encrypted IP addresses can be realized for many analyses since the ciphertexts can preserve all the subnet structure of the original data (sharing a prefix in the original IP addresses also results in sharing the same length of prefix in the encrypted IP addresses).

Motivated by such prefix properties, many other types of datasets can also be encoded into prefix-aware bits such as the geo-location [299], DNA sequences [300],

items in market baskets [301], numerical data [302], and timestamps [303] (ensuring utility for performing different analyses on the encrypted data) using a prefix-aware tree:

Definition 8 (Prefix-aware Tree)**.** *Given the data domain, we generate a balanced tree in which nodes (from the root to leaf) signify a sequence of prefixes, all the sibling nodes share the same prefix. Then, each value in the domain can be represented by the bits concatenated from the top (except the root) to each leaf node.*

Example 1 (Prefix-aware Tree for IPv4)**.** *IPv4 adopts 32 bits to form the* 2^{32} *addresses, thus its prefix-aware tree is a full binary tree with 33 levels (including the root node), where each IP address is formed by concatenating the bits from the top to each leaf node.*

In the following, we will discuss the prefix-aware encoding for some representative data types, and illustrate the corresponding prefix preserving data analysis.

8.2.2.1 Locations in Location-based Services (LBS). The location data usually include the 2-dimensional latitude and longitude coordinates of different places, which are highly precise float numbers (up to 8 decimal digits) for representing the locations in map applications, e.g., Google Map and Bing Map. In the *Bing Map Tiles System* [304], the map is recursively divided into four tiles equally to reach the required resolution for users to quick map zoom in/out. Specifically, given the resolution, the system can map the longitude and latitude coordinates to bit strings called *quad key*, which is uniquely represented as the index of tile for the coordinates (can be used for map image retrieval).

Motivated by such hierarchical structure, we encode the coordinates into bit strings by concatenating the index of each level for one specific location. As shown in Figure 8.1, there is a root node at the top. At each level, the four children of each

node can be encoded using two bits $00, 01, 10, 11$ to represent four tiles, respectively. Thus, after concatenating the bits from the first level, every location can be encoded by a leaf node, and all the locations can be encoded with the same length of bits if the coordinates use the same precision. Location precision can be increased with additional levels and longer bit strings. Our experimental location data utilizes a length of 46 bits (23 levels) with a ground resolution 4.78m×4.78m (tile) at the finest level, which is sufficiently accurate for precise location coordinates.

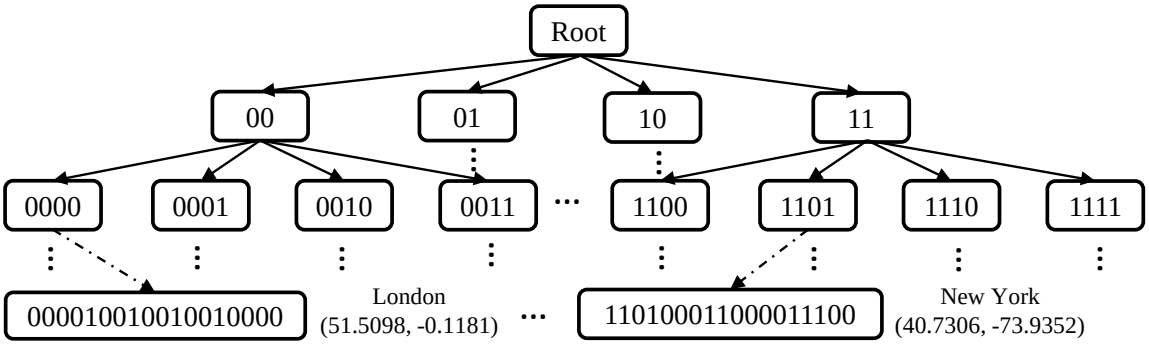


Figure 8.1. A Prefix-aware Tree for Location Data

Prefix Preserving Data Analysis for LBS. As discussed above, for the encoded bit strings of coordinates, utility can be fully preserved for data analysis since prefixes can be preserved in the encrypted locations (while preserving the privacy). For instance, "central park" and "the empire state building" in New York share a prefix, and the encrypted data for these two locations should also share the same length of prefix (e.g., might be two other places in London with the same proximity). Thus, the structure of the locations and the distance between such locations (besides other features such as frequency) can be preserved in the outsourced data. Besides the basic queries (e.g., counting), the location data analysis which uses distance-based metrics can achieve the same utility while analyzing the encrypted prefix preserving data, e.g., user mobility prediction [299].

Thus, the PrefixPE on the encoded locations can fully preserve the utility in the general data analyses for LBS. We have conducted experiments to validate such utility with prefix preserving encrypted location data in Section 8.7.3.

Table 8.1. Prefix-aware Encoding for Representative Data

Data Type	IP	Location	Market Basket	Numerical/Timestamps	Genome
Encoding	Default (IPv4/6)	4 Tiles (00, 01, 10, 11)	$\log(n)$-bit	Binary	AGCT
Pref. Utility	Fully	Fully	Fully	Partially (Proximity)	Partially (Prefix Analysis)

8.2.2.2 Market Basket Dataset [301]. This dataset includes the purchased items by the customers. With the item generalization hierarchy [301], we can encode the items into the bit strings with the same length. Similar to the location hierarchy (assigning 2 bits to each level since it is partitioned into 4 blocks), in each level of the item hierarchy, we assign $\log(n)$ bits to the n generalized items. The bit string of each item can be encoded by concatenating all the bits assigned in each level. The PrefixPE can fully preserve the item generalization path (e.g., the encrypted values of "apples" and "pears" also share a long prefix to make them close). With such prefix-aware encoding, data mining applications (e.g., frequent itemset mining [305]) can be performed on the prefix preserving encrypted data. Thus, PrefixPE on the encoded data/items can fully preserve the utility of the market basket data in the related data mining.

8.2.2.3 Numerical Data [302] and Timestamps [303]. Numerical data and timestamps are commonly included in a wide variety of datasets, e.g., network traffic [289], transaction data [306], and search logs [307]. Such values in different datasets can be simply converted into bit strings with meaningful prefix (aka. the converted binary numbers). Encrypting such bit strings with PrefixPE can also fully

preserve the prefixes in numerical values and day-time formats (e.g., "yyyy-mm-dd hh:mm:ss[.nnnnnnnnn]") for outsourcing. The proximity between any two values of these two data types can be preserved in the ciphtertexts (but not preserving the order of them). For instance, the ciphertexts of two close numerical values (or timestamps) also possess the same degree of proximity. Since the order of two numerical values and timestamps cannot be fully preserved, PrefixPE can preserve the partial utility in the encoded data of numerical and timestamp data.

8.2.2.4 Genome Dataset [308]. Genomic features (e.g., DNA sequences) of different organisms are studied to significantly advance the biological and medical research. For instance, DNA sequencing studies the order of adenine (A), guanine (G), cytosine (C), and thymine (T). Thus, each of A, G, C, T can be encoded using two bits, and the sequence can be concatenated with all the bit in order. In the encrypted DNA sequences for analysis, e.g., Private Edit Distance [300], PrefixPE on such encoded data can preserve the prefixes (order of nucleotide in the shared prefixes), but cannot preserve the full distance (e.g., Hamming distance) between two random sequences. Thus, PrefixPE using such simple encoding can partially preserve the utility on the genome data. We will also explore other encoding schemes in conjunction with PrefixPE to fully preserve the utility of the DNA sequences in the future.

8.2.2.5 Summary. Table 8.1 summarizes the prefix-aware encoding for some representative data and the utility preservation in the generic analyses. Note that the prefix-aware encoding can be tailored with different prefix definitions if necessary.

8.3 Security Models

8.3.1 Inference Attacks. The inference attacks on the property preserving encryption schemes [289, 292, 309–312] have been extensively studied recently. We

then introduce two typical inference attacks on the PrefixPE encrypted data, and briefly discuss other inference attacks [310–312].

Inference Attacks using Frequency and ℓ_p-Optimization. Frequency analysis [296] is the most well known inference attack with the auxiliary background knowledge. Extended from frequency analysis, ℓ_p-optimization [292] is further utilized to form a family of attacks that maximally infer the original data from property preserving encrypted data (e.g., order preserving encryption (OPE) and deterministic encryption (DTE)). Such attacks find the most matches from ciphertexts to plaintexts by minimizing the ℓ_p distance between the histograms of the encrypted dataset and an auxiliary dataset (as the background knowledge). The auxiliary dataset may be obtained by the adversary from some public statistics or prior versions of the original dataset. We explain such attacks on the encrypted location data as below:

Example 2 (Frequency and ℓ_p-Optimization based Inference Attacks [292, 296])**.** *As shown in Figure 8.2, the geo-locations can be encoded to bits (fully preserving the utility with the preserved prefixes, see Section 8.2.2) which are represented in hex format (for shortening the notations). The adversary may simply get some auxiliary information from publicly known data sources, e.g., top 50 most popular spots in New York. Then, the adversary can match the auxiliary location list (sorted by frequency) with the ciphertext of the locations also sorted by frequency [296].*

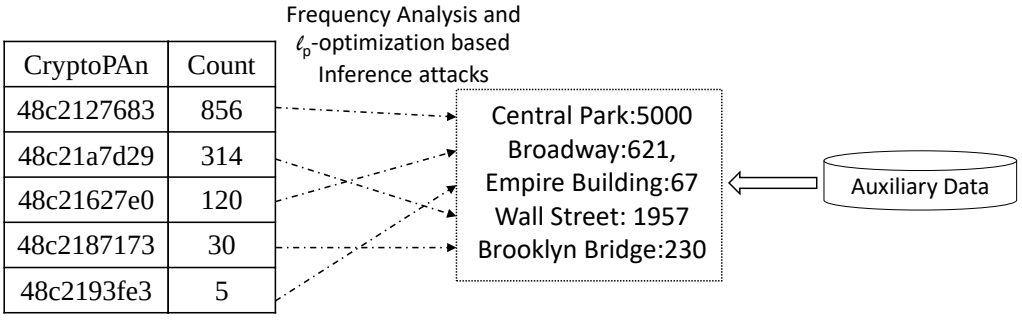


Figure 8.2. Frequency and ℓ_p-Optimization

Inference Attacks using Fingerprinting. Adversaries can perform strong inference attacks by injecting data into the original data collection and fingerprinting the records in the encrypted data [297, 309, 313]. Then the adversary could identify his/her own data from the encrypted data via the combination of other information, e.g., the timestamps and frequencies of the injected checked-in locations, the timestamps, port numbers and protocols of the network traffic data (with IP addresses). Then, the adversary can obtain a set of matches between the original data and the encrypted data, and eventually learn the prefixes of other values that share the prefix with the identified prefixes.

Example 3 (Fingerprinting based Inference Attacks [297, 298])**.** *Some detailed information of the same check-in location dataset in Example 2 are given in Figure 8.3. The leftmost table shows such real world dataset, the table in the middle gives the encoded bit strings for the GPS coordinates (see Section 8.2.2), and the rightmost table is the bit strings encrypted by CryptoPAn.*

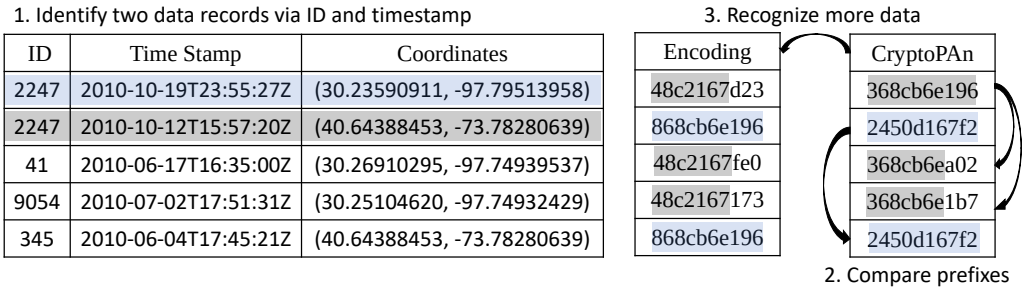


Figure 8.3. Fingerprinting based Inferences

The adversary can inject his/her tuples in such real data (e.g., using the application with its account). Then, he/she can identify two of his/her encrypted bit strings (w.r.t. his/her two location records) via other original information which are retained for utility (e.g., combination of timestamps and pseudonyms). Then, he/she can infer more prefixes and subprefixes by comparing the identified prefixes with other encrypted locations (bit strings). For instance, as "368cb6e196" and "2450d167f2" in

the first two rows are known (the adversary knows the location he/she has visited at that time), the location data of the first row ("368cb6e196") also shares 28*-bit prefix ("368cb6e", 7 digits in hex) with the location in the* 4*th row. Similarly, the* 6*th record can also be breached since the encrypted bit string is identical to "2450d167f2".*

Inference Attacks and Defense. Some other inference attacks [310–312] have been recently identified for OPE, such as exploiting the correlations among attributes [310] and learning query pattern access via statistical learning [312]. All these inference attacks (including the two detailed in the above examples) share some similarities, e.g., identifying the similar patterns from the original data and the encrypted data. We will propose a generalized multi-view outsourcing framework which effectively obfuscates such patterns (detailed in Section 8.4 and 8.5).

8.3.2 Threat Model and Privacy Notion. In our framework, the data owner outsources its encrypted data to the service provider, which is assumed to be *honest-but-curious*. The service provider has possessed background knowledge to implement a given set of inference attacks [289, 292, 297, 298] (*can be any emerging inference attacks on the property preserving encryption*). Moreover, the adversary is assumed to know the set of attributes in the outsourced data, and the domain for the attributes. We also assume that all the communications are in secure channel.

Then, we present the privacy notion in the threat model. Since the degree of privacy is quantified w.r.t. different inference attacks, we first formally give the following definition.

Definition 9 (Inference Attack Function)**.** *Function* $\mathcal{I}(Enc(\mathcal{D}), \{\alpha\})$ *is defined to quantify the leakage derived from performing a given set of inference attacks on encrypted data* $Enc(\mathcal{D})$ *with a set of background knowledge parameters* $\{\alpha\}$*.*

Example 4. *Assume that the adversary pose two kinds of inference attacks: (1)*

frequency and ℓ_p-optimization based inference attacks, and (2) fingerprinting based inference attacks, with two background knowledge parameters α_f, α_s, respectively:

- *the adversary holds a public auxiliary dataset including any α_f of the original data (percent between 0% and 100% of the domain) and their similar count distribution.*
- *the adversary has already identified any α_s of the original data (percent between 0% and 100% out of the domain) via injecting data before encryption.*

As a result, the attack function $\mathcal{I}(Enc(\mathcal{D}), \{\alpha_f, \alpha_s\})$ returns the information leakage, which is defined as below:

Definition 10 (Information Leakage [286])**.** *The percent of bits in $\mathcal{D}$ (encoded as bit strings) inferred from the encrypted data $Enc(\mathcal{D})$ (including the partially inferred prefixes).*

Then, we bound the leakage derived from the inference attacks with the following privacy notion.

