University of Kentucky

UKnowledge

Theses and Dissertations--Electrical and Computer Engineering

Electrical and Computer Engineering

2016

Information-Theoretic Secure Outsourced Computation in Distributed Systems

Zhaohong Wang

University of Kentucky, zwa226@g.uky.edu

Digital Object Identifier: http://dx.doi.org/10.13023/ETD.2016.235

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Zhaohong Wang, Student

Dr. Sen-Ching S. Cheung, Major Professor

Dr. Caicheng Lu, Director of Graduate Studies

INFORMATION-THEORETIC SECURE OUTSOURCED COMPUTATION IN DISTRIBUTED SYSTEMS

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Engineering at the
University of Kentucky

By
Zhaohong Wang
Lexington, Kentucky

Director: Dr. Sen-Ching S. Cheung, Professor of Electrical and Computer Engineering
Lexington, Kentucky

2016

ABSTRACT OF DISSERTATION

INFORMATION-THEORETIC SECURE OUTSOURCED COMPUTATION IN DISTRIBUTED SYSTEMS

Secure multi-party computation (secure MPC) has been established as the de facto paradigm for protecting privacy in distributed computation. One of the earliest secure MPC primitives is the Shamir's secret sharing (SSS) scheme. SSS has many advantages over other popular secure MPC primitives like garbled circuits (GC) – it provides information-theoretic security guarantee, requires no complex long-integer operations, and often leads to more efficient protocols. Nonetheless, SSS receives less attention in the signal processing community because SSS requires a larger number of honest participants, making it prone to collusion attacks. In this dissertation, I propose an agent-based computing framework using SSS to protect privacy in distributed signal processing. There are three main contributions to this dissertation. First, the proposed computing framework is shown to be significantly more efficient than GC. Second, a novel game-theoretical framework is proposed to analyze different types of collusion attacks. Third, using the proposed game-theoretical framework, specific mechanism designs are developed to deter collusion attacks in a fully distributed manner. Specifically, for a collusion attack with known detectors, I analyze it as games between secret owners and show that the attack can be effectively deterred by an explicit retaliation mechanism. For a general attack without detectors, I expand the scope of the game to include the computing agents and provide deterrence through deceptive collusion requests. The correctness and privacy of the protocols are proved under a covert adversarial model. Our experimental results demonstrate the efficiency of SSS-based protocols and the validity of our mechanism design.

KEYWORDS: outsourced computation, multi-party computation, collusion, mechanism design

Author's signature: Zhaohong Wang

Date: June 11, 2016

INFORMATION-THEORETIC SECURE OUTSOURCED COMPUTATION IN DISTRIBUTED SYSTEMS

By
Zhaohong Wang

Director of Dissertation: Sen-Ching S. Cheung

Director of Graduate Studies: Cai-Cheng Lu

Date: June 11, 2016

ACKNOWLEDGMENTS

I would like to express my deepest appreciations to my advisor Dr. Sen-Ching Samson Cheung. This work would not have been possible without his countless hours of mentoring, discussion, reviewing, and editing. I am grateful for his excellent guidance and encouragement boosting my confidence to explore and develop new ideas. I would also like to express the gratitude for my PhD advisory committee members, Dr. Henry Dietz, Dr. Kevin Donohue, and Dr. Andrew Klapper, for providing valuable insights and provoking suggestions.

I would like to thank the Department of Electrical and Computer Engineering at the University of Kentucky, for the financial support and research experience during my graduate study. I am also grateful for the teaching experience in the Department. Thanks also go to the faculty, staff and fellow graduate students who continually offered support, advice, and assistance.

Finally, I am indebted to my parents for their endless love and encouragement, and my wife for her support and great help.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

While not explicitly stated in the U.S. Constitution, the rights of privacy for many aspects of our lives including religious beliefs, personal possession, and personal information are protected under the Bill of Rights. Nonetheless, news about different forms of privacy invasion has become a daily affair. From sale of personal information to identity theft, from Google and YouTube surrendering user data to the mining of phone metadata by the National Security Agency, the number of ways that our privacy can be invaded seems to increase at an alarming rate.

One of the reasons for such erosion is the significant advancement in computing technologies for collecting, storing, and sharing personal information among individuals, private sectors, and government agencies. Anyone can now carry thousands of songs, hundreds of pictures and hours of videos in a small smart phone, ready to be exchanged, sometimes unknowingly, with anyone in the world. The focus of privacy protection often falls on medical or financial records, but it is the multimedia signals – audio, images, and videos – that are driving the entire market of distributed computing while their privacy implications remain poorly understood.

The threats, however, are real. The advance in pattern recognition algorithms such as searchable surveillance or automatic speech recognition systems have turned the once labor-intensive processes into powerful automated systems. They can easily recognize objects of interest like faces, voice, and other biometric signals with high fidelity. Correlating such information with location data such as geo-tags or RFID and other information on social networks allows hackers to easily track activities and associations of any individuals. The matter is further complicated by the unprecedented effort of the government in monitoring activities of private citizens to fight terrorism. It is thus imperative to develop a comprehensive privacy protection framework for personal multimedia data without jeopardizing our homeland security.

While new legislature and policy changes are essential elements of such a framework, technologies are playing an equally pivotal role in safeguarding private information. There are two main technical challenges in protecting multimedia data. First, as signal capturing devices and wireless networks become ubiquitous, diverse applications from multimedia emails and blogging to large-scale surveillance networks begin to demand some forms of privacy protection. A comprehensive framework is needed to identify appropriate sensitive information in different applications, and to provide different levels of protection depending on the role of each user in the system.

Second, from simple enhancement to sophisticated pattern recognition, multimedia data requires various signal-processing operations to become useful. The current cloud and peer-to-peer (P2P) computing platforms have provided ubiquitous data storage for multimedia data. The computation by third parties like those cloud and P2P platforms is called the *outsourced computing*. Cloud and P2P computing platforms can process large-scale of data from multiple sources in a distributed manner. The promise of these computing platforms is grounded on its predicted pervasiveness: Internet customers will contribute their individual data and get useful services from the computing platforms. Figure 1.1 illustrates the situation of customers utilizing third parties to process their data, where customers are denoted as User $1, 2, .., N$ and the computing agents are operated in either cloud or P2P platforms.

There are a myriad of applications of outsourced computing. Examples include cloud storage and back up services, like Amazon Simple Storage Service (Amazon S3) [4], Dropbox [45], MEGA [90], and Carbonite [25]. In the United States of America, the Affordable Care Act (ACA) leverages cloud-based solutions heavily [85]. Due to the fact that human DNA sequences are very large data sets, computing personally tailored drugs and personalized medical treatments according to an individual's DNA sequence requires huge computing resources which are obtainable from the cloud plat-

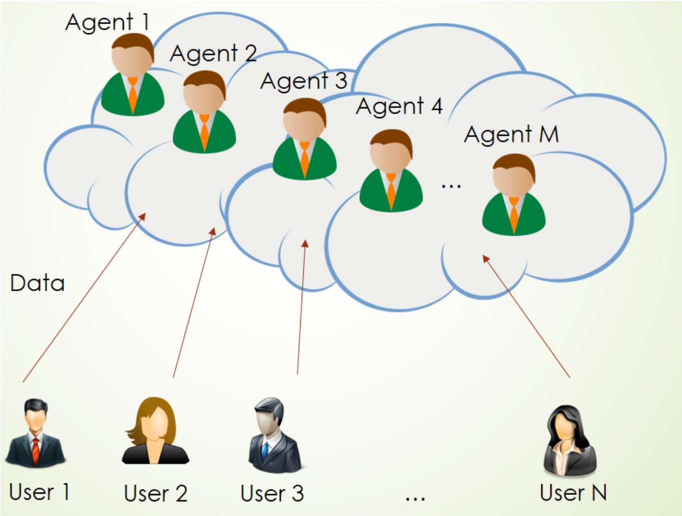

Figure 1.1: Outsourced Computing Scenario

form. A mobile application was presented for the electronic healthcare data storage, update and retrieval using Cloud Computing [44]. GraphLab, a framework for machine learning and data mining in the cloud, was proposed and implemented [86]. To achieve a high quality of service (QoS) provisioning for multimedia service, a media-edge cloud (MEC) architecture was proposed [140].

1.1 Application Scenario and Privacy Concerns

In the future, software developers will undoubtedly take advantage of the enormous power of various distributed computing platforms in offering various types of software services to process data. We consider a distributed computing task to have at least two participants: a user with a private input requires a service from a vendor with a proprietary software algorithm. It is easy to see why privacy is needed in any distributed multimedia processing service platform – the user may want to enhance or process a video taken using a smart phone but lacks the required capability and algorithms. The multimedia software vendor offers proprietary software for the job and charges the user based on the duration of the video needed to be processed. For simplicity, we assume that the vendor's secrets are the key parameters used in an otherwise well-known algorithm. For example, the parameters could be the tap values of a sophisticated filter or thresholds and weights in a neural network. The privacy objective is as follows: the user and the vendor do not trust each other and would like to prevent each other from knowing anything about their own secret information. The two parties come to the cloud server and take advantage of its enormous power and storage. The user clearly wants to protect any private information in his/her video while the software vendor needs to prevent theft of its proprietary algorithm.

Outsourced computing can fulfill its promise only if it provides a wide range of computation tasks while guaranteeing security and privacy for customers' input data. The key challenge is to design an appropriate privacy protection scheme so that it will not compromise any legitimate processing of the data.

Current privacy violation in the outsourced computing is serious. From time to time, there are reports of cloud platforms being hacked and customers' sensitive data compromised. Cloudminr.io, a Bitcoin Cloud Mining service, has been hacked. Its whole database of users is on sale for 1 Bitcoin [98]. In June 2015, LastPass, the cloud-based password manager, announced that its network was hacked and sensitive user information was stolen [121]. The concerns of patient privacy and information security of the ACA remain high [49]. It will be a disastrous result when the ACA medical records stored in the cloud are breached. If people's private medical records are exposed, victims with some types of diseases may be denied jobs.

1.2 Provable Security Approaches

To overcome the concerns of privacy while utilizing powerful computing resources from third party platforms, there are two mainstream approaches for distributed computations on sensitive data with provable security and privacy. They are the differential privacy framework and the secure multi-party computation (secure MPC) framework [46, 34]. In the sequel, we provide a brief review of each framework and evaluate their appropriateness as an outsourcing platform for multimedia data.

Differential Privacy

There is a strong need to quantify privacy in the outsourced scenario. Contributors of data to third-party platforms face several threats to their privacy [114]. Take a person participating in a large-scale health care study for example. First, the patient's data may be recorded and exposed by a malicious application as part of the study. The malicious application could stealthily write the stolen data into a file and put it on the Internet. Then the file could be indexed by a search engine and thus further exposed to the public. Second, even if all computations are done correctly and securely, the final result itself, such as those common statistics computed in the study, may leak sensitive information about the patient's personal medical record. To address the aforementioned concerns, traditional approaches to data privacy are to sanitize the data by syntactic anonymization, in which personally identifiable information such as names, ages, and Social Security numbers are removed. Unfortunately, the approach of anonymization does not provide provable privacy guarantees. For example, public releases of anonymized individual datasets, including AOL search logs [1] and the movie rating records of Netflix subscribers [95], eventually reveal customers' private information. They are evidence that naïve anonymization was easy to reverse in many cases. These incidents motivate a new approach called diferential privacy to protecting data privacy quantitatively [114].

Differential privacy was proposed for the purpose of enabling systems to draw inferences from datasets while preserving the privacy and security of the data and individual identities. Suppose there are n records in a dataset $\mathcal{D} = (\mathbf{x}_1, ..., \mathbf{x}_n)$, where each $\mathbf{x}_i$ is a vector in $\mathbb{R}^d$ and represents the data collected from an individual i. The number of elements, d, in $\mathbf{x}_i$ can be understood as the d features of an individual. Differential privacy is defined as follows [118].

Definition 1.2.1. An algorithm $\mathcal{A}_{priv}(\cdot)$ taking values in a set $\mathcal{T}$ provides ϵ-differential privacy if

$$\mathbb{P}(\mathcal{A}_{priv}(\mathcal{D}) \in \mathcal{S}) \leq e^{\epsilon} \cdot \mathbb{P}(\mathcal{A}_{priv}(\mathcal{D}^{'}) \in \mathcal{S}) \tag{1.1}$$

for all measurable $\mathcal{S} \subseteq \mathcal{T}$ and all datasets $\mathcal{D}$ and $\mathcal{D}^{'}$ differing in a single entry. The algorithm $\mathcal{A}_{priv}(\cdot)$ provides (ϵ, δ)-differential privacy if

$$\mathbb{P}(\mathcal{A}_{priv}(\mathcal{D}) \in \mathcal{S}) \leq e^{\epsilon} \cdot \mathbb{P}(\mathcal{A}_{priv}(\mathcal{D}^{'}) \in \mathcal{S}) + \delta \tag{1.2}$$

for all measurable $\mathcal{S} \subseteq \mathcal{T}$ and all datasets $\mathcal{D}$ and $\mathcal{D}^{'}$ differing in a single entry.

Privacy parameters are ϵ and δ. The smaller ϵ and δ are , the more privacy $\mathcal{A}_{priv}(\cdot)$ would ensure [47, 135]. The computation under the differential privacy framework is in the interactive query model. There are two roles in the computation, one is the user and the other is the curator of the database $\mathcal{D}$. The user submits queries to the curator and the latter replies approximate answers.

For the application of the differential privacy framework in signal processing problems, there has been some progress recently. Rastogi and Nath proposed the first differential private data aggregation algorithm for distributed time-series data that offers good practical utility without relying on any trusted server [110]. To ensure differential privacy while overcoming the poor performance for time-series data by standard differential privacy techniques, they proposed the Fourier Perturbation Algorithm. FAST, an adaptive system to release real-time aggregate statistics under the differential privacy framework, was proposed using Kalman filter to improve the accuracy of data release per time stamp [51].

However, there are some open problems in the area of differential privacy that makes it inappropriate for the application scenario described in Section 1.1. First, the differential privacy framework provides approximate results [118], which is not satisfactory in many signal processing tasks. Second, the choice of the privacy parameters ϵ and δ has little consensus. While there is heuristics on choosing ϵ, the problem of choosing δ for the (ϵ, δ) differential privacy defined in 1.2 is poorly understood [48, 54]. Third, there are no differential privacy systems developed for the joint computation carried out by distrusted parties using their respective private inputs. Despite the fact that Proserpio et al. proposed distributed algorithms for publication for graph topologies [108], and McSherry and Mahajan considered the network trace problem [89], research on the networked information systems in differential privacy is still at its early stage. Thus, the general scenario in Section 1.1 is not yet ready to be deployed in the differential privacy framework.

Our focus is on two or more data owners computing jointly to obtain precise answers. While the differential privacy framework lacks mature joint computation solutions and provides only approximate results to computing tasks, the secure MPC framework provides exact outputs. Hence, we are most interested in the approach of secure MPC.

Secure Multi-Party Computation

Privacy protection in distributed computing enables distrusting parties to participate in joint computation without revealing their secret data. For example, a small company can process its confidential customer data using a proprietary software from a vendor. The standard approach to protect the secrecy of data from all parties in a joint computation is to use secure multiparty computation or secure MPC protocols.

The goal of secure MPC is to enable multiple distrusted parties, in the absence of any trusted third parties, to jointly compute a pre-agreed functionality based on each party's private input. Many protocols for secure MPC have been proposed since 1982 [13]. There has been work on various secure MPC protocols for image denoising [115], joint computation among social network members [58], linear programming [127], auction [22], etc.

The simplest form of a secure MPC is a two-party secure computation shown in Figure 1.2. The model has two participants, User and Vendor, each has his/her secret input. They encrypt their respective input using some secure MPC technique, with the most commonly-used one being the homomorphic encryption (HE) [103] and the garbled circuits (GC) [138]. Hence, the information exchanged between the two parties does not disclose any information about the secrets. This is the model used in encrypted-domain techniques. Data processing operations are to be performed directly on encrypted signals.

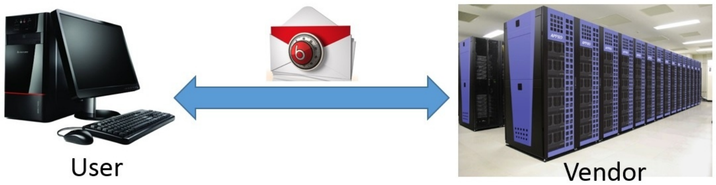

Figure 1.2: Two-party Secure Computation Scenario

Nevertheless, despite being actively researched for more than 30 years, and significant advancement in the last few years [83, 123, 77], secure MPC protocols are still rarely used in practical systems since its invention by Yao [138]. The only reported large-scale application of secure MPC is the Danish sugar beet auction in January 2008 where over 1200 farmers and three servers participated [22]. The execution of the secure MPC ensured that each bid from a farmer submitted to the auction was kept secret from the time it left the bidder's computer. The goal of the secure MPC system is to efficiently compute the price at which contracts should be traded without direct access to the actual bidding prices.

One key reason of the lack of practical secure MPC systems is the high computational costs secure MPC protocols required, especially among those that rely on encrypted-domain processing. These protocols, classified as computationally secure multi-party computation (**CS-MPC**) protocols, achieve their security against any adversary limited to polynomial computing time. The encrypted-domain algorithms compute the desired output using only the encrypted input signals. Such techniques have garnered much attention in recent years due to their potentials in various security and privacy applications such as biometric matching and privacy-preserving data mining [106, 82].

The key challenge facing most encrypted-domain DSP systems is the significant increase in the computational complexity. For example, Yao's garbled circuits [138] with its variants is one of the most popular approaches for encrypted-domain DSP. The best result to-date on the performance of garbled circuit is about $1.6\mu s$ per garble gate or 625000 gates per second [83]. This is minuscule compared to most modern micro-processors, where even the slowest ones can easily execute billions of instructions per second.

There are two reasons for the enormous gap between computation in plaintext and in ciphertext: first, the protection of ciphertext relies on the computational hardness of certain mathematical problems. To ensure even a short-term protection against adversaries, a large security parameter needs to be used which results in a hundred to a thousand-fold increase in the data size. Second, the particular choice of cryptographic primitives used in the system strongly affects the implementation complexity of various basic operations. While the performance of certain operations can be optimized, most other basic operations require more complex procedures and interaction with secret owners.

An alternative is to use information-theoretic secure MPC (ITS-MPC) protocols. In these protocols, information exchanged between different parties are statistically independent of the secret data. As ITS-MPC protocols do not depend on the hardness of specific computational problems, they often admit faster implementations using a smaller prime field for data representations [134, 116, 42]. Typical baseline protocols for ITS-MPC include linear secret sharing [84], additive secret sharing [21], and Shamir's secret sharing (SSS) [119]. Secret sharing schemes are schemes that decompose a secret number into multiple shares distributed to different parties. Individual shares should not reveal any information about the secret but can be combined to recreate it.

SSS is among the earliest and most commonly-used ITS-MPC primitives. In SSS, secret information is first decomposed into multiple random shares using a polynomial secret sharing scheme. Similar to garbled circuits and homomorphic encryption, SSS is also homomorphic - computation with respect to addition and multiplication on secret numbers can be carried out through manipulation of the shares. Compared with encrypted-domain approaches, SSS does not require long keys and its implementations of basic arithmetic operations are usually much simpler. Unlike those encrypted-domain approaches, protocols based on SSS can achieve information-theoretic security – they guarantee security against an adversary with infinite computing power. Direct computation on shares is ideally suitable for privacy and security enhancement of high data-rate applications such as multimedia processing and data mining. It has been applied in many areas from typical privacy-enhanced applications such as auction [22] and private information retrieval [42], to signal-processing applications including medical data visualization [91], image processing [116] and video surveillance [126].

A drawback of SSS is the need to maintain a fraction of the computational parties non-colluding [16]. Specifically for SSS, computation is only feasible with at least three non-colluding semi-honest shareholders. Researchers have long pointed out the danger of collusion attacks in outsourced computation [40]. In fact, these attacks are significant real-life problems and occur in many networked applications. In online Poker and P2P file sharing, cheaters collude to have advantage over other honest players [59, 80]. *PokerStars*, one of the the largest online poker cardrooms in the world, recruits special security personnel to manually investigate special play patterns to uncover collusion patterns [59], and prohibits players from the same country to be in the same game [5]. Colluding communication usually exists in two different forms:

it can occur in side-channels external to the protocols, or as hidden data within, otherwise known as subliminal communication [2]. Algorithmically, it is impossible to design protocols to curtail communications over unknown side-channels. Existing anti-collusion techniques focus on eliminating subliminal communication by relying on either a semi-honest/trusted centralized server [2, 3] or a specially-designed ballot box [73]. These require heavy computation at a fortified centralized server, which defeats the efficiency goal of using SSS based ITS-MPC techniques.

1.3 Contributions

In this dissertation, our main contribution is the collusion deterrence for the ITS-MPC framework. We propose a collusion-deterred outsourced computing platform using SSS as a building block. This platform consists of a set of computing agents that provide computing services for secret-data owners to collaborate on joint computation using secret shares. Collusion, modeled as a covert adversarial behavior, can occur between one of the data owners and a portion of computing agents, or among the computing agents themselves. Unlike existing approaches, we model collusion as games and propose various mechanism designs that lead to honesty as stable strategies. The advantage of our approach is that no centralized server or computationally-intensive protocols are needed, making our solutions ideal for high-throughput signal processing applications. Our treatment is also comprehensive because the proposed models can handle different types of collusion attacks including those that may not be detected at all by the participants. In this dissertation, we further apply Bayesian games to better model the uncertainty in data owners' privacy preference, propose a new censorship scheme to thwart collusion among computing agents, and provide additional experimental results to show the efficiency of SSS-based algorithms over other state-of-the-arts CS-MPC protocols.

1.4 Dissertation Organization

This dissertation is structured as follows. In Chapter 2 we present the information of related work for secure MPC, collusion deterrence, and secret sharing. In Chapter 3, we summarize general operations of signal processing in the secret sharing domain, specifically, using the Shamir's secret sharing (SSS). We also describe the secure MPC based on SSS in terms of its security model, its classification according to the computational power of the adversary, and the computational techniques for building arithmetic operations. To demonstrate the usefulness of secret sharing domain signal processing, we propose an application of image denoising through wavelets transform. In Chapter 4 we discuss the collusion attacks under our proposed framework. In Chapter 5 we propose and analyze our game-theoretic mechanism designs for the collusion deterrence between customers. In Chapter 6 we describe analyze further mechanism designs as well as a censorship scheme for the collusion deterrence for computing agents. Experimental results are presented in Chapter 7. Finally, Chapter 8 concludes this dissertation and points out future directions.