Definition 11 (Γ-Leakage)**.** *While performing a given set of inference attacks on the encrypted data $Enc(\mathcal{D})$ (generated by encrypting the original data $\mathcal{D}$ using PrefixPE), the information leakage is upper bounded by $\Gamma \in [0, 1]$.*

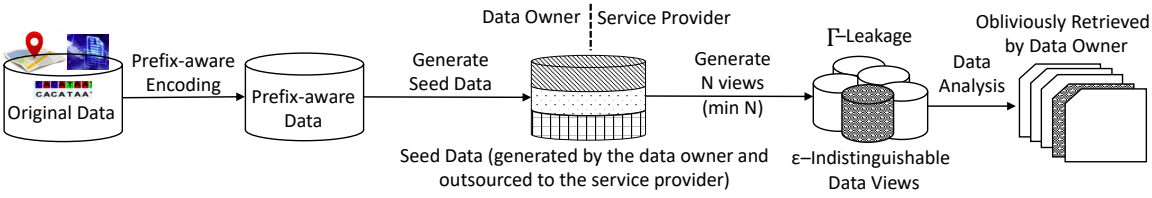


Figure 8.4. Generalized Outsourcing Framework on the Multi-view Approach [289]

8.4 System and Privacy Properties

In this section, we present the system and privacy properties. Table 8.2 summarizes the frequently used notations.

8.4.1 System Model. As illustrated in Figure 8.4, the proposed outsourcing framework involves two entities: (1) data owner: the party owns the original data and outsources the *prefix preserving encrypted data* to the service provider for data analysis, and (2) service provider: a cloud platform or an external company who provides data analysis services, which might intend to infer the original data from the outsourced encrypted data.

Specifically, Mohammady et al. [289] proposed a multi-view approach to defend against inference attacks on the PrefixPE encrypted data. The core idea is to hide the real data view among other fake data views, where the service provider cannot identify the real data view. However, such approach is only limited to network traffic data. We make the following key improvements on the multi-view outsourcing:

- generalize the outsourcing framework to privately outsource a wide variety of data types (any length of the data vs. 32/128-bit IP addresses).
- generalize the defense (bounded leakage) against any given set of inference attacks (to make the illustration more concrete, we conducted experiments on more inference attacks, compared to [289]).
- improve the privacy guarantee (by specifying a privacy bound Γ-Leakage) and computational overheads on analyzing the outsourced data (minimizing the number of views while satisfying Γ-Leakage against a given set of inference attacks).

As shown in Figure 2.1, the generalized multi-view outsourcing framework is detailed as follows:

Table 8.2. Notations for Chapter 6

Symbol	Definition
$\mathcal{D}$	original data
A	number of distinct values in $\mathcal{D}$
$\mathcal{D}_p$	prefix-aware data
$\mathcal{D}_0$	initially encrypted data
$\mathcal{D}_s$	seed data
$\mathcal{D}_1 \dots \mathcal{D}_N$	N generated data views
α_f	background knowledge parameter on frequency and ℓ_p-optimization based inference attacks
α_s	background knowledge parameter on fingerprinting based inference attacks
$f^r(\cdot)$	executing r times CryptoPAn
$\mathcal{I}(\cdot)$	inference attack function: returns the leakage derived from the encrypted data in the attacks
$p(x)$	number of partitions with x-bit shared prefix
$\mathbb{R}$	a pseudorandom matrix
$G_1 \dots G_N$	vectors for generating data views
K_0	non-shared key for CryptoPAn
K_1	outsourced key for CryptoPAn
P_i	data partition i

At the Data Owner.

(1) **Prefix-aware Encoding:** encodes the sensitive attributes of the original data (e.g., locations, IP addresses, set of items) to prefix-aware data $\mathcal{D}_p$ with prefix-aware encoding, as discussed in Section 8.2.2.

(2) **Partitioning and Generating Seed Data:** initially encrypts the encoded data $\mathcal{D}_p$ to $\mathcal{D}_0$ using CryptoPAn with a non-shared CryptoPAn key K_0 (for preventing the brute force attack), and then partitions $\mathcal{D}_0$, as well as generates a seed data $\mathcal{D}_s$ by executing CryptoPAn in each partitions with a random number of iterations using another key K_1. This step obfuscates the data in the seed data $\mathcal{D}_s$, which is safe to share and can be used to generate multiple data views by the service provider (note that K_1 is shared to the service provider for generating the data views). See details in Section 8.5.1.

(3) **Outsourcing:** the Seed Data $\mathcal{D}_s$, the key K_1 and some other required parameters for generating multiple data views will be outsourced to the service provider.

At the Service Provider.

(4) **Generating N Data Views:** generates N data views using $\mathcal{D}_s$, the CryptoPAn key K_1 and other parameters, where *exactly one* out of N data views is the real data view (which fully preserves the prefixes).

(5) **Analyzing N Data Views:** performs the required data analysis on the N data views (note that N is minimized for satisfying Γ-Leakage). The real data views fully preserves the prefixes with the 100% analysis accuracy.

The details are given in Section 8.5.2.

Both Data Owner and Service Provider.

(6) **Obliviously Retrieving Analysis Result:** the data owner leverages an oblivious random access memory (ORAM) protocol [314] to retrieve the correct analysis result out of N results while the service provider does not know which result is retrieved by the data owner.

8.4.2 Privacy Guarantees. We now summarize two desired privacy properties against the inference attacks in semi-honest model.

8.4.2.1 Indistinguishability. The proposed framework first ensures that N data views are *indistinguishable* (inspired from differential privacy [315]):

Definition 12. *[289] The generated N data views are ϵ-indistinguishable against inference attacks if and only if*

$$\exists\ \epsilon \geq 0,\ \ s.t.\ \forall i, j \in \{1, 2, \cdots, N\} \Rightarrow$$
$$e^{-\epsilon} \leq \frac{Pr[data\ view\ i\ may\ be\ real\ ]}{Pr[data\ view\ j\ may\ be\ real]} \leq e^{\epsilon} \tag{8.3}$$

As ϵ is smaller, the N different data views would be more indistinguishable. As a result, a smaller number of fake data views will be filtered out by the adversary. For instance, $e^{-\epsilon} = 0.9$ means that at least 90% of the fake data views are indistinguishable from the real one. We give the indistinguishability analysis in Section 8.5.2.4 and experimentally validate that ϵ is quite small in Section 8.7.2.

8.4.2.2 Bounded Leakage against Inference Attacks. Our framework can ensure that all the encrypted data held by the adversary satisfy Γ-Leakage (Definition 11) as an additional layer of protection:

$$\mathcal{I}(\mathcal{D}_s, \mathcal{D}_1, \ldots, \mathcal{D}_N, \{\alpha\}) \leq \Gamma \tag{8.4}$$

where the adversary performs the inference attacks with background knowledge

$\{\alpha\}$ on the seed data $\mathcal{D}_s$ (received by the service provider) and N different data views $\mathcal{D}_1, \ldots, \mathcal{D}_N$ (generated by itself). In the outsourcing framework, the information leakage is bounded by a very small Γ, as experimentally validated in Section 8.7.

Remark. The first privacy notion ϵ-indistinguishability is defined to measure how indistinguishable the generated N data views can be (hiding the real data view). Indeed, all the fake data views and one real data view could possibly leak information (though they are indistinguishable) if the adversary is armed with background knowledge to perform inference attacks. Thus, another privacy notion Γ-Leakage is defined to bound the overall information leakage in all the data obtained and generated by the adversary. Two privacy notions measure different aspects of the privacy and complement each other in the generalized multi-view outsourcing framework.

8.5 Generalized Framework Design

8.5.1 Prefix-based Partition. The initially encrypted data $\mathcal{D}_0$ is partitioned by assigning all the values sharing at least x-bit prefix into the same partition. Specifically, given $\mathcal{D}_0$, the data (e.g., location) has been encoded into L bits. We first traverse $\mathcal{D}_0$ to get the set of distinct values. Given the prefix length $x \in [1, L]$, we generate a mapping set by grouping all the values sharing length-x prefix to one subset. Finally, we obtain the partitions in $\mathcal{D}_0$ using the mapping set. Denote the number of partitions in $\mathcal{D}_0$ created with length-x prefix as $p(x)$. Thus, we have partitions P_i with length-x prefix $d_i, i \in [1, p(x)]$.

This prefix-based partitioning scheme can potentially result in the identification of the real data view by the adversary due to the collision property of CryptoPAn [286]. In [289], the identification of the real data view was proposed by the collision of the encrypted full IP addresses. Indeed, there also exists subprefix collision of the prefix which can possibly help identify the real data view. We generalize such attack and

name it as *Subprefix Collision Attack*, which is caused by similar prefixes or subprefixes ("close prefix").

Closeness of Prefixes. The data in the same partition share a common prefix (length-x), the prefixes of two different partitions may also share a length-y subprefix where $y < x$. We use the following measure to define such relationship between such two partitions (with *close* prefix).

Definition 13 (β-closeness)**.** *While partitioning $\mathcal{D}_0$ using the shared length-x prefix, given two prefixes d_i and d_j where $i, j \in [1, p(x)]$, if d_i and d_j also share a length-y subprefix ($y < x$) such that $|x - y| \leq \beta$, the two partitions P_i and P_j are said to satisfy β-closeness (or P_i and P_j are β-close).*

As mentioned before, applying CryptoPAn in real data view would also preserve such *closeness* relationship across different partitions in $\mathcal{D}_0$, which may cause *subprefix collision attack*: the real data view will preserve all the prefixes (including subprefixes) among all the partitions whereas the fake data views would not retain them (see Example 5).

Example 5 (Subprefix Collision Attack)**.** *As shown in Figure 8.5, the original data are encoded into prefix-aware bit strings and represented in hexadecimal (for simplicity of notations). The prefix length of the partition is $x = 20$ bits (dash line) in binary (5-digit in hexadecimal). The original data is divided into two partitions, P_1 with prefix 38456 and P_2 with 38457, both of which share a common subprefix of length 19; only the the least significant bit (LSB) is different (i.e., 0 and 1) – these two partitions satisfy 1-closeness. The real data view can be readily distinguished from other data views with the prefix 27c27 and 27c26, which still keep the subprefixes across partitions. Considering that CryptoPAn has collision-resistant property [286], the probability that every fake data view generates a common subprefix in different partitions (by executing*

random number of iterations CryptoPAn) is negligible. Then, the adversary can directly identify the real data view.

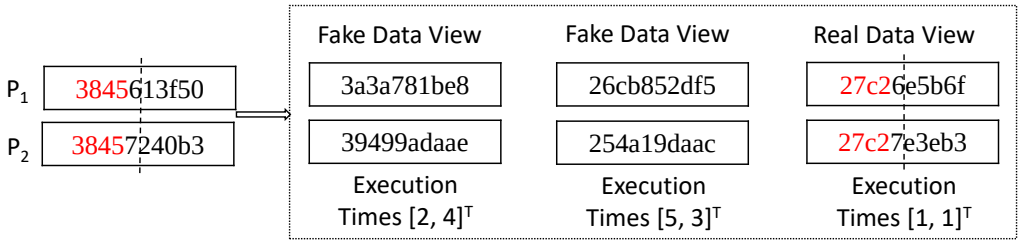


Figure 8.5. Subprefix Collision Attack

Notice that, β-closeness has the following characteristics for this subprefix collision attack:

1. for a small β (different partitions share a longer subprefix), if many β-close partitions are identified, then the real data view can be simply identified;
2. for a large β (short subprefix; an extreme case, many partitions share the first bit), then the β-close partitions would not increase the confidence of such attack.

To tackle such attacks, the proposed scheme aims to create more similar collisions among β-close partitions while generating data views, thus the adversary cannot identify the real view from the subprefix collision (see Section 8.5.2).

8.5.2 Multiple Data Views Generation.

8.5.2.1 Seed Data for Generating Views. The objective of partitioning $\mathcal{D}_0$ is to *obfuscate* the encrypted data across different partitions in the outsourced data but still be able to reconstruct the encrypted data with fully prefix preservation ($\mathcal{D}_0$ or similar data) for analysis. Thus, the next step is to generate the "Seed Data" which is safe to outsource (obfuscated) and oblivious to reconstruct a real data (fully prefix preserved).

Specifically, the framework applies CryptoPAn $f(\cdot)$ to different partitions in $\mathcal{D}_0$ with random number of iterations to generate the seed data $\mathcal{D}_s$. Given shared prefix length x, there are $p(x)$ partitions, $\mathcal{D}_0 = \{P_1, P_2, ..., P_{p(x)}\}$. Then, we define a random vector $G_0 = [v_1^0, v_2^0, ..., v_{p(x)}^0]^T$, where entry $v_i^0, i \in [1, p(x)]$ is an integer representing the number of iterations applying CryptoPAn on the partition P_i. Thus, we have $f(\mathcal{D}_0, G_0, K_1) = [f^{v_1^0}(P_1), f^{v_2^0}(P_2), \ldots, f^{v_{p(x)}^0}(P_{p(x)})]$, where K_1 is the CryptoPAn key for obfuscating the data across $p(x)$ partitions in the seed data $\mathcal{D}_s$ (the same key will be used for reconstructing prefix preserving data). Note that $v_i^0, i \in [1, p(x)]$ can be chosen from the domain $[-h, h]$ where h is comparable to $p(x)$.

8.5.2.2 N **Data Views.** As discussed in Section 8.4.1, generating multiple data views (say N) which include only one real data view (fully prefix preserving) and $N-1$ fake data views (non prefix preserving across partitions) could mitigate the leakage against inference attacks (*since the probabilities of matching the encrypted data to true values can be greatly reduced by providing more data views*). This is experimentally validated in Section 8.7.

To this end, the service provider generates N different data views based on the seed data and N pseudorandom vectors (similar to the procedure of generating the seed data $\mathcal{D}_s$). The data owner generates N pseudorandom vectors which form a matrix $\mathbb{R} = [G_1, \ldots, G_N]$, and then outsources the seed data $\mathcal{D}_s$, pseudorandom matrix $\mathbb{R}$, and the CryptoPAn key K_1 used for generating N views.

Definition 14 (data view)**.** *Given pseudorandom vectors* $G_i = [v_1^i, v_2^i, \ldots, v_{p(x)}^i], i \in [1, N]$ *in the matrix* $\mathbb{R}$*, the* i*th data view can be represented as:*

$$\mathcal{D}_i = [f^{\sum_{j=0}^{i} v_1^j}(P_1), f^{\sum_{j=0}^{i} v_2^j}(P_2), \ldots, f^{\sum_{j=0}^{i} v_{p(x)}^j}(P_{p(x)})]$$

With the associative and inverse property of CryptoPAn encryption $f(\cdot)$, the

ith data view can be the prefix preserving if the entries in the pseudorandom vectors satisfy:

$$\sum_{j=0}^{i} v_1^j = \sum_{j=0}^{i} v_2^j = \cdots = \sum_{j=0}^{i} v_{p(x)}^j \tag{8.5}$$

In other words, if the aggregated pseudorandom vectors for the ith data view $G_0+G_1+\cdots+G_i$ have equivalent entries, the ith data view can be prefix preserving (real data view); otherwise, not prefix preserving (fake data view). If $G_0+G_1+\cdots+G_i = 0$, then the real data view would be the initially encrypted data $\mathcal{D}_0$. Note that only one prefix preserving data is necessarily generated out of N data views (for reducing the probability of identifying it and the leakage). Since the data owner generates all the pseudorandom vectors $G_0, G_1, \ldots, G_N$ and the first vector G_0 (for generating the seed data) is not shared to the service provider, the service provider would not know when Equation 8.5 holds for generating the real data view.