Chapter 2 Related Work

In this chapter, we provide a review for the area of secure MPC. We survey existing tools for both computational secure MPC (CS-MPC) and information-theoretic secure MPC (ITS-MPC). After that, we summarize previous efforts on counter-measuring collusion attacks in secure MPC protocols. Then we provide an overview of existing game-theoretic perspectives on secure MPC, which are relevant but different from our approach. Finally, we finish our review with a discussion of secret sharing schemes.

2.1 Secure MPC Review

The study of secure multiparty computation started in the 1980s [138]. There are four main types of secure MPC primitives: Garbled Circuits (GC) [138], Homomorphic Encryption (HE) [111, 57], GMW protocol [61], and BGW protocol [12]. GC is one of the most well-developed secure MPC protocols. It provides a generic two-party implementation of any binary function by having one party prepare an encrypted Boolean circuit, and another party evaluate the circuit using the public-key based oblivious transfer protocol [75]. In recent years, there have been significant advances in optimizing the basic GC protocols [70, 77, 123], and in developing novel efficient hybrid circuits [83, 97].

Homomorphic encryption is a special kind of public-key encryption where some algebraic manipulations on plaintext numbers can be realized in the encrypted domain. Earlier homomorphic encryption such as RSA [112], ElGamal [50], and Paillier [104] are only partially homomorphic. Fully homomorphic encryption or FHE was proposed in [57] but remained highly complex and impractical for realistic computations [92]. Nevertheless, recent work [39, 23] has demonstrated efficient implementations of small circuits using the so-called somewhat homomorphic encryption (SHE).

Compared with the original plaintext computation circuit, the reliance on public-key protocols like oblivious transfer and homomorphic encryption significantly increases the size of the encrypted circuit. Instead of public-key based representations, the protocol proposed by O. Goldreich, S. Micali, and A. Wigderson, or the GMW protocol [62], and the protocol by M. Ben-Or, S. Goldwasser, and A. Wigderson, oe the BGW protocol [12], use secret sharing schemes such as Shamir's Secret Sharing (SSS) scheme [119] to protect the secrecy of the operands. While GMW still requires oblivious transfer to protect against malicious adversaries, the BGW protocol is free of any expensive public-key operations and can use any prime field to achieve information-theoretic security (ITS). The efficiency of the BGW protocol makes it the basis of some ITS-MPC platforms including FairplayMP [11] and Sharemind [21], the latter being used by the Estonian Government in studying the linkage of tax and education records [20]. The focus of this dissertation is on building the application framework for the BGW protocol, specifically with the SSS building block, for signal processing.

The use of secure MPC in signal processing has a late start due to the substantial challenges in adapting the complex protocols to handle the high data rate and real-time response demanded by typical signal processing applications. One of the earliest projects, SPED (Signal Processing in the Encrypted Domain) and its follow-on research were products of joint efforts between the applied cryptography and signal processing communities, resulting in a number of homomorphic encryption based implementations of fundamental algorithms such as Fourier Transform and filtering [41, 107, 18, 19]. Since then, there have been many other privacy enhanced applications from audio processing to biometric matching developed using both GC and HE. Interested readers should consult survey papers in this area such as [87, 52, 105]. Recently, efficient SSS-based protocols started to emerge in different signal-processing applications including medical data visualization [91], image denoising [116], video surveillance [126], and outsourced image enhancement [78].

2.2 Adversarial Model and Collusion Attacks in ITS-MPC Protocols

A key weakness of the BGW protocol is its susceptibility to collusion attacks. In SSS, the secret can be reconstructed using shares from a subset of the computational parties. The smallest size of such a subset is called the threshold of the SSS scheme and it has been shown that the threshold cannot be smaller than half of the total number of parties [12]. A collusion attack occurs when an adversary with malicious intent controls the threshold or more parties in a joint computation. In the most general case, when parties are connected by pairwise communication channels, collusion-free protocols for computing non-trivial functions are impossible [3]. For our target application of outsourced computation, we provide a detailed analysis of different collusion attack scenarios in Section 4.2. Typical cryptographic approaches to counter collusion include the use of centralized trusted mediators [2, 3] and complex cryptographical constructions [12, 73]. These computationally intensive constructions are incompatible with our efficiency goal in applying secure MPC techniques for high-rate distributed signal processing applications.

Our focus on solving collusion attacks in outsourced computation stems from its significant differences from other types of adversarial attacks. General consideration of attacks on secure MPC protocols include two different aspects: 1) how parties are identified and corrupted by an adversary, and 2) how corrupt parties behave under the control of an adversary. For the first aspect, existing classifications typically focus on whether an honest party becomes corrupted during the course of the computation [67]. For collusion attacks, this question is less important than how an honest party is corrupted by an adversary to collude in stealing a secret - is it due to information received in-band as part of the defined protocol or out-of-band through side channels? The first kind is termed a local adversary attack as it is based on local information anticipated by the protocol. The local adversarial model has been used to model collusion attacks before [24] and will be assumed in our protocol design.

As for the second question, adversarial behaviors are typically classified into semi-honest and malicious. Semi-honest refers to behaviors that do not deviate from the protocols but attempts to extract useful information from received data. The SSS protocols described in Section 4.1 are secure under a semi-honest model. The semi-honest model is quite limited in practice - it certainly cannot capture an attempt of a corrupted party to persuade others to collude. The other end of the spectrum is the malicious model which puts no restriction on the behaviors of a corrupted party, which can disrupt and terminate the protocol at any time. Classical feasibility results have already demonstrated that IT security can be achieved by an addition of a broadcast channel and the use of verifiable secret sharing [109]. These additions, however, significantly limit their applications due to a much higher computation complexity.

More importantly, the malicious model does not model colluding activities adequately. First, in the presence of side channels, collusion can occur entirely out-of-band and it is impossible to counter using only protocol design. Second, it does not address the dynamic nature of collusion in which an honest participant agrees or refuses to collude. Assuming that there is a rational being behind each participant, there must be an external reason behind this decision such as a higher reward or a non-negligible probability of getting caught. Such behaviors can be modeled with a covert adversarial model in which a malicious attack can be deterred if the adversary can be caught with high probability [8]. Many work have used this security notion to design protocols that are more practical and efficient than those based on the notion of malicious adversary. They achieve computational security using the building blocks of oblivious transfer and homomorphic encryption [96, 7]. In [37], the authors showed a general procedure to compile a protocol robust against the passive adversary into a protocol against the covert adversary with cheating behavior being caught at a probability of $\frac{1}{4}$ or higher. Their benchmark on an AES cipher demonstrated a significant performance enhancement over the baseline protocol that is secure against the malicious adversary [38].

2.3 Game-Theoretic Analysis for Secure MPC

More fundamental to the protocol design is to address the rationality behind adversarial behaviors. The use of game theory nicely captures the rationality of adversaries in many commercial, political, and social settings: weighing the gain of cheating against the risk and loss of being caught [66, 120]. The seminal work on rational secret sharing and secure MPC by Halpern and Teague first took into account the rational decision making process of the participants [65]. Since then, there has been several work focusing on various aspects of rational secure MPC including mixed-behavior models [88], rational ITS-MPC over broadcast channels [76], and computationally efficient rational secret sharing [53]. Other work attempts to recast the secure MPC problem from a game-theoretic perspective. In [6], the authors investigated two-party computation in the context of a fail-stop game and extended the notions of privacy, correctness, and fairness through equilibrium definitions from game theory. Recently, rational secret sharing has also been extended to incorporate perfect Bayesian equilibrium to model imperfect designs [131].

However, the problem addressed by the work[6, 131] is different from ours. They provide solutions on how to encourage all agents to honestly carry out the computation so that the agents themselves can benefit from knowing the final answers. Such formulation is not suitable for the computation outsourcing scenarios that we are addressing. For outsourcing, the computational agents have no stake on the actual secrets as they are simply carrying out a pre-defined computation in exchange for some kind of reward. Also, the active involvement of an agent in a collusion attack does not necessarily disrupt the computational process and the customers can still obtain the correct final results. As a result, our work primarily focuses on the mechanism designed to cope with the heterogeneous nature of the games involving both customers and computational agents as well as imperfect knowledge behind their utility functions.

2.4 Secret Sharing Schemes

A secret sharing scheme serves as a procedure that involves two types of roles: the secret owner and the share holding parties. The secret owner acts as a dealer to distribute shares to parties so that only authorized subsets of parties can reconstruct the secret. Secret sharing schemes are important building blocks in cryptography and they are used in many secure protocols. They have found a broad range of important applications. For example, general protocol for multi-party computation [14], Byzantine agreement [109], threshold cryptography [56, 74], access control [93], attribute-based encryption [63], and generalized oblivious transfer [71, 124].

Here we briefly review some of the most interesting constructions of secret sharing schemes that are linear, that is, the distribution scheme is a linear mapping. We are most interested in Shamir's Secret Sharing (SSS), which is the building block for the system proposed in this dissertation. It enjoys the property of homomorphism – that addition and multiplication can be carried out directly on the shares [16]. SSS belongs to the category of threshold secret sharing schemes, where the authorized sets, or the access structure, are all sets whose size is bigger than some threshold. A (t, n) SSS has the threshold being t and there are a total of n parties. Both t and n are integers and $1 \leq t \leq n$. The access structure can be defined as $\mathcal{A}_t = \{B \subseteq \{p_1, ..., p_n\} : |B| \geq t\}$, where $p_i, i \in \{1, ..., n\}$ is the $i-$th party. The secret is hidden as the constant term of a polynomial with random coefficients chosen by the secret owner. The shares are evaluated at different points of the polynomial. A detailed review and discussion of SSS can be found in Section 3.1. Benaloh and Rudich constructed a secret-sharing scheme that has its access structure corresponding to edges of a complete undirected graph with m vertices $v_1, ..., v_m$. A secret bit k is shared into $(m-2)$ bits. Each of the $(m-2)$ parties will have a bit calculated based on the generated bits [10]. Ito et al. defined secret sharing schemes for general access structures. They showed the construction of such schemes for every monotone access structure. Let $\mathcal{A}$ be any monotone access structure. The secret owner shares the secret independently for each authorized set $B \in \mathcal{A}$ [72]. Generalizing the construction of [72], Benaloh and Leichter proposed a construction of secret-sharing schemes for any monotone formulae access structure. The length of the shares becomes exponential in the number of parties [15]. There is another category of secret sharing used for visual cryptography. It allows secret visual information to be embedded and encrypted in a cover image such that a well defined subset of the shared cover image could be overlapped to reconstruct the secret visual information [30].

Chapter 3 Encrypted Domain Signal Processing

In this chapter, we extend the basic homomorphic properties of Shamir's sharing scheme to handle common signal processing operations such as filtering and thresholding. Based on these building blocks, we design an information-theoretically secure protocol for distributed wavelet denoising. The operational setting of the protocol described in Section 1.1 is repeated here as follows: a user U has an image which he/she wants to keep private but requires denoising. A software provider called V the Vendor holds a proprietary wavelet filter and a denoising threshold. After decomposing their secret data into random shares, the shares are transmitted to three computing parties in a network which provides point-to-point secure communication. We assume that all three parties do not collude on their shares, and we will demonstrate that the received data contain no information about the image nor the filter. Once the computation is complete, the resulting shares are transmitted back to the user to reassemble the denoised image. The key contributions of our work include the development of information-theoretic secure signal processing building blocks with secret shares and the demonstration of a realistic image processing algorithm using the proposed framework. The organization of the Chapter is as follows: Section 3.1 reviews the Shamir's secret sharing scheme and defines notations used in the rest of the chapter. Section 3.2 shows how fundamental operations for signal processing can be implemented with secret sharing. Section 3.3 describes our secure image denoising protocol and analyzes its complexity. We provide implementation details and experimental results in Section 3.4, and conclude the chapter in Section 3.5.