Mitigate Subprefix Collision Attacks. As discussed before, our multi-view outsourcing creates more collisions among these prefixes which have common subprefixes (in β-close partitions) to address the subprefix collision attack. Specifically, when generating the pseudorandom number of executions of CryptoPAn on the partitions, we can generate the same execution times (aggregated pseudorandom) for the β-close partitions in the fake data views while generating different aggregated pseudorandom numbers for partitions with different subprefixes or β is too large (collisions may naturally occur in this case). Thus, the random matrix $\mathbb{R}$ $(G_1, \ldots, G_N)$ to determine the execution times for each data view is generated as below:

1. the data owner first generates a random vector G_0 with the size $p(x)$. Then the data owner will determine the minimum N as illustrated in Section 8.5.2.3.

2. the data owner then generates N pseudorandom vectors (as $p(x) \times N$ matrix). $r \in [1, N]$ is the randomly generated index for the real data view (*only data*

owner knows), and the generated pseudorandom vector $G_r, r \in [1, N]$ satisfies Equation 8.5 (e.g., $\sum_{i=0}^{r} G_i = [0, 0, ..., 0]^T$) to preserve all the original prefixes.

3. finally, the data owner generates a set of $N - 1$ pseudorandom vectors for the fake data views. Recall that we expect to create as much collisions among the partitions as possible (for also satisfying β-closeness in the fake data views). Each aggregated vector of G_0 and $G_i, i \in [1, N]$ (e.g., $G_0 + G_1, G_0 + G_1 + G_2, \ldots, G_0 + G_1 + \cdots + G_N$) should have at least two equal execution times for each pair of β-close partitions.

Input : x, N, β, G_0

Output : pseudo random matrix $\mathbb{R} = [G_1, \ldots, G_N]$

1 Generate the set of prefixes with length-x

2 Group the β-close partitions (a reasonable β)

3 **for** $i = 1 : n$ **do**

4 **if** $i \neq r$ **then**

5 generate a length-$p(x)$ pseudorandom vector G_i such that the vector $\sum_{j=0}^{i} G_i$ includes two identical values if the corresponding two partitions are β-close

6 **if** $i = r$ **then**

7 generate a length-$p(x)$ pseudorandom vector G_i: the entries in $\sum_{j=0}^{i} G_i$ are identical

Algorithm 17: Pseudorandom Matrix Generation

Algorithm 17 gives the details for generating such pseudorandom vectors/matrix. Example 6 illustrates how such pseudorandom matrix can address the subprefix collision attacks and hide the real data view.

Example 6. *Figure 8.6 shows 5 partitions with* 20*-bit prefixes. The initial random vector* $G_0 = [-2, 2, 3, 0, -1]^T$*, and the pseudorandom vectors* $G_1 = [3, 0, -2, 2, 3]^T, G_2 =$

$[2,1,2,1,1]^T$, *and* $G_3 = [1,1,-1,-2,-2]^T$ *are generated by the data owner.* $\mathcal{D}_2$ *is the real data view while* $\mathcal{D}_1$ *and* $\mathcal{D}_3$ *are the fake data view.* G_0 *will be held privately, only the pseudorandom matrix* $\mathbb{R} = [G_1, G_2, G_3]$ *are outsourced. Thus, we have:*

- *generating* $\mathcal{D}_1$ *(fake) executes CryptoPAn for* $G_0 + G_1 = [1,2,1,2,2]^T$ *times in 5 partitions, resp. (*$G_1 = [3,0,-2,2,3]^T$ *times by the service provider).*
- *generating* $\mathcal{D}_2$ *(real) executes CryptoPAn for* $G_0 + G_1 + G_2 = [3,3,3,3,3]^T$ *times in 5 partitions, resp. (*$G_1 + G_2 = [5,1,0,3,4]^T$ *times by the service provider).*
- *generating* $\mathcal{D}_3$ *(fake) executes CryptoPAn for* $G_0 + G_1 + G_2 + G_3 = [4,4,2,1,1]^T$ *times in 5 partitions, resp. (*$G_1 + G_2 + G_3 = [6,2,-1,1,2]^T$ *times by the service provider).*

Note that G_0 *for 5 partitions are locally executed to generate the seed data* $\mathcal{D}_s$ *by the data owner, and the service provider cannot reconstruct* G_0 *from the received data.*

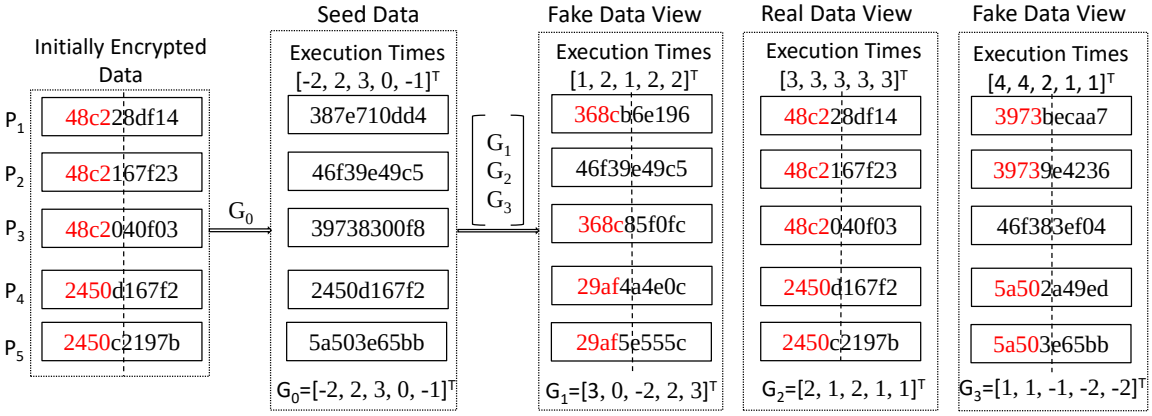


Figure 8.6. Utilizing Pseudorandom Matrix to Generate 3 Data Views (negative execution times refer to repeating inverse CryptoPAn execution)

Example 7. *In Fig 8.6, the seed data generates* 3 *data views. The two fake data views have at least* 2 *subprefix collisions, e.g.,* 368*cb and* 368*c*8 *in fake data view* $\mathcal{D}_1$

and 5a502 *and* 5a503 *in fake data view* $\mathcal{D}_3$*. Thus,* $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ *are subjected to some indistinguishablity (which will be formally analyzed below).*

8.5.2.3 Minimum N with Bounding Γ-Leakage. As shown in Equation 8.4, the leakage drawn from inference attacks is determined by background knowledge parameters $\{\alpha_s, \alpha_f\}$, the received seed data $\mathcal{D}_s$ and generated data views $\mathcal{D}_1, \ldots, \mathcal{D}_N$. As N grows, leakage can be smaller, but more data views should be generated for the same data analysis (more computation). Thus, our multi-view outsourcing seeks a minimum N while satisfying Γ-leakage.

More specifically, the prefix length x determines the partitions $P_1, \ldots, P_{p(x)}$. Then, given any $x \in [1, L]$, there exists a minimum N while bounding the leakage with Γ in our experiments (fixing $\{\alpha_f, \alpha_s\}$ for the background knowledge). As a result, before partitioning the data, the data owner can find an x such that the required minimum N for Γ-leakage is minimized – searching the x and minimum N takes $O(n \log(n))$ since the leakage derived from the fixed inference attacks is anti-monotonic on N.

8.5.2.4 Privacy Analysis. In practice, an adversary will exploit any related information (received data, background knowledge, etc.) to identify whether a data view is the real or fake one. Recall that only the sensitive attributes (prefix-aware encoded) are encrypted with CryptoPAn while other attributes are identical among all the data views. Thus, identifying the real data view only depends on the encrypted partitions, such as comparing N different data views and the leakage derived from them via inference attacks.

Theorem 10. *The generated* N *data views satisfy* ϵ*-indistinguishability where*

$$\epsilon = \ln[\frac{\sum_{k=1}^{b}(\prod_{k=1}^{p(x)\alpha_s}|P_k|)(p(x)\alpha_s)!}{\prod_{j=0}^{A\alpha_f+p(x)\alpha_s-1}(A-j)}], \tag{8.6}$$

$b = \frac{(p(x)-p_f)!}{(p(x)\alpha_s)!}$, $k \in [1, p(x)]$ *and* A *is the number of distinct values in* $\mathcal{D}$.

Proof. The adversary is armed with α_f knowledge of the original data for the frequency and ℓ_p-optimization attacks, and the α_s knowledge for the fingerprinting attacks. Then, we derive the indistinguishability bound ϵ as follows.

Recall that we generate $p(x)$ partitions in $\mathcal{D}_0$ based on the prefix length x. Denote the cardinality of each partition P_i as $|P_i|, i \in [1, p(x)]$. Then, the total number of possibilities (denoted as O) of dividing A distinct values into the $p(x)$ partitions is:

$$O = \frac{A!}{|P_1|!|P_2|! \cdots |P_{p(x)}|!} \tag{8.7}$$

Next, all the possible outcomes of the real data views for the adversary (denoted as T) depends on it's armed background knowledge (α_s, α_f), we thus need to consider the following two aspects: (1) the adversary can reconstruct $A * \alpha_f$ out of A distinct values by ℓ_p-optimization based inference attacks while these compromised $A * \alpha_f$ distinct values are possibly across $p_f \in [1, A * \alpha_f]$ partitions, which will eleminate a number of partitions for at most $p(x) - p_f$; (2) the adversary matches the remaining partitions with the $p(x)\alpha_s$ inferred prefixes via the fingerprinting based inference attacks. Thus, we have the following equation:

$$T = \frac{\sum_{k=1}^{b} (\prod_{k=1}^{p(x)\alpha_s} |P_k|)(A * (1-\alpha_f) - p(x)\alpha_s)!(p(x)\alpha_s)!}{|P_1|!|P_2|! \cdots |P_{p(x)}|!} \tag{8.8}$$

where $b = \frac{(p(x)-p_f)!}{(p(x)\alpha_s)!}$, $k \in [1, p(x)]$.

Finally, we thus have:

$$\forall i,j \in \{1,2,\cdots,N\}, \frac{Pr[\text{data view } i \text{ may be real }]}{Pr[\text{data view } j \text{ may be real}]} = \frac{T}{O} = \frac{\sum_{k=1}^{b}(\prod_{k=1}^{p(x)\alpha_s}|P_k|)(p(x)\alpha_s)!}{\prod_{j=0}^{A\alpha_f+p(x)\alpha_s-1}(A-j)} \tag{8.9}$$

where $b = \frac{(p(x)-p_f)!}{(p(x)\alpha_s)!}$, $k \in [1, p(x)]$. Per Definition 12, we can complete the proof. □

In Section 8.7.2, the experiments on the real datasets show $\epsilon \leq 1.5$ in general. Furthermore, the overall information leakage is also upper bounded as below.

Theorem 11. *Given the attacker with background knowledge $\{\alpha_f, \alpha_s\}$, the information leakage from the seed data and N data views ($\mathbb{D} = \{\mathcal{D}_s, \mathcal{D}_1, \ldots, \mathcal{D}_N\}$) satisfies the Γ-leakage where*

$$\mathcal{I}(\mathbb{D}, \{\alpha_f, \alpha_s\}) \leq \Gamma = \frac{\alpha_f}{N} + \frac{x(\sum_{\forall P_k \in C}|P_k| - A \cdot \alpha_f)}{A \cdot L \cdot N} \tag{8.10}$$

where A is the number of distinct values in $\mathcal{D}$, L is the length of the encoded bit strings, x is the prefix length used for partitioning, C is the union of the two sets of data partitions derived by the inference attacks with background knowledge $\{\alpha_f, \alpha_s\}$, and $\forall P_k \in C, |P_k|$ is the number of distinct values in partition P_k.

Proof. The adversary is armed with α_f knowledge of the original data for the frequency and ℓ_p-optimization attacks, and the α_s knowledge for the fingerprinting attacks. According to Definition 11, to derive the upper bound Γ of information leakage, we consider the worst case scenario by assuming that the unique data or prefixes inferred by two types of inference attacks are disjoint (to derive the highest leakage). Note that we also assume that the adversary attacks all the data $\mathbb{D}$ (seed data and N data views)

and the adversary does not know which data view is the real one (indistinguishability). Then, we compute the leakage on the two types of inference attacks, respectively.

As depicted before, the adversary can reconstruct $A \cdot \alpha_f$ out of A distinct values by ℓ_p-optimization based inference attacks while the adversary can matches $p(x) \cdot \alpha_s$ inferred prefixes by the fingerprinting based inference attacks. Then, we get the bits of information leakage (as percents) by ℓ_p-optimization based inference attacks: $A\alpha_f \cdot L + x(\sum_{\forall P_k} |P_k| - A \cdot \alpha_f)$, where $\forall k \in [1, |A\alpha_f|], |P_k|$ are the number of distinct values across all the $A \cdot \alpha_f$ partitions. Similarly, we can compute the bits of information leakage (as percents) by the fingerprinting based inference attacks $\sum |P_j| \cdot x$, where $\forall j \in [1, |p(x)\alpha_s|], |P_j|$ are the number of distinct values across all the $p(x)\alpha_s$ partitions. To sum up, we can get the leakage (the percent of bits inferred by the adversary, Definition 10) as below:

$$\frac{A\alpha_f \cdot L + (\Sigma|P_k| - A\alpha_f) \cdot x + \Sigma|P_j| \cdot x}{N \cdot A \cdot L} \tag{8.11}$$

Thus, the overall leakage is upper bounded by

$$\Gamma = \frac{\alpha_f}{N} + \frac{x(\sum_{\forall P_k \in C} |P_k| - A \cdot \alpha_f)}{A \cdot L \cdot N} \tag{8.12}$$

where C is the union of the two sets of data partitions inferred by the two types of inference attacks with background knowledge $\{\alpha_f, \alpha_s\}$. This completes the proof. □

We also demonstrate the defense performance (on bounded leakage) of our proposed framework with the given inference attacks in Section 8.7.1.2. To sum up, our framework can ensure any bounded leakage against any given set of inference attacks, whereas the existing multi-view approach [289] cannot strictly bound it.

8.5.3 Privately Retrieving Analysis Result. In Step (7), the service provider performs the same analysis on all the N data views to derive N analysis results. Then, in Step (8), the data owner can privately retrieve the analysis result of the real data view ($\mathcal{D}_r$) via the oblivious random access memory (ORAM) [314] without letting the service provider know which analysis result has been retrieved.

Proposition 2. *The generalized outsourcing framework (with the prefix-aware encoding) ensures 100% accuracy for analyzing the prefix preserving encrypted data.*

Proof. Equation 8.5 ensures that exactly one real data view with fully prefix preserving encrypted data will be generated out of N data views. The accuracy for analyzing such real data view is 100%. Since the data owner knows the end-to-end data encryption (with two CryptoPAn keys K_0 and K_1 for multiple rounds of prefix preserving encryption), it knows the index for the real data view with its locally generated pseudorandom matrix $\mathbb{R} = [G_1, \ldots, G_N]$.