3.1 Shamir's Secret Sharing

Let x be a number in a prime field F_m with m prime. Let n be the number of parties and t, called the threshold, be a positive integer between 1 and n. A (t, n) secret-sharing scheme of a secret number x produces n shares $[x]_i^t, i = 1, 2, \ldots, n$ such that any group of t or more shares can be used to reconstruct x. Any group of less than t shares, however, provides no information about x. Shamir's secret sharing scheme hides the secret as the constant term of a random degree $(t-1)$ polynomial. The polynomial coefficients α_j's are uniformly random numbers selected by the secret owner. The secret owner generates the i^{th} share by evaluating the polynomial at a public constant k_i which usually is agreed to be the identity number of the i^{th} party:

$$[x]_i^t \triangleq \sum_{j=1}^{t-1} \alpha_j k_i^j + x \bmod m. \tag{3.1}$$

The fact that α_j's are uniformly random in a prime field implies that the shares must also be uniformly random, thereby providing no information about x. Given at least t shares, the secret number x can be reconstructed with Lagrange interpolation

$$x = \sum_{i \in K} \gamma_i [x]_i^t \bmod m \tag{3.2}$$

where $\gamma_i \triangleq \prod_{j \in K, j \neq i} \frac{-k_j}{k_i - k_j}$ and K is any subset of $\{1, \ldots, n\}$ with at least t elements.

Let $x, y \in F_m$ be secret numbers and $a, b \in F_m$ be constants. The following properties of Shamir's scheme are well known [16]:

(P1) $[x + a \bmod m]_i^t = [x]_i^t + a \bmod m$

(P2) $[ax \bmod m]_i^t = a[x]_i^t \bmod m$

(P3) $[x + y \bmod m]_i^{\max(s,t)} = [x]_i^s + [y]_i^t \bmod m$

(P4) $[xy \bmod m]_i^{(s+t)-1} = [x]_i^s [y]_i^t \bmod m$

(P5) Assume $x, y \in \{0, 1\}$ and $\oplus$ denotes xor.

$$[x \oplus y]_i^{(s+t)-1} = [x]_i^s + [y]_i^t - 2[x]_i^s[y]_i^t \bmod m$$

(P6) $[x]_i^t = \sum_{j=1}^n \gamma_j [[x]_j^{(s+t)-1}]_i^t \bmod m$

P1 through P5 form the foundation of computation in secret shares – they show that performing certain operations on each share of secret numbers is equivalent to applying those operations first on the secret numbers and then creating the shares. These operations include addition and multiplication with constants, with other secret numbers, and exclusive-or on secret bits. These operations are universal in the sense that any computation on a digital computer can be composed by successive applications of these fundamental operations. Since the original shares do not reveal any information about the secret numbers, no successive operations on the shares can gain further knowledge. At the end of the operations, the secret owner can collect enough shares to reveal the result.

There are however hidden communication cost associated with some of these operations. Multiplication and exclusive-or operations (P4 and P5) produce results in a sharing scheme with a higher threshold $(s + t) - 2$, where s and t are the original thresholds of the two secret operands. Repeated applications of such operations will eventually arrive at a threshold larger than the number of parties n and the final result cannot be reconstructed even if all the shares are available. We can solve this problem by renormalizing the threshold using P6: each party further breaks his/her share into separate shares and sends a share to its corresponding party. The final share at each party is computed by a weighted summation of these newly received shares from other parties. Since each party receives only one share from any other party, no secret information is leaked. This threshold reduction requires $n(n - 1) \log m$ bits to be exchanged among the n parties. Denote computing parties as P_k for $k = 1, ..., n$. To highlight the communication between two computing parties P_i and P_j, we use the arrow notation "$\longrightarrow$":

(P6') $P_j : [[\mathrm{expr}(x)]_j^q]_i^t \longrightarrow P_i : [x]_i^t$ for $i \neq j$

The $\mathrm{expr}(\cdot)$ operator in P6' can include a composition of different operations which increase the threshold to be $q > t$ that may result in one or more steps of renormalization. After those steps of renormalization, the new share will again has the threshold to be t.

An obvious omission from the above properties is division between two secret numbers. To compute $xy^{-1} \bmod m$, y^{-1} must exist in F_m. We denote the inverse operation as follows:

(P7) $\mathrm{INVERSE}\,([y]_i^t \text{ all } i) \longrightarrow P_i : ([y^{-1}]_i^t)$

INVERSE can be implemented by repeated multiplications according to the Carmichael's theorem [113]: $y^{-1} = y^{\lambda(m)-1} \bmod m$, where $\lambda(m)$ is the (reduced) totient function. Notice that with this equation, the inverse of 0 is defined and is equal to 0. For prime m, $\lambda(m) = m - 1$. For large $\lambda(m) - 1$, the inverse operation is expensive as every multiplication requires a renormalization step. To reduce the number of multiplications, we can first express $\lambda(m) - 1$ as a sum of powers of two, say $\lambda(m) - 1 = 1011$ base 2 which implies $y^{\lambda(m)-1} = y^4 y^2 y$. We can then recursively compute $[y^2]_i^t$ and $[y^4]_i^t$ before multiplying them together to get the final answer. The communication complexity will be $O(\log \lambda(m))$ rather than $O(\lambda(m))$ in the sequential multiplication.

3.2 Signal Processing on Random Shares

Armed with the basic properties, many commonly used signal processing operations can be implemented. For example, in a linear convolution operation, two parties holding a secret signal $x(t)$ and a filter $h(t)$ can create n shares independently and distribute them to n parties to perform a privacy-protected linear filtering:

$$\left[\sum_{\tau} x(t-\tau)h(\tau)\right]_i^{2t-1} = \sum_{\tau} [x(t-\tau)]_i^t [h(\tau)]_i^t \tag{3.3}$$

In this section, we survey a set of state-of-art arithmetic operations implemented in the secret sharing domain. These protocols can form the toolbox for privacy protecting signal processing tasks. These protocols are designed for integer and fixed point numbers. They achieve perfect privacy or statistical privacy.

With the domain and range in a prime field $\mathbb{Z}_m$, a function can be represented in the form of an arithmetic circuit or in the form of a Boolean circuit. While the basic addition and multiplication of shares can be computed in an arithmetic circuit, non-linear operations such as equality test and comparison are more easily done in the Boolean circuit. Thus a first step to process shares will be to decompose them into bit representations. To simplify our notation for a share, we drop the subscripts i and t which stand for the i^{th} party and the threshold. Suppose a secret x has l bits. The share of x is denoted by $[x]$. The bitwise sharing of x is denoted as $[x]_B$. The goal is to get $[x_{l-1}], ..., [x_0]$ which satisfies $x = \sum_{i=0}^{l-1} 2^i x_i$, i.e. $[x]_B = [x_{l-1}], ..., [x_0]$. Denote a random number from $\mathbb{Z}_m$ as r. We use several primitive protocols to simplify our descriptions. They are $\texttt{Rand}(\mathbb{Z}_m)$ which generates a random element from the prime field $\mathbb{Z}_m$, $\texttt{Output}([x])$ which reveals the number x from its shares $[x]$, and $[x < y ? 1 : 0] \leftarrow \texttt{Bitlt}([x]_B, [y]_B)$ which carries out bit-wise comparison of x and y in shares and outputs 1 if $x < y$, otherwise 0 [36].

Bit Decomposition

Damgård et al. proposed bit decomposition with unconditional security with constant rounds [36]. Nishide and Ohta improved upon Damgård's work to achieve less complexity [99]. Their bit decomposition protocol is summarized in Protocol 1.

```
**Require**: $[x]$
**Ensure**: $[x]_B$

1. $[r]_B \leftarrow \texttt{Rand}(\mathbb{Z}_m)$
2. $[r] \leftarrow [r]_B$
3. $[c] = [x] - [r]$
4. $c \leftarrow \texttt{Output}([c])$. If $c = 0$ done, because $[r]_B = [x]_B$ by a coincidence.
5. If $c \neq 0$, compute $[q] = [m \leq r + c] \leftarrow 1 - \texttt{Bitlt}([r]_B, [m - c]_B)$
6. Compute $d_i$ for $i = 0, ..., l-1$ that satisfies $2^l + c - m = \sum_{i=0}^{l-1} 2^i d_i$, and compute
   $f_i$ for $i = 0, ..., l-1$ that satisfies $c = \sum_{i=0}^{l-1} 2^i f_i$.
7. Compute $[g_i] = (d_i - f_i)[q] + f_i$ for $i = 0, ..., l-1$, i.e. $[g]_B$.
8. Compute $[h]_B = [r]_B + [g]_B$.
9. Discard $[h_l]$ from $[h]_B$ to obtain $[x]_B$.
```

Protocol 1: DECOMPOSITION $[x] \rightarrow [x]_B$

Truncation

The truncation operation is a core component in the fixed-point arithmetic. Denote signed integers as $\mathbb{Z}_{\langle k \rangle} = \{\bar{x} \in \mathbb{Z} | -2^{k-1} \leq \bar{x} \leq 2^{k-1} - 1\}$. Signed integers are encoded in the prime field $\mathbb{Z}_m$ by $x = x \bmod m, m > 2^k$ where m is a prime. The truncation protocol computes $\bar{d} = \lfloor \bar{x}/2^s + u \rfloor$, where $\bar{x} \in \mathbb{Z}_{\langle k \rangle}$ and $s \in [1, ..., k-1]$. The bit u depends on the rounding method. Catrina and Hoogh proposed a truncation protocol that achieves statistical privacy [27]. Their protocol is summarized in Protocol 2.

Equality Test

Given two secret numbers $x, y \in \mathbb{Z}_m$, the equality test protocol computes $[(x == y?1 : 0)]$. The problem of deciding whether $x = y$ is equivalent to whether $x - y = 0$. Let $a = x - y$, and $c = a + r$ where r is a random number. Note that $c = r$ if $a = 0$. Nishide and Ohta proposed an equality test protocol without decomposing the secret

```
Require: $[x], k, s$
Ensure: $[d]$
1. $[r''] \leftarrow \texttt{Rand}(\mathbb{Z}_{\langle k \rangle})$, $[r'] \leftarrow \texttt{Rand}(\mathbb{Z}_{\langle k \rangle})$
2. $[r']_B \leftarrow \texttt{Decomposition}([r'])$
3. $c \leftarrow \texttt{Output}(2^{k-1} + [x] + 2^s[r''] + [r'])$
4. $c' \leftarrow c \bmod 2^s$
5. $[d] \leftarrow ([x] - c' + [r'])(2^{-s} \bmod m)$
```

Protocol 2: TRUNCATION $[d] \leftarrow \texttt{Trunc}([x], k, s)$

d into bits. Their protocol is summarized in Protocol 3.

```
Require: $[x], [y]$
Ensure: $[w] = [(x == y)?1 : 0]$
1. $[r]_B \leftarrow \texttt{Rand}(\mathbb{Z}_m)$
2. $[r] \leftarrow [r]_B$
3. $[a] = [x - y]$, $[c] = [a] + [r]$
4. $c \leftarrow \texttt{Output}([c])$
5. Represent $c$ bitwise as $c_{l-1}, ..., c_0$, and denote new variable $[c_i']$ for $i = 0, ..., l-1$.
       If $c_i = 1$, $[c_i'] = [r_i]$.
       Else if $c_i = 0$, $[c_i'] = 1 - [r_i]$. Note that $c_i' = 1$ iff $c_i = r_i$.
6. $[w] = \bigwedge_{i=0}^{l-1}[c_i']$.
```

Protocol 3: EQUALITY-TEST $[w] \leftarrow \texttt{EqualityTest}([x], [y])$

Interval Test

The interval test protocol is to decide whether a secret element $x \in \mathbb{Z}_m$ is within the interval $[c_1, c_2]$ or not, where public constants $c_1, c_2 \in \mathbb{Z}_m$ and $c_1 < c_2$. Nishide and Ohta proposed an interval testing protocol without decomposing the secret x into bits. Their protocol is summarized in Protocol 4.