Thus, the data owner can privately retrieve the analysis result of the real data view, which ensures 100% utility on the fully preserved prefixes (validated in Section 8.7.3). □

8.6 Discussion

Worst Case Leakage and Amplification Effect. We bound the worst leakage for all the attacks with Γ, e.g., the maximum leakage resulted from different combinations of background knowledge in different inference attacks. Moreover, amplification effect of different inference attacks are also considered in the experiment. For instance, as fingerprinting based inference attacks have recovered some encrypted data with background knowledge α_s, then the accuracy of frequency and ℓ_p-optimization based inference attacks can be improved. Thus, the leakage bounded by Γ is derived from multiple attacks with the amplification effect.

New Inference Attacks. Our generalized outsourcing defines Γ to bound the leakage from any combinations of the inference attacks. In case of other threat models (e.g., other inference attacks [313,316], or newly identified attacks [310–312], the data owner only needs to simulate the inference attacks to estimate the leakage and specify the x which results in the minimum N to bound the leakage.

Communication Overheads. Although the framework generates N (could be hundreds) data views to ensure privacy, the data owner only sends one seed data $\mathcal{D}_s$ *with the same size as the original data*, some pseudorandom vectors (matrix) $\mathbb{R}$ and the CryptoPAn key K_1 to the service provider. Moreover, the data owner privately retrieves the corresponding analysis result (with a small size in general) via ORAM. Thus, the total communication overheads are quite close to that of a regular data outsourcing.

8.7 Experimental Evaluations

We implemented our outsourcing framework on the CloudLab platform [317] where one server works as the client and another as the service provider. We utilize two different real world datasets in the experiments.

Traveler Check-in Location Data. It includes 6,442,890 check-ins records of 196,591 users on a social network (http://snap.stanford.edu/data/loc-gowalla.html). We integrated the data into 633,743 distinct locations in total. A single data record consists of the user IDs, timestamps, locations (GPS coordinates) and location IDs.

Network Traffic Data (https://www.unb.ca/cic/datasets/dos-dataset.html). It is collected from DoS attacks. We extracted the source/destination IPs, timestamps, packet types, and port numbers from a 4.8GB raw dataset. 104,820 records are attributed to 778 distinct source IPs.

We encode the traveler check-in location data (i.e., GPS locations) into bit strings with prefix-aware encoding. For the network traffic data, IP addresses can also be binarized. Then, CryptoPAn can be applied to preserve prefixes in the encrypted bit strings for both datasets.

8.7.1 Experiments on the Inference Attacks. We have implemented two common inference attacks on the encrypted data: 1) Frequency and ℓ_p-optimization ($p = 2$) based inference attacks [292]; 2) Fingerprinting based inference attacks [289,297] and set the background knowledge parameters as α_f and α_s. While attacking two encrypted datasets, leakage [286] out of the original data (Definition 11) is adopted as the metric to evaluate the confidence of the attacks.

To model the background knowledge of the adversary in the frequency or ℓ_p-optimization based inference attacks, we setup the corresponding auxiliary dataset (including α_f of the original locations/IP addresses' similar frequencies). In addition, any α_s of the original locations/IP addresses are assumed to be identified by the adversary via fingerprinting. We repeated each attacking experiment for 100 times and average the results as the leakage. The average runtimes of different inference attacks on two datasets are shown in Table 8.3 (all the attacks can be efficiently performed by the service provider and simulated by the data owner).

Table 8.3. Average Runtime of Attacks (sec)

Data Attacks	Frequency	ℓ_2-opt	ℓ_3-opt	Fingerprinting
Location Data	2.64	15.19	18.73	5.24
Network Data	0.12	3.17	4.38	1.23

8.7.1.1 Attacking CryptoPAn. We first implement the attacks on the two datasets encrypted by CryptoPAn (keys are randomly generated).

1. fixing the fingerprinting-based background knowledge $\alpha_s = 10\%, 50\%$, and measure the leakage via varying the other background knowledge $\alpha_f \in [10\%, 90\%]$ (*from weak to very strong background knowledge*);

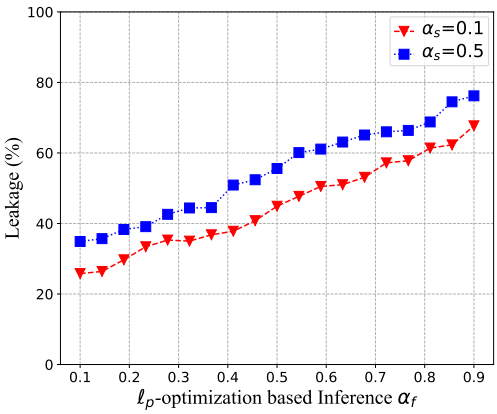
2. fixing the background knowledge for frequency and ℓ_p-optimization based inference attacks $\alpha_f = 50\%, 90\%$ and varying the fingerprinting inference $\alpha_s \in [10\%, 50\%]$ (*data injection does not exceed* 50% *in general*).

In Figure 8.7(a) and 8.7(c), the leakage grows from 40% to 80% of the original locations/IP addresses as α_f increases from 10% to 90% (changing $\alpha_s = 10\%$ to 50% does not increase the leakage much, compared to α_f). In Figure 8.7(b) and 8.7(d), we learn a similar trend for both datasets. These empirical results demonstrate that encrypted locations/IP addresses (by CryptoPAn) are very vulnerable to both ℓ_p-optimization and fingerprinting based inference attacks.

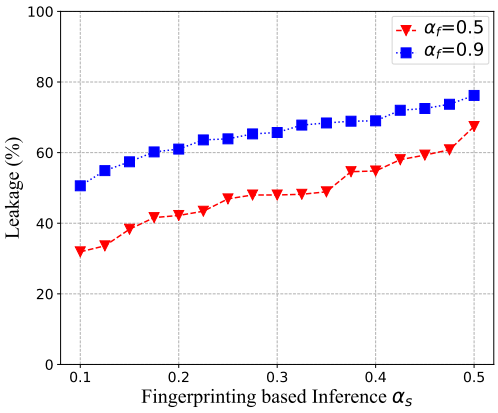
8.7.1.2 Attacking Multi-view Outsourcing.

Intuitively, the more N data views are generated, the less the leakage will be derived from the inference attacks since the adversary cannot distinguish them.

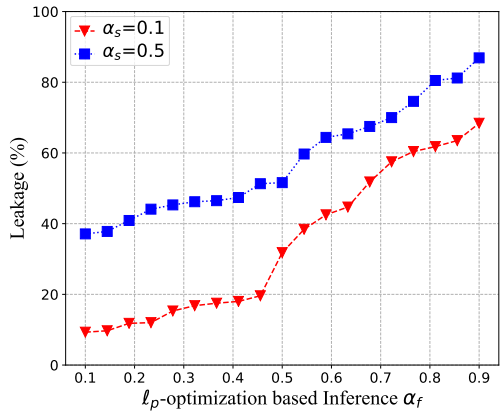
To validate this, we consider the worst case scenario. Given the indistinguishability bound ϵ, the real data view is ϵ-distinguishable with other generated $N-1$ fake data views. We also assume that the adversary attacks all the data views. Denote the probability of identifying any data view as the real data view as Pr_r and the leakage as γ_r. Also, denote the probability of identifying any data view as a fake data view as Pr_i, $i \in [1, N], i \neq r$ and the leakage as γ_i (γ_r might be larger while γ_i might be smaller since the data is fake). Then, the leakage in the worst case can be obtained as $\gamma_{total} = \sum_{i=1, i\neq r}^{N} \gamma_i * Pr_i + \gamma_r * Pr_r$, where $Pr_i = \frac{1}{N-1+e^\epsilon}$, $Pr_r = \frac{e^\epsilon}{N-1+e^\epsilon}$ in the worst case (since $\forall i \in [1, N], i \neq r, \frac{Pr_r}{Pr_i} \leq e^\epsilon$).



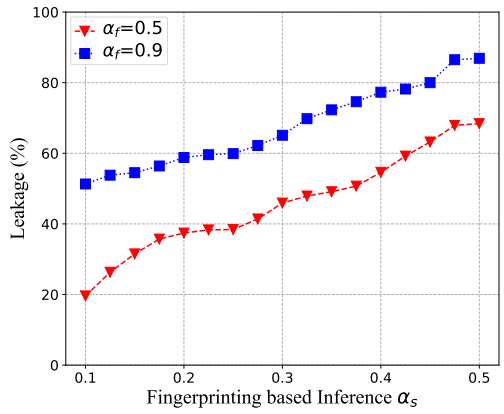
(a) Encrypted Location Data



(b) Encrypted Location Data



(c) Encrypted Network Data



(d) Encrypted Network Data

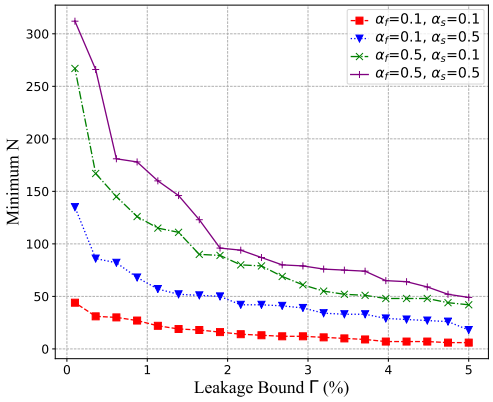
Figure 8.7. Inference Attacks on Data Encrypted by CryptoPAn

We now examine the bounded leakage of multi-view outsourcing against the same inference attacks. Figure 8.8 presents the required minimum number of data views N on the encrypted location and network traffic data, respectively. Specifically, if the leakage bound Γ increases (from 0.1% to 5%), the required minimum number N declines from ~ 300 to ~ 50 (against strong attackers $\alpha_f = 50\%$ and $\alpha_s = 50\%$), and declines from ~ 50 to ~ 5 (against weak attackers $\alpha_f = 10\%$ and $\alpha_s = 10\%$). While increasing the background knowledge α_f from 10% to 90%, the required minimum N increases for all the leakage bound and α_s (Figure 8.8(c)). Similarly, while increasing

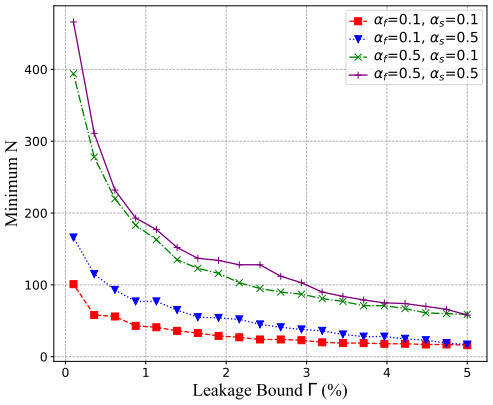
the background knowledge of fingerprinting from 10% to 50%, the required minimum N also increases for all the leakage bound and α_f in the multi-view outsourcing (see Figure 8.8(e)). Figure 8.8 (b,c,d) shows a similar trend on network traffic data. Table 8.4 shows the optimal x for different leakage bound $\Gamma \in [0.1\%, 5\%]$ on the location data, and different background knowledge of two types of inference attacks (α_f, α_s). Most x values are greater than 20 (out of 46). Such long prefixes in the optimal case (minimum N) would generate more partitions.

Table 8.4. Optimal x for Encrypting Locations

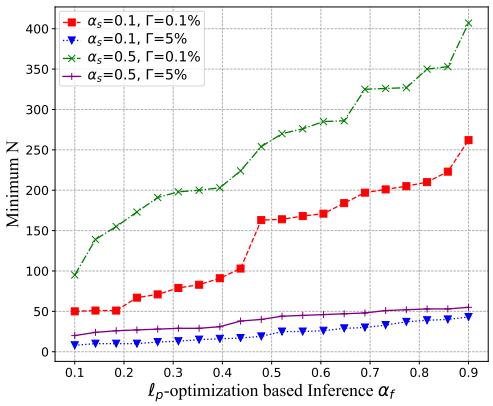
(α_f, α_s) \ Γ (%)	0.1	0.5	1	1.5	2	2.5	3	3.5	4	5
(0.1, 0.1)	30	25	27	29	20	26	27	20	28	23
(0.1, 0.5)	22	27	29	26	20	29	30	22	25	30
(0.5, 0.1)	24	26	26	23	26	29	21	29	26	29
(0.5, 0.5)	26	27	26	28	26	24	25	26	23	24



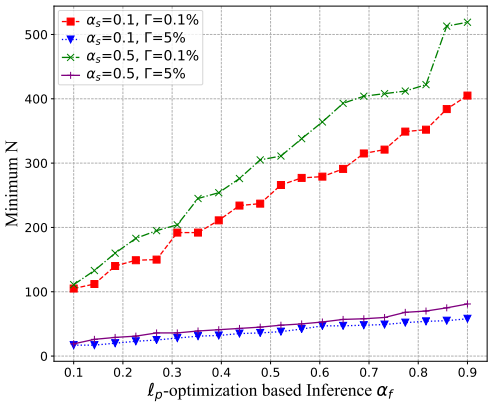
(a) N vs. Γ



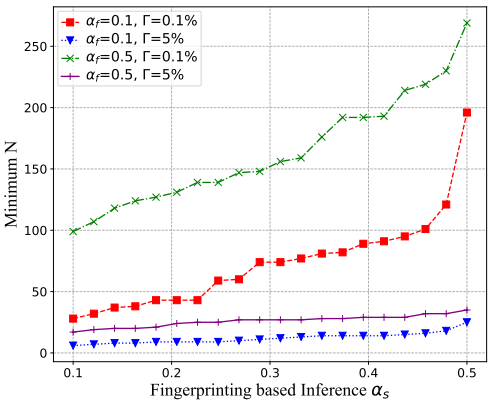
(b) N vs. Γ



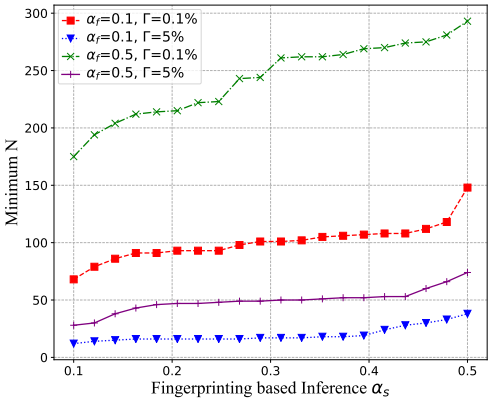
(c) N vs. α_f



(d) N vs. α_f



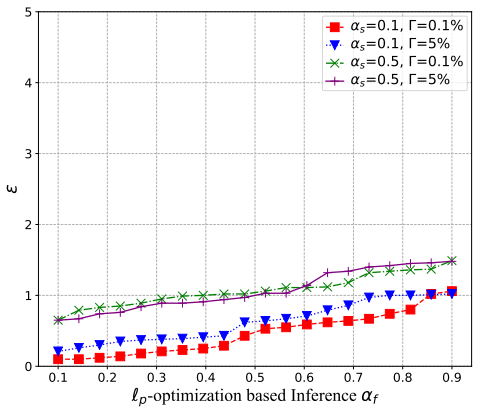
(e) N vs. α_s



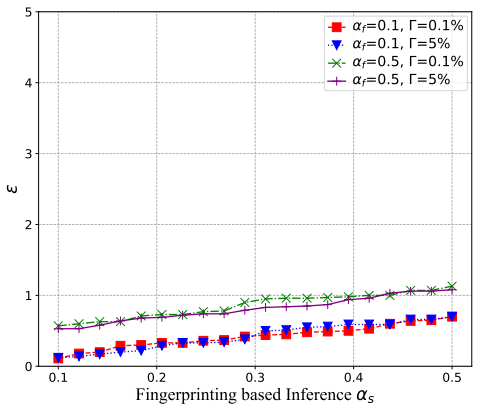
(f) N vs. α_s

Figure 8.8. Minimum N on Location Data (a,c,e), Network Data (b,d,f)

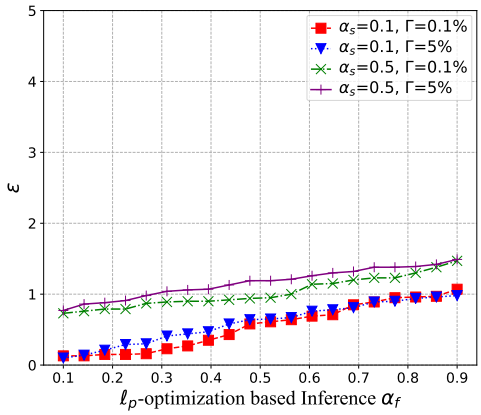
8.7.2 Indistinguishability. We also demonstrate the indistinguishablity bound ϵ w.r.t. different α_f, α_s and leakage bound Γ. As illustrated in Figure 8.9 (a,c) and (b,d), ϵ increases as α_s or α_f grows. This indicates that a stronger attacker would be more likely to identify the real data view. Moreover, ϵ is relatively small even if the adversary holds a strong background knowledge. For instance, in case of $\alpha_f = 90\%, \alpha_s = 50\%$, ϵ only equals 1.47 (which is also proven to be bounded in Theorem 10). Note that the leakage bound has no significant effect on ϵ since the indistinguishability among the data views is mainly determined by the background knowledge.