Require: $[x], c_1, c_2$
Ensure: $[d] = [(([c_1 < x < c_2])?1 : 0]$

```
1. $[r]_B \leftarrow \texttt{Rand}(\mathbb{Z}_m)$
2. $[r] \leftarrow [r]_B$
3. $[c] = [a] + [r]$
4. $c \leftarrow \texttt{Output}([c])$
5. If $c_1 < c < c_2$ is not true,
       If $c_2 \leq c$,
         $[d_1] = \texttt{Bitlt}([c - c_2]_B, [r]_B)$, $[d_2] = \texttt{Bitlt}([c - c_1]_B, [r]_B)$,
         $[d] = [d_1] \times [d_2]$.
       If $c \leq c_1$,
         $[d_1] = \texttt{Bitlt}([c + m - c_2]_B, [r]_B)$, $[d_2] = \texttt{Bitlt}([c + m - c_1]_B, [r]_B)$,
         $[d] = [d_1] \times [d_2]$.
6. If $c_1 < c < c_2$ is not true,
       $[d_1] = \texttt{Bitlt}([c - c_1 - 1]_B, [r]_B)$, $[d_2] = \texttt{Bitlt}([c + m - c_2 + 1]_B, [r]_B)$,
       $[d_3] = [d_1] \times [d_2]$,
       $[d] = 1 - [d_3]$.
```

Protocol 4: INTERVAL-TEST $[d] \leftarrow \texttt{IntervalTest}([x], c_1, c_2)$

Exponentiation

Given a public constant $a \in \mathbb{Z}_m$ and the secret $x \in \mathbb{Z}$, the secure MPC protocol of exponentiation calculates $x^a \bmod m \in \mathbb{Z}_m$. Damgård et al. proposed a private exponentiation protocol with perfect privacy summarized as follows in Protocol 5 [36].

Comparison

Comparison is central to non-linear signal processing. To implement comparison, we first need to clarify its semantics in the prime field F_m. We allow negative secret

Require: $[x], a$
Ensure: $[x^a]$

1. $[r] \leftarrow \texttt{Rand}(\mathbb{Z}_m)$, compute $[r^a]$.
2. Compute $[xr]$, and reveal $xr \leftarrow \texttt{Output}([xr])$.
3. Compute $y = (xr)^a = x^a r^a$.
4. $[x^a] = y[r^{-a}] = y[(r^a)^{-1}]$.

Protocol 5: EXPONENTIATION $\texttt{Exp}([x], a)$

number $-x$ to be represented by $m - x$ in F_m. Thus, a comparison of $x > 0$ is equivalent to $x \bmod m < \lceil m/2 \rceil$. Second, we need to select m to be at least *twice* as big as the dynamic range of any intermediate and final values in the target signal processing algorithm. This is necessary to ensure that we can represent both $x_{max} - x_{min}$ and $x_{min} - x_{max}$ in F_m where x_{max} and x_{min} are the largest and smallest numbers. To handle the non-linear nature of comparison, we rely on the radix-2 representations of the numbers. Let us start with a simpler version of comparison that compares two secret numbers v and w *in plaintext* stored at two different parties. To simplify the description of the protocol, we assume a three-party computation, i.e. $n = 3$ and $t = 2$. The notation of this comparison protocol is as follows:

(P8) COMPARE2 (v at P_1, w at P_2) $\longrightarrow P_i : [v > w]_i^2$

where the binary predicate $v > w$ is 1 if true and 0 if false. Notice the binary output is represented in F_5, which is the smallest prime field one can use to represent a secret among three parties. We use such a small prime field to minimize the communication overhead. The details of the protocol are given in Protocol 6 [117].

Correctness: Steps 1 and 2 create shares for each bit in v and w. Step 3 computes the expression $(v_0 - w_0 + 1)$, which is 2 if $v_0 > w_0$, 0 if $v_0 < w_0$, and 1 otherwise. For the subsequent bits, b_j at step 4 is 1 until the corresponding bits from v and w begin to differ, and all the subsequent b_j's will be 0. $(v_j - w_j + 2 - b_j)$ will be 1 until

Require: v at P_1, w at P_2, and $[b_0]^2_i \triangleq [1]^2_i$
Ensure: $[c]^2_i$ at party P_i for $i = 1, 2, 3$ where $c \triangleq (v > w)$.

```
1. $P_{1,2} : v \triangleq v_0 \ldots v_{l-1}$ base 2, $w \triangleq w_0 \ldots w_{l-1}$ base 2
2. $P_{1,2} : [v_j]^2_i, [w_j]^2_i \longrightarrow P_i$ for $j = 0, \ldots, l - 1$.
3. $P_i : [c]^2_i \triangleq [v_0 - w_0 + 1]^2_i$
4. for $k = 1$ to 3 and $j = 1$ to $l - 1$
       $P_k : [[b_{j-1}(1 - v_{j-1} \oplus w_{j-1})]^3_k]^2_i \longrightarrow P_i : [b_j]^2_i$
       $P_k : [[c(v_j - w_j + 2 - b_j)]^3_k]^2_i \longrightarrow P_i : [c]^2_i$
5. $P_k : [[c^4]_k]^2_i \longrightarrow P_i : [c]^2_i$
```

Protocol 6: COMPARE2(v, w)

the bits start to differ – it will be 0 if $v < w$ and 2 if $w > v$. This leads to $c = 0$ if $v < w$. The subsequent steps inside the loop will not produce another 0 because $b_j = 0$ and $v_j - w_j + 2 - b_j$ can only be 1 or 3. As such, c is 0 if and only if $v < w$. At step 5, we invoke the Carmichael's theorem to ensure that the output can only be 0 or 1. *Security & Complexity*: Communications occur at steps 2, 4, and 5. Step 2 is the initial distribution of shares and the rest are all renormalization steps. As such, none of them provide any additional information. The protocol is thus secure. The communication complexity of COMPARE2 is $(20l - 6) \log 5$ bits.

COMPARE2 requires the plaintext of the secret numbers while a proper COMPARE function should assume that each party has access to only a share of the secret numbers. We will take the difference between the two secret numbers as $[x]^t_i$ and develop the COMPARE function to determine if $x < \lceil m/2 \rceil$ as follows:

(P9) COMPARE$\left([x]^2_i \text{ all } i\right) \longrightarrow P_i : [x < \lceil m/2 \rceil]^t_i$

Correctness: Steps 1 through 4 creates plaintext numbers v and w at party 2 and 3 to represent x using (3.2) such that

$$x = (v \bmod m + w \bmod m) \bmod m.$$

```
Require: $[x]_i^2$ at party $P_i$ for $i = 1, 2, 3$
Ensure: $[d]_i^2$ at party $P_i$ for $i = 1, 2, 3$ where $d \triangleq (x < \lceil m/2 \rceil)$.
  1. $P_1$ : a random $r \longrightarrow P_2$.
  2. $P_1 : u \triangleq \gamma_1[x]_1^2 - r \longrightarrow P_3$.
  3. $P_2 : v \triangleq \gamma_2[x]_2^2 + r$
  4. $P_3 : w \triangleq \gamma_3[x]_3^2 + u$
  5. COMPARE2$(m - v, \lfloor m/2 \rfloor + w) \longrightarrow [a]_i^2$
  6. COMPARE2$(v, m - w) \longrightarrow [b]_i^2$
  7. COMPARE2$(m + \lceil m/2 \rceil - v, w) \longrightarrow [c]_i^2$
  8. $P_k : [[a \oplus (bc)]_k]_i^2 \longrightarrow P_i : [d]_i^2$
```

Protocol 7: COMPARE$\big([x]_i^t$ for $i = 1, 2, 3\big)$

If $x < \lceil m/2 \rceil$, then one of the following two statements must hold:

$$v \bmod m + w \bmod m \quad < \quad \lceil m/2 \rceil \tag{3.4}$$

$$m \quad \leq \quad v \bmod m + w \bmod m \quad < \quad m + \lceil m/2 \rceil \tag{3.5}$$

Step 5 checks the inequality in (3.4), and steps 6 and 7 check (3.5). Since the two steps cannot both return 1, the output can be computed using an exclusive-or as in step 8. *Security & Complexity*: Since r is random, party 2 and 3 cannot gain any knowledge about the original share of party 1. The security of the rest of the protocol is guaranteed by that of COMPARE2. The communication complexity of COMPARE is $2l + (60l - 6) \log 5$ bits.

3.3 Privacy-Protected Image Denoising

This section describes the adaptation of wavelet denoising into secret shares. As described in Section 1.1, two customers, User U and a software provider Vendor V come to a third party computing platform for a joint task. U owns a secret image and V owns the filters and the denoising parameters. All secret data will be decomposed into random shares which are distributed to three non-colluding semi-honest computational parties to execute the algorithm.

Discrete Wavelet Transform (DWT)

DWT is the first step of many important image processing functions including denoising, compression, and enhancement. The most common implementation of DWT is via a dyadic tree, shown in Figure 3.1 (a). Such an implementation involves the repeated application of linear filtering. In order to preserve the homomorphism, a renormalization step needs to be applied after each level of filtering. Let's consider the communication complexity of implementing a 2-level 2D-DWT. Suppose the size of the initial secret share is l bits per pixel for the image. We ignore the contributions from the filters which are small compared with the image. Initial dissemination creates $3l$ bits per pixel. After horizontal filtering, the first renormalization generates an additional $6l$ bits per pixel. As the second level of DWT applies only to the lowest frequency subband, it generates an addition $6 \cdot \frac{l}{4} = \frac{3l}{2}$ bits per pixel for each of the vertical and horizontal filtering. The overall communication overhead is thus $3l + 6l + 3l = 12l$ bits per pixel. Alternatively, we can flatten the dyadic tree into a filter bank of seven subband filters as shown in Figure 3.1 (b). The filter bank itself can be computed by passing a single impulse as an input signal to the dyadic DWT. As the input has only 1 pixel, the communication overhead is negligible. The subsequent processing on the random shares of the image does not need any communication beyond the initial distribution of the shares, which amount to only $3l$ bits per pixel or $\frac{1}{4}$ of that required by a dyadic tree implementation. The computation complexity increases slightly due to the use of 2-D filters.

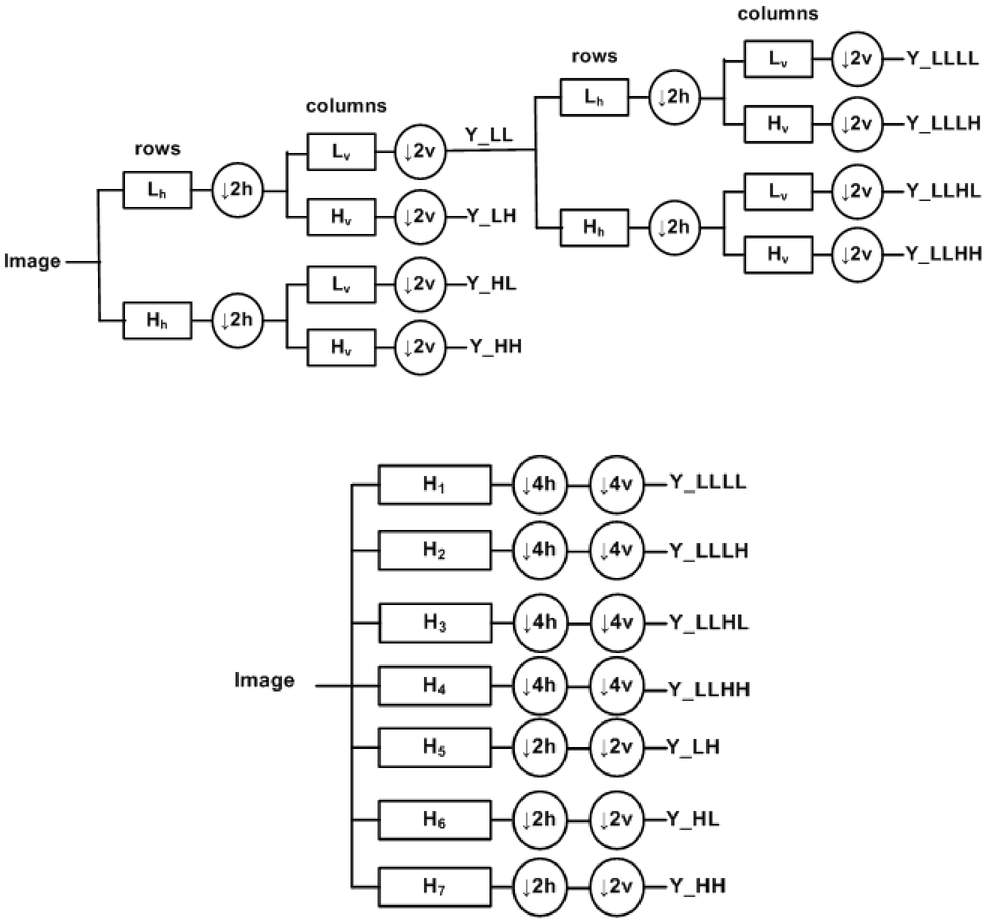

Figure 3.1: Two implementations of DWT: (a) via a dyadic tree, (b) the flattened version of (a)

Wavelet shrinkage

In this section, we describe our implementation of a popular wavelet shrinkage denoising technique called BayesShrink [29]. While keeping the wavelet coefficients from the lowest frequency subband y_L intact, coefficients from the high frequency subbands y_H are shrunk by a threshold λ if their absolute values are bigger than λ or set to zero if smaller. In the share domain, this shrinkage function can be implemented with the COMPARE protocol from Section 3.2:

$$\begin{aligned}\delta_\lambda(y_H(i)) \triangleq\ & \text{COMPARE}(y_H(i),\lambda)(y_H(i)-\lambda)+ \\ & \text{COMPARE}(-\lambda,y_H(i))(y_H(i)+\lambda) \qquad (3.6)\end{aligned}$$

There are many different approaches to determine λ and we use $\lambda = \sigma_y^2/\sigma_w$, where σ_y is the estimated noise variance of the image:

$$\sigma_y = \frac{\sum_{i=1}^{N} |y_{HH}(i)|}{0.6745N} \tag{3.7}$$

with y_{HH} denoting the highest diagonal frequency subband and N its size. σ_w estimate the noise variance of the particular subband to be denoised:

$$\sigma_w = \left[\max\left(\frac{1}{N_w}\sum_{i=1}^{N_w} y_w(i)^2 - \sigma_y^2, 0\right)\right]^{1/2} \tag{3.8}$$

with y_w denoting the subband and N_w denoting the number of coefficients in y_w. This implementation is a slight variation of that described in [29].

While it is possible to implement these equations entirely in the secret share domain, the protocol will reveal the threshold calculation process – a key step which the software provider may want to keep as a secret. To keep this step proprietary, we compute subband statistics including $\sum_i |y_{HH}(i)|$ and $\sum_i y_w(i)^2$ in shares but reconstruct the final statistics into plaintext for the software provider to determine λ. Such statistics reveal little information about the details of the image but enable the software provider the flexibility in determining the threshold in private. Finally, the provider will create 3 shares of λ and send them back to the computational agents to complete the shrinkage from Equation (3.6). A final remark about the calculations of the two subband statistics: to prevent possible overflow, we use multiple words to keep track of overflow digits. For example, if we want to hold the running sum in two words S_0 and S_1, we will first compute the carry bit $C = \text{COMPARE}(S_0, R_{max} - x)$ where R_{max} is the range for one word and $x < R_{max}$ is the next number to be added. Then, we store the results in two words: $S_1 \triangleq S_1 + C$ and $S_0 \triangleq S_0 - CR_{max} + x$.

3.4 System Implementation and Experiments

We implement our protocols in MATLAB on five different machines on the same LAN. Machines representing the user and the filter owner carry out the simple tasks generating shares and assembling the final results. The bulk of the computation are done at the three machines corresponding to the computational agents. They all run windows 7 with 4 GB RAM on Intel Core 2 Quad CPU Q9650 @ 3 GHz. For the wavelet filters, we use the rationalized version of the 9/7 biorthogonal pair of Cohen, Daubechies, and Feauveau [125]. We chose the modulus m to be the prime just greater than 2^{31} and evaluate the shares at $k_i = 1, 2$, and 3. Table 3.1 provides time measurements including data transmission time of the proposed algorithms using a 9/7 wavelet filter on a 128 × 128 image. RENORMALIZATION is essentially the same as SHARE CREATION plus network time in distributing shares. Their difference highlights the fact that a significant portion of time is devoted for networking. COMPARE2 represents a significant increase in complexity compared with the more elementary functions. Even though we execute each step of our protocol over the entire image so that we can accumulate enough information for better network utilization, the repeated usage of renormalization in COMPARE2 significantly increases the time spent on data transmission. For a 1-level denoising scheme, roughly 10 COMPARE operations are used per pixel – two are for shrinkage while the remaining eight are for statistics computation.

Table 3.1: Time measurements of different protocols

Algorithm	Time (per pixel)
SHARE CREATION	3.600 us
RENORMALIZATION	5.587 us
COMPARE2	1.126 ms
COMPARE	3.425 ms
1-Level DENOISING	30.406 ms

3.5 Summary

In this chapter, we have presented information-theoretic secure protocols for privacy-protected signal processing. Using the classical Shamir's secret sharing scheme, we have developed algorithms to handle various fundamental signal processing operations. These operations are used to build a realistic wavelet denoising system over three non-colluding computing agents and performance numbers are measured. Other signal processing applications on secret shares and network protocols to deter collusion in a distributed environment will be described in details in the remaining of this dissertation.

Chapter 4 Collusion Attacks in the Outsourced Computing

In this chapter, we begin with an overview of our computing framework. We then provide a background review on game theoretic mechanisms in Section 4.1 which is needed for the remainder of the dissertation. Section 4.2 presents our collusion attack model.

4.1 Background

Outsource Computation

Our computing framework is composed of two types of participants: the computing platform *agents* and the platform *customers*, i.e. the secret-data owners. Denote any pair of platform customers as U and V who want to cooperate in a joint computation. We focus our discussions on two parties but the scheme is general enough for arbitrary number of parties. U and V do not trust each other with their secret data and they do not possess the necessary resources for the computation. As such, they outsource their computation to the computing platform by means of the SSS protocols.

At the heart of any SSS protocols is the assumption of the availability of multiple computing agents. Compared with other encrypted-domain techniques, SSS is particularly suitable for protecting privacy in distributed signal processing because it is information-theoretic secure and does not require computation in a large prime field. We denote a computing agent in the computing platform as A_i where $i \in \{1, 2, \ldots\}$. To coordinate different agents, we assume that there is a coordinator C who is responsible for keeping records of the IDs of participants but does not handle any actual secret data. This architecture is illustrated in Figure 4.1. Note that U and V do not trust A_i's and C with their data either. The agents A_is are localized covert adversaries. Details of the adversarial model for A_i's and C will be discussed in Section 4.2.

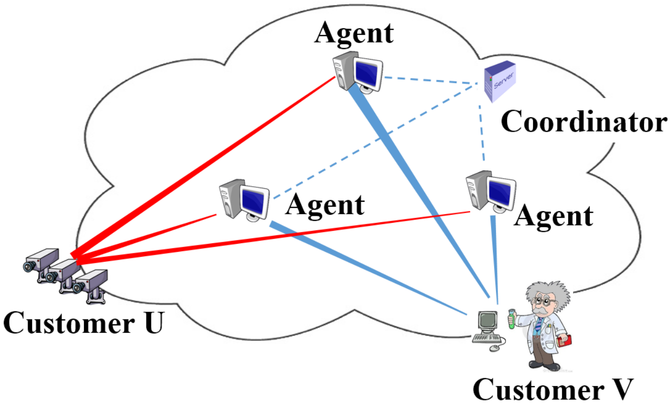

Figure 4.1: Outsourced Computation Framework

Despite its simplicity, this computational framework is an abstraction of many practical scenarios. For example, in the context of cloud computing, A_i provides platform as a service (PaaS) while U can be a user with sensitive data and V is a proprietary software vendor providing software as a service (SaaS) [128]. Another example is privacy-preserving data mining in which a data mining algorithm running at A_i's are applied on a large dataset comprising of private data from both U and V [81]. It is also important to note that our emphasis is on protecting privacy of data, rather than the programming instructions. In general, we assume that U and V are fully aware of the intention and flow of the program as they need to prepare their data in the appropriate form. The actual program is carried out at each of the agents A_i. However, A_i has access only to encrypted data so the actual program it is executing could be obfuscated to hide data communication patterns that might reveal important information about the data. More discussions on this issue can be found in Section 4.2.

Renormalization for Multiplication in Shamir's Secret Sharing

Shamir's Secret Sharing (SSS) protects privacy by decomposing a secret into input shares. It is *information theoretic* secure: an adversary has no knowledge of the secret at all regardless of its computing power if the number of input shares it obtained does not satisfy the pre-defined access structure of the underlying sharing scheme [35]. For the (t, n)-SSS scheme where n is the number of computing agents and $t < n$ is a designed parameter called *threshold*, the access structure is for any entity holding at least t shares. The details of the SSS has been review in Section 3.1.

Assume that there are n computing agents and U has a private input x from a prime field $\mathbb{F}_m$, where m is a prime number. U hides x as the constant term of a random $(t-1)$-degree polynomial, and generates n shares, $[x]_i^t$ for $i = 1, 2, \ldots, n$. The second party, V, follows the same procedure in using a random $(t-1)$-degree polynomial of his/her choice to break his/her secret number $y \in \mathbb{F}_m$ into shares $[y]_i^t$ for $i = 1, 2, \ldots, n$. The i-th shares of both x and y are sent to agent i for processing.