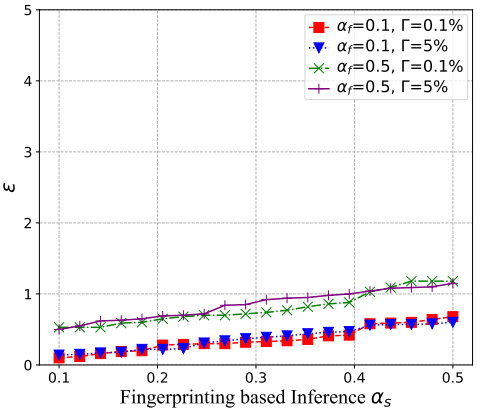
(a) ϵ vs. α_f



(b) ϵ vs. α_s



(c) ϵ vs. α_f



(d) ϵ vs. α_s

Figure 8.9. Indistinguishability on Location Data (a,b) and Network Data (c,d)

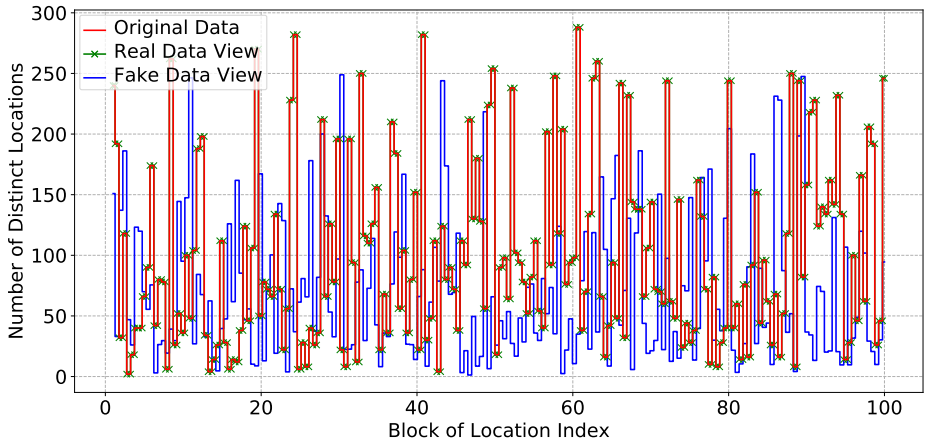
8.7.3 Utility of the Outsourced Data. Furthermore, we evaluate the utility for the outsourced location data using the Periodic Mobility Model (PMM) [299], which can be used to predict the mobility of the users in one week by analyzing their historical check-in data.

First, in Figure 8.10(a), we plot the location distribution of 100 blocks (each block includes multiple locations) at different times in the original data, real data view and fake data view. We observe that the distributions between the original data and the real data view are identical. Such 100% accuracy is ensured by the prefix preserving property in the outsourced data.

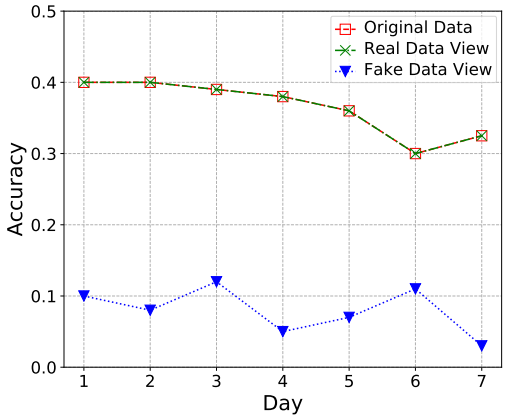
Second, we evaluate the accuracy and relative error distance for the real data view and one randomly selected fake data view. In Figure 8.10(b), the average accuracy of the real data views is exactly the same as the original dataset, both of which have a better accuracy than the fake data view. This also matches the fact that the outsourced data can fully preserve the prefixes of the real data without changing other attributes. Figure 8.10(c) demonstrates the results for relative error of distance, which also validate such excellent utility.

8.7.4 System Performance. Finally, we also evaluate the computational and communication overheads for outsourcing different datasets. In Figure 8.11(a), as the number of data views N grows, the runtime increases almost linearly for fixing different x as the prefix length to generate data partitions. While enlarging the prefix length x, the runtime also increases since the number of partitions also increases. Then, the overall computational costs become higher (with more CryptoPAn execution in more data partitions).

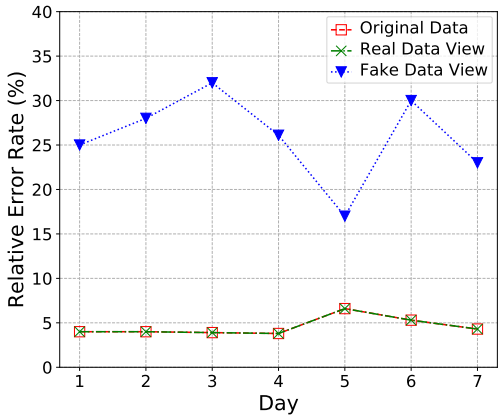
Figure 8.11(b) shows the experimental results of runtime versus different data sizes with fixed prefix length $x = 16$ (which is also a linear increase). We leverage



(a) Location Distribution



(b) Accuracy vs. Days

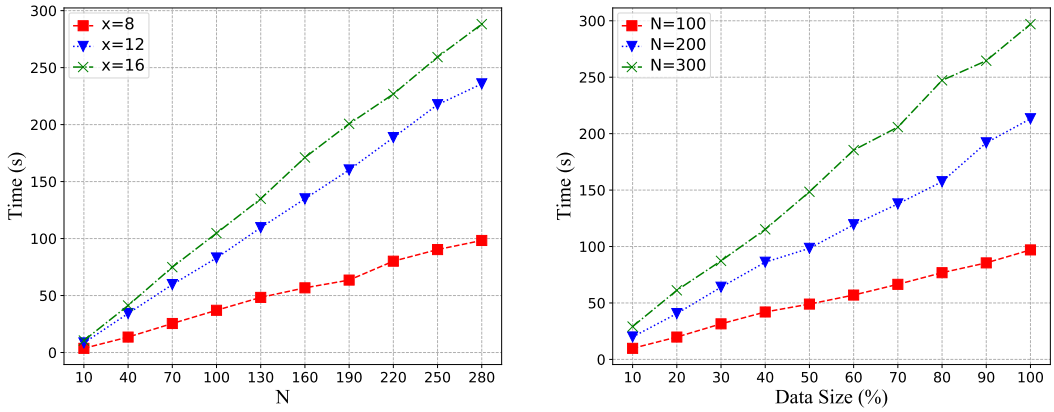


(c) Relative Error Rate vs. Days

Figure 8.10. Utility Evaluation on Location Data

the Path-ORAM [318, 319] to implement private result retrieval. The communication bandwidth is around 0.93MB and runtime is only 289ms for outsourcing the location data on average. Such experimental results are reasonable since the data owner only retrieves the analysis result corresponding to the real data view rather than the entire dataset. This is also confirmed in [319].

Finally, bounded by the same information leakage Γ, the proposed generalized framework requires less number of data views compared to [289], and thus reduces the



(a) Time vs. N (b) Time vs. Data Size ($x = 16$)

Figure 8.11. Running Time on Location Data

computational overheads at the service provider end (on analyzing all the generated data views). Thus, we also validate this using two real datasets. Specifically, given the prefix length, we apply both our generalized framework and multi-view framework for generating the data views to ensure the same bounded information leakage Γ. For the multi-view framework, we fix the length of prefix to be 16/23 for the network traffic and location data, respectively. Table 8.5 summarizes the running time of conducting the analysis on all the generated data views by both approaches with different leakage bound Γ. We can observe that our generalized framework needs less time to analyze all the data views on both two datasets.

8.8 Related Work

Securely Outsourcing Analysis. Securely outsourcing data analysis to third-party service providers has recently grown rapidly, especially with the increasing popularity of cloud technology [320, 321]. For this purpose, provably secure outsourcing has attracted significant attention during past decade. For instance, Sion et al. [322] define the requirements to build a secure outsourcing mechanism. Zhou et al. [323] propose a secure key management scheme which ensures that the source of the data can be

Table 8.5. Running Time (sec) vs. Information Leakage

Dataset	Scheme \ Leakage	1%	3%	5%	7%	10%
Network	Multi-view [289]	11.6	8.3	6.7	4.5	3.2
	Ours	8.2	6.5	5.8	3.7	2.8
Location	Multi-view [289]	16.2	13.3	11.0	7.5	5.4
	Ours	11.3	10.4	8.3	5.9	3.6

securely accessed by different parties under different requirements. Alternatively, oblivious random access memory (ORAM) [324] aims to hide the access patterns of the users, which has been well developed on different topics [318,325–327]. In addition, Franz et al. [328] propose a method which can make the data owner delegate rights to new clients for accessing to the outsourced data via a curious server based on ORAM. Stefanov et al. [318] propose a simple ORAM protocol with a small amount of client storage, which is formally proven to require small bandwidth and overheads.

Property Preserving Encryption Schemes. Broadly, various encryption schemes have been proposed to protect the data in different security levels, including fully homomorphic encryption (FHE) [76, 329], functional encryption [330, 331], searchable symmetric encryption [332, 333] and oblivious RAM (ORAM) [314, 319]. Moreover, there are a number of property preserving encryption schemes based on the CryptDB [334], such as order preserving encryption [287, 288] and deterministic encryption [285]. CryptoPAn [286] was proposed by Xu et al. to ensure the prefix preserving property on IP addresses from the cryptographic view. Kerschbaum [335] proposes a new order preserving encryption scheme which can hide the frequency pattern of plaintexts via

randomizing the ciphertexts to mitigate frequency analysis. Wang et al. [336] design a more efficient oblivious data structure which achieves a high efficiency.

Inference Attacks. Brekne et al. [298] presents the attacks via frequency analysis to compromise IP addresses under two prefix preserving anonymization schemes. There are several works which focus on the practical attacks to the encrypted data [292,293,337,338]. Islam et al. [338] introduce the first inference attack which leverages the leakage of access pattern and auxiliary information to get more information about the remaining queries. Naveed et al. [292] present a series of inference attacks on the property preserving encrypted database and implement the attacks on the medical databases to show the effectiveness of the attacks. Recently, Kellaris et al. [293] develop a generic reconstruction attacks on the range queries in the outsourced databases where the access patterns and communication volume are leaked.

CHAPTER 9

CONCLUSION

As IoT systems have been overwhelming to the whole society and environment, the trustworthiness of IoT is very important to ensure the sustainable development and social wellness. My dissertation has presented a complete research with two key trustworthy aspects, i.e., data privacy and robustness in aspects of two critical system components: MAS and ML systems. My work systemically studied various applications or domains in IoT systems, including smart grid, video recognition, natural language and cloud computing systems. To tackle the data privacy and robustness issues, I propose various privacy-enhancing and robust schemes with the integration of multiple foundational theories, such as applied cryptography, differential privacy. I believe my dissertation research can motivate to build more and more trustworthy IoT systems.

BIBLIOGRAPHY

[1] A. Sobe and W. Elmenreich, "Smart microgrids: Overview and outlook," *CoRR*, vol. abs/1304.3944, 2013.

[2] S. Maharjan, Q. Zhu, Y. Zhang, S. Gjessing, and T. Basar, "Dependable demand response management in the smart grid: A stackelberg game approach," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 120–132, 2013.

[3] W. Saad, Z. Han, H. V. Poor, and T. Basar, "Game-theoretic methods for the smart grid: An overview of microgrid systems, demand-side management, and smart grid communications," *IEEE Signal Process. Mag.*, vol. 29, no. 5, pp. 86–105, 2012.

[4] Z. Alibhai, W. Gruver, D. Kotak, and D. Sabaz, "Distributed coordination of micro-grids using bilateral contracts," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 2, 2004, pp. 1990–1995.

[5] A. Ouammi, H. Dagdougui, L. Dessaint, and R. Sacile, "Coordinated model predictive-based power flows control in a cooperative network of smart microgrids," *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 2233–2244, 2015. [Online]. Available: http://dx.doi.org/10.1109/TSG.2015.2396294

[6] Y. Hong, S. Goel, and W. M. Liu, "An efficient and privacy-preserving scheme for p2p energy exchange among smart microgrids," *International Journal of Energy Research*, vol. 40, no. 3, pp. 313–331, 2016.

[7] S. Xie, Y. Hong, and P.-J. Wan, "A privacy preserving multiagent system for load balancing in the smart grid," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019, pp. 2273–2275.

[8] V. Mathew, R. K. Sitaraman, and P. Shenoy, "Energy-aware load balancing in content delivery networks," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 954–962.

[9] M. Yu and S. H. Hong, "Supply and demand balancing for power management in smart grid: A stackelberg game approach," *Applied Energy*, vol. 164, pp. 702 – 710, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261915016128

[10] J. Schwill, "How to balance supply and demand on new electricity markets," *Gencom's Newsletter*, 2016.

[11] E. Csanyi, "Substation main functions and classification," http://electrical-engineering-portal.com/substation-main-functions-and-classification, accessed: 2018-04-10.

[12] G. Ács and C. Castelluccia, "I have a dream! (differentially private smart metering)," in *Information Hiding*, 2011, pp. 118–132.