SSS is homomorphic in addition, scaling (by a known factor) and multiplication, which are universal in building any arithmetic circuit [12]. For multiplication of shares, the resulting polynomial has a higher degree of $2t-2$:

$$[xy \bmod m]_i^{2t-1} = [x]_i^t [y]_i^t \bmod m. \tag{4.1}$$

Notice that the degree of the product polynomial increases to $2t-2$. Thus, the threshold for reconstruction will also increase to $2t-1$, requiring almost twice as many shares, or equivalently agents, to reconstruct the product. There are two solutions to this problem: the first solution is to increase the number of parties n to guarantee that n is large enough to accommodate all multiplication operations. The second solution is to apply a "renormalization" procedure to reduce the threshold back to t [16]: each agent breaks its product share into n separate shares, and sends one share to each of the corresponding agents. The final share at each agent is computed as a weighted summation of these newly received shares from other agents as shown below:

$$[xy \bmod m]_i^t = \sum_{j=1}^{n} \gamma_j [[xy \bmod m]_j^{2t-1}]_i^t \bmod m. \tag{4.2}$$

It can be shown that the renormalization process is information-theoretic secure [16].

The first approach is less flexible because a large n must be used throughout the entire computation process. In fact, it can be shown that the renormalization is more scalable than adding more agents. Suppose there are η multiplications in a procedure with shares generated at the original threshold t. The first solution will require at least $\eta(t-1)+1$ agents. All secret numbers used in the calculations must be decomposed into $\eta(t-1)+1$ shares and the final reconstruction must require collection of all the shares. Assuming a new secret number is used in each multiplication, there will be $\eta+2$ rounds of shares dissemination including the initial secret and final collection. The total bandwidth required is proportional to $(\eta+2)\cdot(\eta(t-1)+1) = \mathcal{O}(\eta^2)$.

For the second approach, the number of agents n is independent of the number of multiplication operations and can be set to the smallest value of $n = 2t-1$. For each renormalization step, each agent needs to send shares to every other agent, resulting in $(2t-1)(2t-2) = 4t^2 - 6t + 2$ shares being exchanged. Taking into account the share generation of all $\eta+1$ secret numbers, $\eta-1$ renormalization steps and the final reconstruction, the total bandwidth is proportional to $(\eta+2)\cdot(2t-1)+(\eta-1)\cdot(4t^2-6t+2) = \mathcal{O}(\eta)$. It is thus expected that the second approach is more scalable to complex signal processing algorithms. With the use of renormalization, we concentrate our discussions on the simplest access structure with $t = 2$ and $n = 2t-1 = 3$ agents which is adequate for any computation procedure. Nevertheless, all of our proposed algorithms apply equally well to general access structures.

Game Theory

Game theory provides a mathematical foundation to analyze situations where two or more participants or players make rational decisions that influence one another's welfare. The interactions may include both conflict and cooperation. There are many categories of games in the literature. In our work, we adopt *strategic form games* to model collusion attacks as non-cooperative games. Specifically, we utilize the strategic form games in the context of population evolution to demonstrate that the proposed games indeed have stable solutions. Game-theoretic concepts used in our framework are summarized below and we refer readers to the excellent coverage of the topics in [136] and [94] for details.

Definition 4.1.1. A *strategic form game $\mathit{\Gamma}$* is defined as a tuple $\langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$, where $N = \{1, 2, ..., n\}$ is a finite set of players, S_i is the set containing all available strategies of player i, and $u_i : S_1 \times ... \times S_n \rightarrow \mathbb{R}$ for $i = 1, 2, ..., n$ are mappings called the utility functions or payoff functions.

Then a game between two players could be described in a table where the entries of the first column stand for the strategies of the row player and the entries of the first row stand for the strategies of the column player. Each combination of a row strategy and a column strategy determines a utility to a player.

In game theory, players are assumed to be interested in maximizing his/her utility. A celebrated solution concept is the *Nash Equilibrium*. We describe the two-player case here.

Definition 4.1.2. A *Nash Equilibrium* (NE) for two player games is a pair of strategies (s_1^*, s_2^*) such that $u_1(s_1^*, s_2^*) \geq u_1(s_1, s_2^*), \forall s_1 \in S_1$ and $u_2(s_1^*, s_2^*) \geq u_2(s_1^*, s_2), \forall s_2 \in S_2$.

Another solution concept is the *Dominant Strategy Equilibrium* (DSE). The number of players N could be bigger than 2. We denote the strategy of player i as s_i and the strategy profile of i's opponent players as s_i. We present the strong sense of dominance here:

Definition 4.1.3. Given a game $\mathit{\Gamma} = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$, a strategy $s_i \in S_i$ is said to strongly dominate another strategy $s'_i \in S_i$ if $u_i(s_i, s_{-i}) > u_i(s'_i, s_{-i}), \forall s_{-i} \in S_{-i}$

Definition 4.1.4. A strategy $s^*_i \in S_i$ is said to be a strongly dominant strategy for player i if it strongly dominates every other strategy $s_i \in S_i$.

Definition 4.1.5. In a n-person game, a strategy profile $(s^*_1, ..., s^*_n)$ is a strongly dominant strategy equilibrium of the game $\mathit{\Gamma} = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ if s^*_i is a strongly dominant strategy for all i.

It is easy to see that any (strongly) DSE is also a NE but the converse is not true. The conditions for DSE are very strong and many games do not even have a DSE. On the other hand, NE always exists in a finite game [136]. In Section 5.1, we will consider collusion and honesty as possible strategies in our secure MPC games, and derive conditions under which honesty is part of these different solution concepts.

In a distributed systems, agents usually have additional *private information* that affects their decision making. For example, in an auction, each bidder may have his/her own private valuation of the item that is kept secret to themselves, but the actual bid they submit could be different than their valuation. Similarly, in secure MPC, each user may value their secrets differently. When players have their private information about the game that other players do not know, the game can be analyzed using *Bayesian Game* theory.

Definition 4.1.6. A *Bayesian Game* $\mathit{\Gamma}$ is defined as a tuple

$$\langle N, (\mathit{\Theta}_i)_{i \in N}, (S_i)_{i \in N}, (\alpha_i)_{i \in N}, (u_i)_{i \in N} \rangle,$$

where

- $N = \{1, 2, ..., n\}$ is a finite set of players.
- Θ_i is the set of private information, or *types*, of player i where $i \in N = \{1, ..., n\}$. Denote $\Theta = \Theta_1 \times \ldots \times \Theta_N$.
- S_i is the strategy set of player i. Denote $S = S_1 \times \ldots \times S_N$.
- The probability function α_i specifies a probability distribution $\alpha_i(.|\theta_i)$ over the set Θ_{-i}. It represents the belief of player i on the types of other players, denoted as Θ_{-i}, if his/her own type is θ_i.
- $u_i : \Theta \times S \rightarrow \mathbb{R}$ for $i = 1, 2, ..., n$ are utility functions.

The solution concepts of NE and DSE can be easily extended to Bayesian games [94]. The computation of Bayesian NE depends on the knowledge of the prior belief functions α_i, which can be difficult to obtain in many practical applications. As such, it is generally preferable to use DSE to analyze Bayesian games as their analysis does not require the belief functions [129]. We will follow this practice in our work by first using NE to analyze the simpler case where all players have the same privacy preference, and then switching to DSE when we use Bayesian games to analyze uncertainty in privacy preference.

To study large distributed systems, we need to map game theoretic analysis to a population of similar players playing the same strategic form game with different strategies over a period of time. The framework to analyze such games is called the Evolutionary Game Theory (EGT). EGT does not assume the players to be hyper-rational as in the case of traditional GT. Instead, the players can learn from their previous payoffs and update their strategies accordingly. Besides its origin in modeling biological processes [136], EGT is also suitable to analyze situations where many autonomous agents with conflicting interests interact with each others in a distributed system to achieve specific goals. For example, EGT has been used to model P2P streaming [31], wireless network selection [100], spectrum sharing [101], and social networks [102]. In this dissertation, we use EGT in Section 7.2 to demonstrate the effectiveness of our countermeasures while treating collusion as a strategic form game.

The alternative to NE in EGT is the possible existence of an evolutionary endpoint of adopting a specific strategy s^*, called the *evolutionary stable strategy* (ESS). Given a 2-player strategic form game, suppose the game is played between any pair of players from a population. We define a *population profile* $\mathbf{x}$ such that $\mathbf{x}(s)$ for $s \in S$ denotes the fraction of the population playing s. To consider if a particular strategy s^* is evolutionary stable, we specialize the notation $\mathbf{x}_\varepsilon$ to mean $\mathbf{x}(s^*) = 1 - \varepsilon$ and $\sum_{s \neq s^*} \mathbf{x}(s) = \varepsilon$ for $\varepsilon \in [0, 1]$. The second term is typically referred to as the mutant population. Then, we have the following definition of ESS:

Definition 4.1.7. The strategy s^* is an evolutionary stable strategy or ESS if there exists an $\overline{\varepsilon}$ such that for every $0 < \varepsilon < \overline{\varepsilon}$ and $u(s^*, \mathbf{x}_\varepsilon) > u(s, \mathbf{x}_\varepsilon), \forall s \in S \setminus \{s^*\}$ where $u(s, \mathbf{x}) = \sum_{s' \in S} \mathbf{x}(s')u(s, s')$.

In other words, $u(s, \mathbf{x}_\varepsilon)$ denotes the average payoff of a new player entering the population playing strategy s against a random player from population profile $\mathbf{x}_\varepsilon$. As this new player is more likely to choose s^*, the mutant population will diminish, further strengthening the condition in the definition. The population profile when no further evolution occurs anymore is called *evolutionary stable state.* It is also important to note that ESS is a stronger concept than NE. In fact, an ESS must also be a NE but a NE is not necessarily evolutionary stable [94].

In addition to ESS, another core concept in EGT is the *evolutionary game dynamics.* It describes how the population profile changes over time based on the fitness of each strategy [69]. The most common approach to model the dynamics is through a set of *replicator* equations for each strategy in the form of

$$\frac{d}{dt}\mathbf{x}(s) = \mathbf{x}(s) \cdot \left(u(s, \mathbf{x}) - \sum_{s' \in S} \mathbf{x}(s')u(s', \mathbf{x}) \right) \tag{4.3}$$

where $\mathbf{x}$ is the population profile. Equation (4.3) describes a simple exponential growth model. This is generally applicable to autonomous agent systems where there is inertia in staying with the same strategy and the rate of change of the population using a particular strategy depends on the difference between the utility of that strategy and the average utility. In Section 7.2, we will use replicator dynamics to simulate our mechanisms to validate the theoretical analysis.

4.2 Collusion Attack Models

Unlike two-party garbled circuits and homomorphic encryption, SSS-based protocols are prone to collusion attacks (CA). We now describe the different types of collusion attacks that can occur under the SSS-based outsourced computation framework. We will continue to use the same notations as defined in Sections 3.1 and 4.1.