[13] Y. Hong, W. M. Liu, and L. Wang, "Privacy preserving smart meter streaming against information leakage of appliance status," *IEEE Trans. Information Forensics and Security*, vol. 12, no. 9, pp. 2227–2241, 2017. [Online]. Available: https://doi.org/10.1109/TIFS.2017.2704904

[14] A. Hussain, V. Bui, and H. Kim, "A resilient and privacy-preserving energy management strategy for networked microgrids," *IEEE Trans. Smart Grid*, vol. 9, no. 3, pp. 2127–2139, 2018. [Online]. Available: https://doi.org/10.1109/TSG.2016.2607422

[15] L. Sankar, S. R. Rajagopalan, S. Mohajer, and H. V. Poor, "Smart meter privacy: A theoretical framework," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 837–846, 2013.

[16] S. Xie, Y. Hong, and P.-J. Wan, "A privacy preserving multiagent system for load balancing in the smart grid," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 2273–2275. [Online]. Available: http://dl.acm.org/citation.cfm?id=3306127.3332082

[17] S. Xie, Y. Hong, and P.-J. Wan, "Pairing: Privately balancing multiparty real-time supply and demand on the power grid," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1114–1127, 2020 ©2020 IEEE. Reprinted, with permission, from Shangyu Xie, Yuan Hong and Peng-Jun Wan, Pairing: Privately Balancing Multiparty Real-Time Supply and Demand on the Power Grid, IEEE Transactions on Information Forensics and Security, 2020.

[18] C. Rottondi, G. Verticale, and A. Capone, "Privacy-preserving smart metering with multiple data consumers," *Computer Networks*, vol. 57, no. 7, pp. 1699–1713, 2013.

[19] C.-K. Chu, J. K. Liu, J. W. Wong, Y. Zhao, and J. Zhou, "Privacy-preserving smart metering with regional statistics and personal enquiry services," in *ASI-ACCS*, 2013, pp. 369–380.

[20] K. Kursawe, G. Danezis, and M. Kohlweiss, "Privacy-friendly aggregation for the smart-grid," in *PETS'11*, 2011, pp. 175–191.

[21] C. Rottondi, A. Barbato, L. Chen, and G. Verticale, "Enabling privacy in a distributed game-theoretical scheduling system for domestic appliances," *IEEE Transactions on Smart Grid*, vol. 8, no. 3, pp. 1220–1230, May 2017.

[22] A. C. Yao, "How to generate and exchange secrets," in *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, IEEE. Los Alamitos, CA, USA: IEEE Computer Society, 1986, pp. 162–167.

[23] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game - a completeness theorem for protocols with honest majority," in *Proceedings of the 19th ACM Symposium on the Theory of Computing*. New York, NY: ACM, 1987, pp. 218–229. [Online]. Available: http://doi.acm.org/10.1145/28395.28420

[24] P. Paillier, "Public key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - Eurocrypt '99 Proceedings, LNCS 1592*, 1999, pp. 223–238.

[25] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay - secure two-party computation system," in *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, 2004, pp. 287–302. [Online]. Available: http://www.usenix.org/publications/library/proceedings/sec04/tech/malkhi.html

[26] B. Yang, H. Nakagawa, I. Sato, and J. Sakuma, "Collusion-resistant privacy-preserving data mining," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '10. New York, NY, USA: ACM, 2010, pp. 483–492. [Online]. Available: http://doi.acm.org/10.1145/1835804.1835867

[27] S. Sridhar and G. Manimaran, "Data integrity attacks and their impacts on scada control system," in *IEEE PES General Meeting*, July 2010, pp. 1–6.

[28] Y. Liu, M. K. Reiter, and P. Ning, "False data injection attacks against state estimation in electric power grids," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, 2009, pp. 21–32. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653666

[29] R. Lasseter, "Microgrids," in *Power Engineering Society Winter Meeting, 2002. IEEE*, vol. 1, 2002, pp. 305–308.

[30] Z. Huang, T. Zhu, D. Irwin, A. Mishra, D. Menasche, and P. Shenoy, "Minimizing transmission loss in smart microgrids by sharing renewable energy," *ACM Trans. Cyber-Phys. Syst.*, vol. 1, no. 2, pp. 5:1–5:22, Dec. 2016. [Online]. Available: http://doi.acm.org/10.1145/2823355

[31] R. A. van der Veen and R. A. Hakvoort, "The electricity balancing market: Exploring the design challenge," *Utilities Policy*, vol. 43, pp. 186 – 194, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957178716303125

[32] A. Ricci, B. Vinerba, E. Smargiassi, I. D. Munari, V. Aisa, and P. Ciampolini, "Power-grid load balancing by using smart home appliances," in *2008 Digest of Technical Papers - International Conference on Consumer Electronics*, Jan 2008, pp. 1–2.

[33] D. Naccache and J. Stern, "A new public-key cryptosystem," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1997, pp. 27–36.

[34] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1998, pp. 308–318.

[35] P. Vithayasrichareon, T. Lozanov, J. Riesz, and I. MacGill, "Impact of operational constraints on generation portfolio planning with renewables," in *2015 IEEE Power Energy Society General Meeting*, July 2015, pp. 1–5.

[36] D. Montenegro, M. Hernandez, and G. Ramos, "Real time opendss framework for distribution systems simulation and analysis," in *2012 Sixth IEEE/PES Transmission and Distribution: Latin America Conference and Exposition (T&D-LA)*. IEEE, 2012, pp. 1–5.

[37] O. Goldreich, "Secure multi-party computation," Sep. 1998, (working draft). [Online]. Available: http://www.wisdom.weizmann.ac.il/~oded/pp.html

[38] "tor," https://www.torproject.org/, accessed: 2018-05-10.

[39] O. Goldreich, *The Foundations of Cryptography.* Cambridge University Press, 2004, vol. 2, ch. Encryption Schemes. [Online]. Available: http://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/enc.ps

[40] J. Katz and Y. Lindell, *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series).* Chapman & Hall/CRC, 2007.

[41] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology — CRYPTO' 86*, A. M. Odlyzko, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194.

[42] Y. Hong, H. Wang, S. Xie, and B. Liu, "Privacy preserving and collusion resistant energy sharing," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 6941–6945.

[43] Z. Wang, K. Yang, and X. Wang, "Privacy-preserving energy scheduling in microgrid systems," *IEEE Trans. Smart Grid*, vol. 4, no. 4, pp. 1810–1820, 2013.

[44] T. Zhu, S. Xiao, Y. Ping, D. Towsley, and W. Gong, "A secure energy routing mechanism for sharing renewable energy in smart microgrid," in *SmartGridComm*, 2011, pp. 143–148.

[45] J. Vaidya, M. Kantarcioglu, and C. Clifton, "Privacy preserving naive bayes classification," *International Journal on Very Large Data Bases*, vol. 17, no. 4, pp. 879–898, Jul. 2008.

[46] J. Alwen, J. Katz, Y. Lindell, G. Persiano, a. shelat, and I. Visconti, "Collusion-free multiparty computation in the mediated model," in *Advances in Cryptology - CRYPTO 2009*, S. Halevi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 524–540.

[47] X. Wang, S. Ranellucci, and J. Katz, "Authenticated garbling and efficient maliciously secure two-party computation," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, 2017, pp. 21–37. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134053

[48] R. A. Popa, A. J. Blumberg, H. Balakrishnan, and F. H. Li, "Privacy and accountability for location-based aggregate statistics," in *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, 2011, pp. 653–666. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046781

[49] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, "Efficient genome-wide, privacy-preserving similar patient query based on private edit distance," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, 2015, pp. 492–503. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813725

[50] F. G. Mármol, C. Sorge, O. Ugus, and G. M. Pérez, "Do not snoop my habits: preserving privacy in the smart grid," *IEEE Communications Magazine*, vol. 50, no. 5, pp. 166–172, 2012.

[51] W. Yang, N. Li, Y. Qi, W. H. Qardaji, S. E. McLaughlin, and P. McDaniel, "Minimizing private data disclosures in the smart grid," in *ACM Conference on Computer and Communications Security*, 2012, pp. 415–427.

[52] O. Tan, D. Gündüz, and H. V. Poor, "Increasing smart meter privacy through energy harvesting and storage devices," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 7, pp. 1331–1341, 2013.

[53] A. Rial and G. Danezis, "Privacy-preserving smart metering," in *WPES*, 2011, pp. 49–60.

[54] Z. Erkin and G. Tsudik, "Private computation of spatial and temporal power consumption with smart meters," in *ACNS*, 2012, pp. 561–577.

[55] H.-Y. Lin, W.-G. Tzeng, S.-T. Shen, and B.-S. P. Lin, "A practical smart metering system supporting privacy preserving billing and load monitoring," in *ACNS*, 2012, pp. 544–560.

[56] X. Liang, X. Li, R. Lu, X. Lin, and X. Shen, "Udp: Usage-based dynamic pricing with privacy preservation for smart grid," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 141–150, 2013.

[57] C. Tham and T. Luo, "Sensing-driven energy purchasing in smart grid cyber-physical system," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 4, pp. 773–784, July 2013.

[58] S. D. J. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, and T. Funabashi, "Multi-agent systems for power engineering applications—part i: Concepts, approaches, and technical challenges," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1743–1752, Nov 2007.

[59] J. Cerquides, G. Picard, and J. A. Rodríguez-Aguilar, "Designing a marketplace for the trading and distribution of energy in the smart grid," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, 2015, pp. 1285–1293. [Online]. Available: http://dl.acm.org/citation.cfm?id=2773316

[60] S. Xie, Y. Hong, and P. Wan, "A privacy preserving multiagent system for load balancing in the smart grid," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, 2019, pp. 2273–2275. [Online]. Available: http://dl.acm.org/citation.cfm?id=3332082

[61] W. Tushar, B. Chai, C. Yuen, D. B. Smith, K. L. Wood, Z. Yang, and H. V. Poor, "Three-party energy management with distributed energy resources in smart grid," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2487–2498, April 2015.

[62] S. Nguyen, W. Peng, P. Sokolowski, D. Alahakoon, and X. Yu, "Optimizing rooftop photovoltaic distributed generation with battery storage for peer-to-peer energy trading," *Applied Energy*, vol. 228, pp. 2567–2580, 2018.

[63] P. Agrawal, A. Kumar, and P. Varakantham, "Near-optimal decentralized power supply restoration in smart grids," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, 2015, pp. 1275–1283. [Online]. Available: http://dl.acm.org/citation.cfm?id=2773315

[64] Y. Hong, H. Wang, S. Xie, and B. Liu, "Privacy preserving and collusion resistant energy sharing," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6941–6945.

[65] W. Tushar, C. Yuen, H. Mohsenian-Rad, T. Saha, H. V. Poor, and K. L. Wood, "Transforming energy networks via peer-to-peer energy trading: The potential of game-theoretic approaches," *IEEE Signal Processing Magazine*, vol. 35, no. 4, pp. 90–111, 2018.

[66] C. Zhang, J. Wu, M. Cheng, Y. Zhou, and C. Long, "A bidding system for peer-to-peer energy trading in a grid-connected microgrid," *Energy Procedia*, vol. 103, pp. 147–152, 2016, renewable Energy Integration with Mini/Microgrid – Proceedings of REM2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1876610216314746

[67] E. McKenna and M. Thomson, "Photovoltaic metering configurations, feed-in tariffs and the variable effective electricity prices that result," *IET Renewable Power Generation*, vol. 7, no. 3, pp. 235–245, 2013.

[68] F. Fioretto, W. Yeoh, E. Pontelli, Y. Ma, and S. J. Ranade, "A distributed constraint optimization (DCOP) approach to the economic dispatch with demand response," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, 2017, pp. 999–1007. [Online]. Available: http://dl.acm.org/citation.cfm?id=3091267

[69] S. Xie, H. Wang, Y. Hong, and M. Thai, "Privacy preserving distributed energy trading," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pp. 322–332 ©2020 IEEE. Reprinted, with permission, from Shangyu Xie, Han Wang, Yuan Hong and My Thai,Privacy preserving distributed energy trading, IEEE ICDCS, 2020.

[70] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 1999.

[71] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.

[72] W. Yang, N. Li, Y. Qi, W. Qardaji, S. McLaughlin, and P. McDaniel, "Minimizing private data disclosures in the smart grid," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 415–427. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382242

[73] A. Mas-Colell, M. Whinston, and J. Green, *Microeconomic Theory*, 1995.

[74] P. Samadi, A. Mohsenian-Rad, R. Schober, V. W. S. Wong, and J. Jatskevich, "Optimal real-time pricing algorithm based on utility maximization for smart grid," in *2010 First IEEE International Conference on Smart Grid Communications*, Oct 2010, pp. 415–420.

[75] A. Ben-David, N. Nisan, and B. Pinkas, "Fairplaymp: a system for secure multi-party computation," in *Proceedings of the 15th ACM conference on Computer and communications security*, ser. CCS '08. New York, NY, USA: ACM, 2008, pp. 257–266. [Online]. Available: http://doi.acm.org/10.1145/1455770.1455804

[76] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, 2009, pp. 169–178. [Online]. Available: https://doi.org/10.1145/1536414.1536440

[77] A. T. Al-Awami and E. Sortomme, "Coordinating vehicle-to-grid services with energy trading," *IEEE Transactions on smart grid*, vol. 3, no. 1, pp. 453–462, 2011.

[78] Z. Wang, C. Gu, F. Li, P. Bale, and H. Sun, "Active demand response using shared energy storage for household energy management," *IEEE Transactions on Smart Grid*, vol. 4, no. 4, pp. 1888–1897, 2013.

[79] L. Yang, H. Kim, J. Zhang, M. Chiang, and C. W. Tan, "Pricing-based decentralized spectrum access control in cognitive radio networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 2, pp. 522–535, April 2013.

[80] X. Yuan, X. Qin, F. Tian, Y. T. Hou, W. Lou, S. F. Midkiff, and J. H. Reed, "Coexistence between wi-fi and lte on unlicensed spectrum: A human-centric approach," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 4, pp. 964–977, April 2017.

[81] Y. Wang, W. Saad, Z. Han, H. V. Poor, and T. Başar, "A game-theoretic approach to energy trading in the smart grid," *IEEE Transactions on Smart Grid*, vol. 5, no. 3, pp. 1439–1450, 2014.

[82] S. Xie, Y. Hong, and P. Wan, "Pairing: Privately balancing multiparty real-time supply and demand on the power grid," *IEEE Transactions on Information Forensics and Security*, vol. ©, pp. 1–1, 2019.

[83] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[84] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 839–858.

[85] A. Barak, M. Hirt, L. Koskas, and Y. Lindell, "An end-to-end system for large scale p2p mpc-as-a-service and low-bandwidth mpc for weak participants," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 695–712. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243801

[86] J. Furukawa, Y. Lindell, A. Nof, and O. Weinstein, "High-throughput secure three-party computation for malicious adversaries and an honest majority," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 225–255.

[87] M. Byali, A. Joseph, A. Patra, and D. Ravi, "Fast secure computation for small population over the internet," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 677–694. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243784

[88] S. Barker, A. Mishra, D. Irwin, P. Shenoy, and J. Albrecht, "SmartCap: Flattening peak electricity demand in smart homes," *2012 IEEE International Conference on Pervasive Computing and Communications, PerCom 2012*, pp. 67–75, 2012.

[89] X. He, X. Zhang, and C.-C. J. Kuo, "A distortion-based approach to privacy-preserving metering in smart grids," *IEEE Practical Innovations: Open Solutions*, vol. 1, no. 3, pp. 67–78, 2013.

[90] T. Dimitriou and G. Karame, "Privacy-friendly tasking and trading of energy in smart grids," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013, pp. 652–659.

[91] H. Wang and J. Huang, "Incentivizing energy trading for interconnected microgrids," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 2647–2657, July 2018.

[92] H. S. V. S. K. Nunna and S. Doolla, "Multiagent-based distributed-energy-resource management for intelligent microgrids," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 4, pp. 1678–1687, April 2013.

[93] K. Gai, Y. Wu, L. Zhu, M. Qiu, and M. Shen, "Privacy-preserving energy trading using consortium blockchain in smart grid," *IEEE Transactions on Industrial Informatics*, 2019.

[94] J. Yang, P. Ren, D. Zhang, D. Chen, F. Wen, H. Li, and G. Hua, "Neural aggregation network for video face recognition," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 5216–5225.

[95] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 1933–1941.

[96] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 6479–6488.

[97] V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, and J. K. Hedrick, "Learning a deep neural net policy for end-to-end control of autonomous vehicles," in *2017 American Control Conference, ACC 2017, Seattle, WA, USA, May 24-26, 2017*, 2017, pp. 4914–4919.

[98] T. Onishi, T. Motoyoshi, Y. Suga, H. Mori, and T. Ogata, "End-to-end learning method for self-driving cars with trajectory recovery using a path-following function," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[99] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[100] S. Xie, H. Wang, Y. Kong, and Y. Hong, "Universal 3-dimensional perturbations for black-box attacks on video recognition systems," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 1390–1407 ©2022 IEEE. Reprinted, with permission, from Shangyu Xie, Han Wang, Yu Kong and Yuan Hong, Universal 3–Dimensional Perturbations for Black–Box Attacks on Video Recognition Systems, IEEE Symposium on Security and Privacy, 2022.

[101] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014*, 2014.

[102] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 2574–2582.

[103] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 86–94.

[104] K. T. Co, L. Muñoz-González, S. de Maupeou, and E. C. Lupu, "Procedural noise adversarial examples for black-box attacks on deep convolutional networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, 2019, pp. 275–289.

[105] Y. Mirsky, T. Mahler, I. Shelef, and Y. Elovici, "Ct-gan: Malicious tampering of 3d medical imagery using deep learning," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 461–478.

[106] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," *arXiv preprint arXiv:1812.05271*, 2018.

[107] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 513–530.

[108] G. Chen, S. Chen, L. Fan, X. Du, Z. Zhao, F. Song, and Y. Liu, "Who is real bob? adversarial attacks on speaker recognition systems," *arXiv preprint arXiv:1911.01840*, 2019.

[109] Z. Li, Y. Wu, J. Liu, Y. Chen, and B. Yuan, "Advpulse: Universal, synchronization-free, and targeted audio adversarial attacks via subsecond perturbations," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, p. 1121–1134.

[110] X. Wei, J. Zhu, S. Yuan, and H. Su, "Sparse adversarial perturbations for videos," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8973–8980.

[111] S. Li, A. Neupane, S. Paul, C. Song, S. V. Krishnamurthy, A. K. Roy-Chowdhury, and A. Swami, "Stealthy adversarial perturbations against real-time video classification systems," in *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[112] Z. Wei, J. Chen, X. Wei, L. Jiang, T.-S. Chua, F. Zhou, and Y.-G. Jiang, "Heuristic black-box adversarial attacks on video recognition models," *arXiv preprint arXiv:1911.09449*, 2019.

[113] L. Jiang, X. Ma, S. Chen, J. Bailey, and Y.-G. Jiang, "Black-box adversarial attacks on video recognition models," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 864–872.

[114] C. Xiao, R. Deng, B. Li, T. Lee, B. Edwards, J. Yi, D. Song, M. Liu, and I. Molloy, "Advit: Adversarial frames identifier based on temporal consistency in videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3968–3977.

[115] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[116] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.

[117] S. Cheng, Y. Dong, T. Pang, H. Su, and J. Zhu, "Improving black-box adversarial attacks with a transfer-based prior," *arXiv preprint arXiv:1906.06919*, 2019.

[118] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[119] A. Shafahi, M. Najibi, Z. Xu, J. Dickerson, L. S. Davis, and T. Goldstein, "Universal adversarial training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5636–5643.

[120] X. Yin, S. Kolouri, and G. K. Rohde, "Adversarial example detection and classification with asymmetrical adversarial training," *arXiv preprint arXiv:1905.11475*, 2019.

[121] M. Lécuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, 2019, pp. 656–672.

[122] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," *arXiv preprint arXiv:1902.02918*, 2019.

[123] F. Suya, J. Chi, D. Evans, and Y. Tian, "Hybrid batch attacks: Finding black-box adversarial examples with limited queries," in *29th USENIX Security Symposium (USENIX Security 20)*, Aug. 2020, pp. 1327–1344.

[124] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, 2015, pp. 4489–4497.

[125] M. S. Khandare and A. Mahajan, "Mobile monitoring system for smart home," in *2010 3rd International Conference on Emerging Trends in Engineering and Technology*. IEEE, 2010, pp. 848–852.

[126] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. Li, "Large-scale video classification with convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA*, 2014, pp. 1725–1732.

[127] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[128] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[129] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 4724–4733.

[130] A. Costin, "Security of cctv and video surveillance systems: Threats, vulnerabilities, attacks, and mitigations," in *Proceedings of the 6th international workshop on trustworthy embedded devices*, 2016.

[131] J. Obermaier and M. Hutle, "Analyzing the security and privacy of cloud-based video surveillance systems," in *Proceedings of the 2nd ACM international workshop on IoT privacy, trust, and security*, 2016.

[132] N. Kalbo, Y. Mirsky, A. Shabtai, and Y. Elovici, "The security of ip-based video surveillance systems," *Sensors*, vol. 20, no. 17, 2020.

[133] H. Kuehne, H. Jhuang, E. Garrote, T. A. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, 2011, pp. 2556–2563.

[134] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 15–26.

[135] 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.

[136] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 2137–2146.

[137] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[138] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[139] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," *arXiv preprint arXiv:2002.08347*, 2020.

[140] A. Athalye and N. Carlini, "On the robustness of the cvpr 2018 white-box adversarial example defenses," *arXiv preprint arXiv:1804.03286*, 2018.

[141] N. Carlini and D. Wagner, "Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples," 2017.

[142] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in neural information processing systems*, 2006, pp. 739–746.

[143] K. Perlin, "An image synthesizer," *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, p. 287–296, Jul. 1985. [Online]. Available: https://doi.org/10.1145/325165.325247

[144] A. Lagae, S. Lefebvre, R. Cook, T. DeRose, G. Drettakis, D. S. Ebert, J. P. Lewis, K. Perlin, and M. Zwicker, "A survey of procedural noise functions," in *Computer Graphics Forum*, vol. 29, no. 8. Wiley Online Library, 2010, pp. 2579–2600.

[145] K. Perlin, "Improving noise," *ACM Trans. Graph.*, vol. 21, no. 3, p. 681–682, Jul. 2002. [Online]. Available: https://doi.org/10.1145/566654.566636

[146] A. Lagae, S. Lefebvre, G. Drettakis, and P. Dutré, "Procedural noise using sparse gabor convolution," *ACM Trans. Graph.*, vol. 28, no. 3, p. 54, 2009.

[147] D. A. Szafir, "Modeling color difference for visualization design," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 392–401, 2017.

[148] D. Gabor, "Theory of communication. part 1: The analysis of information," *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[149] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *European conference on computer vision*. Springer, 2010, pp. 143–156.

[150] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. Ieee, 1995, pp. 39–43.

[151] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," 1988.

[152] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*. Springer, 1987, pp. 7–15.

[153] F. Glover, "Tabu search—part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[154] R. Hassan, B. Cohanim, O. De Weck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference*, 2005, p. 1897.

[155] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. abs/1212.0402, 2012. [Online]. Available: http://arxiv.org/abs/1212.0402

[156] C. Osborne, "amazon-surveillance-cameras-infected-with-malware," 2016. [Online]. Available: https://www.zdnet.com/article/amazon-surveillance-cameras-infected-with-malware/

[157] Shodan, "Shodan report," 2016. [Online]. Available: https://www.shodan.io/report/UMAja2tN

[158] "Ffmpeg guide," 2020. [Online]. Available: https://ffmpeg.org/ffmpeg.html

[159] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[160] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" *arXiv preprint arXiv:1904.12843*, 2019.

[161] 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*, 2017, pp. 135–147.

[162] K. Roth, Y. Kilcher, and T. Hofmann, "The odds are odd: A statistical test for detecting adversarial examples," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 5498–5507.

[163] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1778–1787.

[164] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[165] A. Kumar, A. Levine, T. Goldstein, and S. Feizi, "Curse of dimensionality on randomized smoothing for certifiable robustness," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5458–5467.

[166] A. Blum, T. Dick, N. Manoj, and H. Zhang, "Random smoothing might be unable to certify ℓ_∞ robustness for high-dimensional images," *Journal of Machine Learning Research*, vol. 21, no. 211, pp. 1–21, 2020.

[167] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5286–5295.

[168] T. Pang, K. Xu, and J. Zhu, "Mixup inference: Better exploiting mixup to defend adversarial attacks," *arXiv preprint arXiv:1909.11515*, 2019.

[169] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[170] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[171] M. Cheng, T. Le, P. Chen, H. Zhang, J. Yi, and C. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[172] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," *arXiv preprint arXiv:1606.04435*, 2016.

[173] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, p. 506–519. [Online]. Available: https://doi.org/10.1145/3052973.3053009

[174] 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.

[175] S. Xie, Y. Yan, and Y. Hong, "Stealthy 3d poisoning attack on video recognition models," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2022 ©2022 IEEE. Reprinted, with permission, from Shangyu Xie, Yan Yan and Yuan Hong, Stealthy 3D Poisoning Attack on Video Recognition Models, IEEE Transactions on Dependable and Secure Computing, 2022.

[176] 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.

[177] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 6103–6113.

[178] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[179] O. Suciu, R. Marginean, Y. Kaya, H. D. III, and T. Dumitras, "When does machine learning FAIL? generalized transferability for evasion and poisoning attacks," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1299–1316.

[180] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 7614–7623. [Online]. Available: http://proceedings.mlr.press/v97/zhu19a.html

[181] A. Turner, D. Tsipras, and A. Madry, "Clean-label backdoor attacks," 2018.

[182] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[183] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 957–11 965.

[184] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, and Y.-G. Jiang, "Clean-label backdoor attacks on video recognition models," in *IEEE CVPR*, 2020, pp. 14 443–14 452.

[185] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," in *NIPS*, 2017.

[186] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *NIPS*, 2018, pp. 8000–8010.

[187] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 273–294.

[188] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *IEEE SP*. IEEE, 2019, pp. 707–723.

[189] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[190] C. Feichtenhofer, "X3d: Expanding architectures for efficient video recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 203–213.

[191] D. Tran, H. Wang, L. Torresani, and M. Feiszli, "Video classification with channel-separated convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5552–5561.

[192] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[193] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.

[194] K. Perlin, "Improving noise," ser. SIGGRAPH'02, 2002.

[195] G. Takács, "Convex polyhedron learning and its applications," 2009.

[196] J. Bao, Y.-H. He, E. Hirst, J. Hofscheier, A. Kasprzyk, and S. Majumder, "Polytopes and machine learning," *arXiv preprint arXiv:2109.09602*, 2021.

[197] V. Sehwag, A. N. Bhagoji, L. Song, C. Sitawarin, D. Cullina, M. Chiang, and P. Mittal, "Analyzing the robustness of open-world machine learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, ser. AISec'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 105–116. [Online]. Available: https://doi.org/10.1145/3338501.3357372

[198] N. Inkawhich, W. Wen, H. H. Li, and Y. Chen, "Feature space perturbations yield more transferable adversarial examples," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7066–7074.

[199] T. Goldstein, C. Studer, and R. Baraniuk, "A field guide to forward-backward splitting with a fasta implementation," *arXiv preprint arXiv:1411.3406*, 2014.

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

[201] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[202] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[203] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[204] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," 2009.

[205] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *arXiv preprint arXiv:1801.09344*, 2018.

[206] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[207] C. Zhang and Y. Ma, *Ensemble machine learning: methods and applications*. Springer, 2012.

[208] E. Wenger, J. Passananti, A. N. Bhagoji, Y. Yao, H. Zheng, and B. Y. Zhao, "Backdoor attacks against deep learning systems in the physical world," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6206–6215.

[209] M. Shen, Z. Liao, L. Zhu, K. Xu, and X. Du, "Vla: A practical visible light-based attack on face recognition systems in physical world," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 3, no. 3, sep 2019. [Online]. Available: https://doi.org/10.1145/3351261

[210] T. Pham, T. Tran, D. Phung, and S. Venkatesh, "Predicting healthcare trajectories from medical records: A deep learning approach," *Journal of biomedical informatics*, vol. 69, pp. 218–229, 2017.

[211] A. Hosny, C. Parmar, J. Quackenbush, L. H. Schwartz, and H. J. Aerts, "Artificial intelligence in radiology," *Nature Reviews Cancer*, vol. 18, no. 8, pp. 500–510, 2018.

[212] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[213] M. Chen, B. Lee, G. Bansal, Y. Cao, S. Zhang, J. Y. Lu, J. Tsay, Y. Wang, A. M. Dai, Z. Chen, T. Sohn, and Y. Wu, "Gmail smart compose: Real-time assisted writing," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[214] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[215] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization." *Journal of Machine Learning Research*, vol. 12, no. 3, 2011.

[216] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 19–38.

[217] J. Cabrero-Holgueras and S. Pastrana, "Sok: Privacy-preserving computation techniques for deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 4, pp. 139–162, 2021.

[218] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 35–52.

[219] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1999, pp. 223–238.

[220] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.

[221] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3-4, pp. 211–407, 2014.

[222] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 2014, pp. 464–473.

[223] 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*, 2016, pp. 308–318.

[224] J. Vaidya, B. Shafiq, A. Basu, and Y. Hong, "Differentially private naive bayes classification," in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 1. IEEE, 2013, pp. 571–576.

[225] M. Mohammady, S. Xie, Y. Hong, M. Zhang, L. Wang, M. Pourzandi, and M. Debbabi, "R2dp: A universal and automated approach to optimizing the randomization mechanisms of differential privacy for utility metrics with no known optimal distributions," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 677–696. [Online]. Available: https://doi.org/10.1145/3372297.3417259

[226] Y. Huang, Z. Song, D. Chen, K. Li, and S. Arora, "TextHide: Tackling data privacy in language understanding tasks," in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for

Computational Linguistics, Nov. 2020, pp. 1368–1382. [Online]. Available: https://www.aclweb.org/anthology/2020.findings-emnlp.123

[227] Y. Huang, Z. Song, K. Li, and S. Arora, "InstaHide: Instance-hiding schemes for private distributed learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 4507–4518. [Online]. Available: http://proceedings.mlr.press/v119/huang20i.html

[228] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[229] S. Xie and Y. Hong, "Reconstruction attack on instance encoding for language understanding," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 2038–2044. [Online]. Available: https://aclanthology.org/2021.emnlp-main.154

[230] S. Xie and Y. Hong, "Differentially private instance encoding against privacy attacks," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*. Hybrid: Seattle, Washington + Online: Association for Computational Linguistics, Jul. 2022, pp. 172–180. [Online]. Available: https://aclanthology.org/2022.naacl-srw.22

[231] A. Abboud and K. Lewi, "Exact weight subgraphs and the k-sum conjecture," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2013, pp. 1–12.

[232] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[233] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018.

[234] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2615–2632.

[235] C. Song and A. Raghunathan, "Information leakage in embedding models," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 377–390.

[236] S. Hisamoto, M. Post, and K. Duh, "Membership inference attacks on sequence-to-sequence models: Is my data in your machine translation system?" *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 49–63, 2020. [Online]. Available: https://www.aclweb.org/anthology/2020.tacl-1.4

[237] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer, "Membership inference attacks from first principles," *arXiv preprint arXiv:2112.03570*, 2021.

[238] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1322–1333. [Online]. Available: https://doi.org/10.1145/2810103.2813677

[239] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton, "A methodology for formalizing model-inversion attacks," in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE, 2016, pp. 355–370.

[240] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf

[241] N. Carlini, S. Deng, S. Garg, S. Jha, S. Mahloujifar, M. Mahmoody, S. Song, A. Thakurta, and F. Tramer, "An attack on instahide: Is private learning possible with instance encoding?" *arXiv preprint arXiv:2011.05315*, 2020.

[242] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," *arXiv preprint arXiv:2012.07805*, 2020.

[243] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 937–16 947, 2020.

[244] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 202–210. [Online]. Available: https://doi.org/10.1145/773153.773173

[245] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," *arXiv preprint arXiv:1611.04482*, 2016.

[246] G. Kerrigan, D. Slack, and J. Tuyls, "Differentially private language models benefit from public pre-training," in *Proceedings of the Second Workshop on Privacy in NLP*. Online: Association for Computational Linguistics, Nov. 2020, pp. 39–45. [Online]. Available: https://aclanthology.org/2020.privatenlp-1.5

[247] D. Yu, S. Naik, A. Backurs, S. Gopi, H. A. Inan, G. Kamath, J. Kulkarni, Y. T. Lee, A. Manoel, L. Wutschitz *et al.*, "Differentially private fine-tuning of language models," *arXiv preprint arXiv:2110.06500*, 2021.

[248] X. Li, F. Tramer, P. Liang, and T. Hashimoto, "Large language models can be strong differentially private learners," *arXiv preprint arXiv:2110.05679*, 2021.

[249] C. Dupuy, R. Arava, R. Gupta, and A. Rumshisky, "An efficient dp-sgd mechanism for large scale nlu models," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4118–4122.

[250] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[251] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. [Online]. Available: https://www.aclweb.org/anthology/N18-1101

[252] A. Warstadt, A. Singh, and S. R. Bowman, "Neural network acceptability judgments," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 625–641, Mar. 2019. [Online]. Available: https://www.aclweb.org/anthology/Q19-1040

[253] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. [Online]. Available: https://www.aclweb.org/anthology/D13-1170

[254] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[255] C. Dwork and A. Smith, "Differential privacy for statistics: What we know and what we want to learn," *Journal of Privacy and Confidentiality*, vol. 1, no. 2, 2010.

[256] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2006, pp. 486–503.

[257] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[258] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[259] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[260] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 1209–1222.

[261] 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)*, 2016, pp. 601–618.

[262] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "{GAZELLE}: A low latency framework for secure neural network inference," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 1651–1669.

[263] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, 2016, pp. 201–210.

[264] P. Xie, B. Wu, and G. Sun, "Bayhenn: Combining bayesian deep learning and homomorphic encryption for secure dnn inference," *arXiv preprint arXiv:1906.00639*, 2019.

[265] Q. Zhang, C. Wang, H. Wu, C. Xin, and T. V. Phuong, "Gelu-net: A globally encrypted, locally unencrypted deep neural network for privacy-preserved learning."

[266] S. Xie, B. Liu, and Y. Hong, "Privacy-preserving cloud-based dnn inference," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021 ©2021 IEEE. Reprinted, with permission, from Shangyu Xie, Bingyu Liu and Yuan Hong, Privacy-Preserving Cloud-Based DNN Inference, IEEE ICASSP, 2021, pp. 2675–2679.

[267] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437.

[268] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[269] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[270] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[271] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

[272] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, 1987, pp. 218–229.

[273] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 169–178. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536440

[274] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, and M. Orrù, "Homomorphic secret sharing: optimizations and applications," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 2105–2122.

[275] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *Cryptology ePrint Archive*, 2012.

[276] S. Halevi and V. Shoup, "Faster homomorphic linear transformations in helib," in *Annual International Cryptology Conference*. Springer, 2018, pp. 93–120.

[277] S. Halevi, Y. Polyakov, and V. Shoup, "An improved rns variant of the bfv homomorphic encryption scheme," in *Cryptographers' Track at the RSA Conference*. Springer, 2019, pp. 83–105.

[278] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 19–38.

[279] C. Lu, C. Hsieh, C. Chang, and C. Yang, "An improvement to data service in cloud computing with content sensitive transaction analysis and adaptation," in *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*, July 2013, pp. 463–468.

[280] N. Perlroth, "All 3 billion yahoo accounts were affected by 2013 attack," https://www.nytimes.com/2017/10/03/technology/yahoo-hack-3-billion-users.html, 2017.

[281] "Gdpr," https://gdpr-info.eu/art-28-gdpr/, accessed: 2019-10-30.

[282] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," 1999.

[283] P. Karn, W. A. Simpson, and P. Metzger, "The esp triple des transform," 1995.

[284] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, 2010, pp. 24–43. [Online]. Available: https://doi.org/10.1007/978-3-642-13190-5_2

[285] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Annual International Cryptology Conference*. Springer, 2007, pp. 535–552.

[286] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon, "Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme," in *10th IEEE International Conference on Network Protocols, 2002. Proceedings.* IEEE, 2002, pp. 280–289.

[287] A. Boldyreva, N. Chenette, Y. Lee, and A. O'neill, "Order-preserving symmetric encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2009, pp. 224–241.

[288] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 2004, pp. 563–574.

[289] M. Mohammady, L. Wang, Y. Hong, H. Louafi, M. Pourzandi, and M. Debbabi, "Preserving both privacy and utility in network trace anonymization," in *Proceedings of the 2018 ACM SIGSAC Conference*

on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018, 2018, pp. 459–474. [Online]. Available: https://doi.org/10.1145/3243734.3243809

[290] S. Xie, M. Mohammady, H. Wang, L. Wang, J. Vaidya, and Y. Hong, "A generalized framework for preserving both privacy and utility in data outsourcing," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021 ©2021 IEEE. Reprinted, with permission, from Shangyu Xie, Meisam Mohammady, Han Wang and Lingyu Wang, Jaideep Vaidya and Yuan Hong, A Generalized Framework for Preserving Both Privacy and Utility in Data Outsourcing, IEEE Transactions on Knowledge and Data Engineering, 2021.

[291] S. Xie, M. Mohammady, H. Wang, L. Wang, J. Vaidya, and Y. Hong, "A generalized framework for preserving both privacy and utility in data outsourcing (extended abstract)," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022 ©2022 IEEE. Reprinted, with permission, from Shangyu Xie, Meisam Mohammady, Han Wang and Lingyu Wang, Jaideep Vaidya and Yuan Hong, A Generalized Framework for Preserving Both Privacy and Utility in Data Outsourcing (Extended Abstract), IEEE Transactions on Knowledge and Data Engineering, 2022, pp. 1549–1550.

[292] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 644–655. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813651

[293] G. Kellaris, G. Kollios, K. Nissim, and A. O'neill, "Generic attacks on secure outsourced databases," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1329–1340.

[294] M. Burkhart, D. Schatzmann, B. Trammell, E. Boschi, and B. Plattner, "The role of network trace anonymization under attack," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 5–11, 2010.

[295] J. King, K. Lakkaraju, and A. Slagell, "A taxonomy and adversarial model for attacks against network log anonymization," in *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM, 2009, pp. 1286–1293.

[296] I. A. Al-Kadit, "Origins of cryptology: The arab contributions," *Cryptologia*, vol. 16, no. 2, pp. 97–126, 1992. [Online]. Available: https://doi.org/10.1080/0161-119291866801

[297] T.-F. Yen, X. Huang, F. Monrose, and M. K. Reiter, "Browser fingerprinting from coarse traffic summaries: Techniques and implications," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, U. Flegel and D. Bruschi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 157–175.

[298] T. Brekne, A. Årnes, and A. Øslebø, "Anonymization of ip traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies," in *International Workshop on Privacy Enhancing Technologies*. Springer, 2005, pp. 179–196.

[299] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD

'11. New York, NY, USA: ACM, 2011, pp. 1082–1090. [Online]. Available: http://doi.acm.org/10.1145/2020408.2020579

[300] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, "Efficient genome-wide, privacy-preserving similar patient query based on private edit distance," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 492–503. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813725

[301] Y. He and J. F. Naughton, "Anonymization of set-valued data via top-down, local generalization," *PVLDB*, vol. 2, no. 1, pp. 934–945, 2009.

[302] Z. Huang, "Clustering large data sets with mixed numeric and categorical values," in *Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(PAKDD)*. Singapore, 1997, pp. 21–34.

[303] C. Chatfield, *The analysis of time series: an introduction*. Chapman and Hall/CRC, 2016.

[304] "Bing maps — microsoft learn," https://docs.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system, accessed: 2019-10-30.

[305] J. Pei, J. Han, and L. V. Lakshmanan, "Mining frequent itemsets with convertible constraints," in *Proceedings 17th International Conference on Data Engineering*. IEEE, 2001, pp. 433–442.

[306] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Acm sigmod record*, vol. 22, no. 2. ACM, 1993, pp. 207–216.

[307] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke, "Publishing search logs - A comparative study of privacy guarantees," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 3, pp. 520–532, 2012. [Online]. Available: https://doi.org/10.1109/TKDE.2011.26

[308] A. Rheinländer, M. Knobloch, N. Hochmuth, and U. Leser, "Prefix tree indexing for similarity search and similarity joins on genomic data," in *International Conference on Scientific and Statistical Database Management*. Springer, 2010, pp. 519–536.

[309] D. Riboni, A. Villani, D. Vitali, C. Bettini, and L. V. Mancini, "Obfuscation of sensitive data in network flows," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 2372–2380.

[310] F. B. Durak, T. M. DuBuisson, and D. Cash, "What else is revealed by order-revealing encryption?" in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1155–1166.

[311] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart, "Leakage-abuse attacks against order-revealing encryption," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 655–672.

[312] P. Grubbs, M.-S. Lacharité, B. Minaud, and K. G. Paterson, "Learning to reconstruct: Statistical learning theory and encrypted database attacks," 2019.

[313] J. Vaidya, B. Shafiq, X. Jiang, and L. Ohno-Machado, "Identifying inference attacks against healthcare data repositories," *AMIA Summits on Translational Science Proceedings*, vol. 2013, p. 262, 2013.

[314] X. S. Wang, Y. Huang, T.-H. H. Chan, A. Shelat, and E. Shi, "Scoram: Oblivious ram for secure computation," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 191–202. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660365

[315] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12.

[316] K. Chen, G. Sun, and L. Liu, *Towards Attack-Resilient Geometric Data Perturbation*, pp. 78–89.

[317] "cloudlab," https://docs.cloudlab.com/, accessed: 2019-06-20.

[318] E. Stefanov, M. V. Dijk, E. Shi, T.-H. H. Chan, C. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path oram: An extremely simple oblivious ram protocol," *Journal of the ACM (JACM)*, vol. 65, no. 4, p. 18, 2018.

[319] Z. Chang, D. Xie, and F. Li, "Oblivious ram: A dissection and experimental evaluation," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1113–1124, Aug. 2016. [Online]. Available: https://doi.org/10.14778/2994509.2994528

[320] J. Zhang, N. Borisov, and W. Yurcik, "Outsourcing security analysis with anonymized logs," in *2006 Securecomm and Workshops*, Aug 2006, pp. 1–9.

[321] W. Ding, W. Yurcik, and X. Yin, "Outsourcing internet security: Economic analysis of incentives for managed security service providers," in *Internet and Network Economics*, X. Deng and Y. Ye, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 947–958.

[322] R. Sion, "Secure data outsourcing," in *Proceedings of the 33rd International Conference on Very Large Data Bases*, ser. VLDB '07. VLDB Endowment, 2007, pp. 1431–1432. [Online]. Available: http://dl.acm.org/citation.cfm?id=1325851.1326036

[323] M. Zhou, Y. Mu, W. Susilo, J. Yan, and L. Dong, "Privacy enhanced data outsourcing in the cloud," *Journal of Network and Computer Applications*, vol. 35, no. 4, pp. 1367 – 1373, 2012, intelligent Algorithms for Data-Centric Sensor Networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804512000367

[324] O. Goldreich, "Towards a theory of software protection and simulation by oblivious rams," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '87. New York, NY, USA: ACM, 1987, pp. 182–194. [Online]. Available: http://doi.acm.org/10.1145/28395.28416

[325] M. T. Goodrich and M. Mitzenmacher, "Privacy-preserving access of outsourced data via oblivious ram simulation," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2011, pp. 576–587.

[326] E. Stefanov and E. Shi, "Oblivistore: High performance oblivious cloud storage," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 253–267.

[327] D. Cash, A. Küpçü, and D. Wichs, "Dynamic proofs of retrievability via oblivious ram," *Journal of Cryptology*, vol. 30, no. 1, pp. 22–57, 2017.

[328] M. Franz, P. Williams, B. Carbunar, S. Katzenbeisser, A. Peter, R. Sion, and M. Sotakova, "Oblivious outsourced storage with delegation," in *International Conference on Financial Cryptography and Data Security*. Springer, 2011, pp. 127–140.

[329] V. Vaikuntanathan, "Computing blindfolded: New developments in fully homomorphic encryption," in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, Oct 2011, pp. 5–16.

[330] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 506–522.

[331] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, ser. TCC '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 457–473. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00457-5_27

[332] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 79–88. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180417

[333] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Advances in Cryptology – CRYPTO 2013*, R. Canetti and J. A. Garay, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 353–373.

[334] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: Protecting confidentiality with encrypted query processing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, ser. SOSP '11. New York, NY, USA: ACM, 2011, pp. 85–100. [Online]. Available: http://doi.acm.org/10.1145/2043556.2043566

[335] F. Kerschbaum, "Frequency-hiding order-preserving encryption," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 656–667. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813629

[336] X. S. Wang, K. Nayak, C. Liu, T.-H. H. Chan, E. Shi, E. Stefanov, and Y. Huang, "Oblivious data structures," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 215–226. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660314

[337] B. Ribeiro, W. Chen, G. Miklau, and D. Towsley, "Analyzing privacy in enterprise packet trace anonymization," in *In Proceedings of the 15 th Network and Distributed Systems Security Symposium*, 2008.

[338] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation." in *Ndss*, vol. 20. Citeseer, 2012, p. 12.