A1: Side-channels among agents for collusion

If an adversary controls t or more computing agents involved in the computation, they can exchange their secret shares freely through their pre-established side channels to reconstruct the secret numbers x and y. As the communication through the side channels is independent from the information exchanged within the protocol, such attacks cannot be detected within the protocol and must be tackled at the architecture level. One possible approach to deter such attacks is by *obfuscating the computing task* to make it difficult for an adversary to identify the computing agents involved in a specific task and determine their functions. A classical example is the aforementioned online gambling – poker room companies assign players from certain countries to be at different tables to reduce the possibility of pre-existing side channels [5]. The exact

approach when applied in the context of distributed computation will depend on the infrastructure behind the computing platform, P. We consider two scenarios:
(i) **P2P**: P is formed by amassing a large number of independent computing agents on the internet that contribute their CPU cycles in exchange for small payments. To deter an adversary from identifying the set of agents involved in a task, we can rely on the *coordinator* C that randomly assigns agents to a task. C is assumed to be trusted with keeping the mapping secret. Since C is not involved in the actual computation, the additional measure required to secure C should not significantly affect the scalability of the platform. Such a hybrid approach of mixing computation/communication peers with coordinators is quite common among peer-to-peer systems [122]. The use of anonymity network protocols such as Tor [43] can also prevent the formation of the side channels and force the communication to the assigned communication channels.
(ii) **Enterprise Cloud**: P is centrally managed by an enterprise system. The solution is to obfuscate the computation process so that different agents on the same cloud would not be able to recognize that their processes originate from the same task. All identifiable information, such as IP addresses of the user and vendor, must be obfuscated while dummy instructions and data should be added to mask the traffic pattern [9].

Nevertheless, these approaches on system architecture cannot provide any guarantee on preventing the formation of side channels. The detailed implementation of these approaches are system oriented and beyond the scope of our work. In the sequel, we will assume that no such side channels exist among agents and model the adversaries as localized or restricted to the assigned communication channels [24].

A2: Collusion between agents and U or V

We will focus on the agents' collusion with U, as the case for V is identical. As each agent possesses secret shares from both U and V, it is possible for U to collude with t

or more agents to reconstruct y from V. No changes in infrastructure can block such an attack as it is necessary for U to communicate with the agents. Neither can the collusion be detected from the communication between V and the agents. To deter such an attack, we propose a retaliation mechanism such that a heavy penalty can be levied on U if V can provide convincing evidence of the leakage of his/her secret through U. In Section 5.1, we study the choice to collude or to stay honest under retaliation as a game between U and V. We show that being honest is the solution, provided that there exist effective tools to collect evidence of theft.

On the other hand, it is not always possible to collect any evidence or it may be too costly to go through with the retaliation process. To cope with such an "undetectable" theft, we observe that enough agents must be involved in a collusion attack for it to be successful. Thus, collusion can be deterred by having undercover police officers disguised as corrupted users/vendors in catching agents who are willing to collude. In Section 6.1, we formulate such an interaction as an evolutionary game and show that if there are enough police officers, being honest is indeed an NE for the agents.

A3: Collusion attack by computing agents

The direct communication among agents is essential as it is needed in the renormalization procedure and the reconstruction of necessary intermediate values in more complicated protocols. On the other hand, it also opens doors for them to collude. The difference between *A1* and *A3* is that the communication is localized in that the coalition of agents in *A3* forms after the random assignment of agents to U and V. As such, it is possible for U and V to thwart this collusion by inspecting the communication among agents. Note that the communications among agents consists of random secret shares so it is challenging to identify if they actually contain subliminal data for collusion. Also, uncontrolled examination by U (or V) may reveal information about the original secret data from other parties. In Section 5.1, we suggest a censorship scheme in which U and V collect the data from each agent and randomize them before sending them back to the agents. Subliminal communication becomes impossible due to the injection of random noise known only to U and V.

Chapter 5 Anti-collusion for Customers

In this section, we use game-theoretic techniques to model *A2* collusion attacks as described in Section 4.2, identify conditions and propose countermeasures to deter such attacks. Collusion attacks *A2* refer to the collusion formed between the agents and either U or V to steal the other's secret. In Section 5.1, we first study the strategy of either U or V in participating in a collusion attack to steal the other's secret as a game called *User-Vendor Game*. Then, in Section 6.1, we will consider the influence of the agents and propose countermeasures in a game called *Customer-Agent Game*.

5.1 User-Vendor Game

We assume that, before starting the joint computation, there exists a legally-binding contract in place so that U and V both understand that they should not collude with agents in stealing each other's secrets. This contract would stipulate that if one party, say V, finds out that U tried to steal V's secret, U would be liable to pay for the damages based on charges brought by V. In retaliation, U could countercharge V with similar accusations. The judgement in resolving such a conflict would need to be carried out by proper authority, possibly after a long proceeding in evaluating the legitimacy of evidence provided by both parties. We call this strategy undertaken by U and V *retaliation*.

With the initial strategy of staying honest or cheating and the follow-up strategy of possible retaliation, there are four possible combinations for each player or a total of 16 different interaction outcomes between both of them. All possible cases are listed in Table 5.1 with $C_U, C_V = 1$ represent cheating and $R_U, R_V = 1$ represents retaliation.

Among the different combinations, there are cases that we believe are unlikely. We

Table 5.1: Different Outcomes in User-Vendor Games

Case	C_U	C_V	R_U	R_V	**Outcome**
1	0	0	0	0	D
2	0	0	0	1	X
3	0	0	1	0	X
4	0	0	1	1	A
5	0	1	0	0	E
6	0	1	0	1	X
7	0	1	1	0	X
8	0	1	1	1	A
9	1	0	0	0	C
10	1	0	0	1	X
11	1	0	1	0	X
12	1	0	1	1	A
13	1	1	0	0	B
14	1	1	0	1	X
15	1	1	1	0	X
16	1	1	1	1	A

mark these unlikely cases with outcome X based on the *an-eye-for-an-eye* assumption: if one party retaliates by filing charges for a suspected cheating offense, the other party will retaliate with a counter-lawsuit. This simplistic world-view is based on the fact that a player must be made aware of the retaliation action from the other and the only rational reaction to protect oneself is to countersue. This is also supported in real life by the large number of litigation, especially in the United States, from simple small-claim charges to multinational patent infringement lawsuits between companies [33]. Based on this assumption, case 2, 3, 6, 7, 10, 11, 14, 15 from Table 5.1 are excluded from further considerations.

For the remaining cases, the goal is to investigate the preference of different outcomes in order to derive the optimal strategies [136]. We group the remaining cases into five possible classes of outcomes from A through E. Case 4, 8, 12 and 16 all involve mutual retaliation with outcomes ultimately decided by an external entity (court). Since the process is likely to be long, tedious and highly uncertain, we make the assumption that *any case with mutual retaliation always results in the least desir-*

able outcome and collectively label these outcomes A. The assumption that retaliation is undesirable does not imply that cheater can ignore such a possibility. Rather, it means that both players will either avoid this outcome by staying honest or retaliating if the evidence against the other is overwhelming and the value of the secret is higher than the cost of retaliation. The decision to retaliate is at the heart of our mechanism design. Its mathematical underpinning will be discussed later in this section.

The remaining four outcomes do not involve any retaliation, which means that the computation completes successfully and each party gets rewarded for carrying out his/her task. To study the preference ranking of these outcomes, we assume the perspective of U because the case for V is identical. To consider possible preference orders, we first use a cost and benefit analysis to eliminate unlikely orders and then analyze the remaining ones using different games to study the equilibrium strategies.

From the perspective of U, a rational judgement on the preference would be based on the relative values between the two secrets and the additional cost associated with cheating. While U clearly knows the value of his secret, his estimate of V's secret is imprecise. In addition, the value of the secret depends on whether the secret holder decides to collude and cheat – it is unlikely that a cheater will put forth a genuine secret in the joint computation. The cost of cheating would include additional cost to get the majority of the computing agents into a collusion.

While outcome A is the least desirable, outcome C is the most desirable because U also successfully steals V's secret and suffers no consequence. As V is honest, V's secret should be of high enough value to cover U's cost in collusion if this collusion attack is a rational act. All the other outcomes are not as good: outcome D represents the case when U is honest without any additional gain, though it is the socially optimal behavior; outcome E represents the case when U is honest and suffers a loss as his secret was stolen by V; outcome B represents the case when both U and V cheat and steal each other's secret. Note that for this case V's secret may not be of high enough value to cover U's cost because V cheats. While it is clear that outcome D should be ranked higher than E, it is unclear where outcome B should be. In summary, there are three preference orders we need to consider:

$$\mathrm{C} \succ \mathrm{D} \succ \mathrm{E} \succ \mathrm{B} \succ \mathrm{A} \tag{5.1}$$

$$\mathrm{C} \succ \mathrm{D} \succ \mathrm{B} \succ \mathrm{E} \succ \mathrm{A} \tag{5.2}$$

and

$$\mathrm{C} \succ \mathrm{B} \succ \mathrm{D} \succ \mathrm{E} \succ \mathrm{A} \tag{5.3}$$

where the symbol $\succ$ denotes "is preferred over". The three preference orders differ in the ranking of outcome B. A useful way to understand these orders is based on the cost of collusion. The first order (5.1) ranks B the lowest because the high cost of collusion exceeds even the damage of losing one's own secret in E. On the other hand, the last order (5.3) implies that the collusion cost is lower that the gain of stealing the secret from the dishonest V and results in a net gain for U. From the viewpoint of mechanism design, it is intuitive to make collusion cost as high as possible for deterrence, which will be the subject of Sections 6.1 and 6.2. In the remainder of this section, we study how retaliation impacts the strategies of honesty versus cheating for different preference orders.

Symmetric Games

In this subsection, we study the initial strategy of staying honest versus cheating under possible retaliation, assuming that all the players have the same preference order. The follow-on strategy of retaliation involves complicated factors including the detection of collusion attacks and the availability of evidence in support of retaliation. Due to the difficulty in modeling the payoff utility for retaliation, we instead model it as a parameter q and study its relationship with other factors. Specifically, we define q as the "non-retaliate" probability for both U and V, conditioned on the other's cheating behavior. The extreme value $q = 1$ means that no one retaliates while $q = 0$ means that one always retaliates if his/her secret is stolen. A useful alternative interpretation of q is to view it as the normalized reward for cheating, which is only worthwhile if the cheater can get away with the retaliation. We are ignoring the scenario of retaliation in the absence of cheating behaviors (case 4 in Table 5.1) – we believe that this is an extremely unlikely situation considering that retaliation is the most undesirable outcome.

As we have five outcomes to consider, we denote the normalized utility values for these outcomes as $0 = p_0 < p_1 < p_2 < p_3 < p_4 = 1$. For the three preference orders corresponding to high collusion cost (5.1), medium collusion cost (5.2) to low collusion cost (5.3), the mappings of the utility values to the three cases can be easily deduced and are shown in Table 5.2.

Table 5.2: Normalized Utility of Different Outcomes in User-Vendor Games

	U's Preference		
Strategies	**High cost (5.1)**	**Mid cost (5.2)**	**Low cost (5.3)**
Retaliate (A)	p_0	p_0	p_0
Both cheat (B)	p_1	p_2	p_3
U cheats only (C)	p_4	p_4	p_4
No one cheats (D)	p_3	p_3	p_2
V cheats only (E)	p_2	p_1	p_1

Each of the three assignments can form a two-player, two-strategy symmetric strategic form game. There are many possible solutions to such games. In this section, we consider their solutions under Nash Equilibrium and derive the conditions that lead to honesty being the stable strategy for both players. The detailed proof can be found in Section 5.1.

Theorem 5.1.1. *(honest, honest) is a Nash Equilibrium if*

1. $p_3 \geq q$ *for order (5.1);*

2. $p_3 > q$*, or* $p_3 = q$ *and* $p_1 > qp_2$ *for order (5.2);*

3. $p_2 > q$, or $p_2 = q$ and $p_1 > qp_3$ for order (5.3).

Theorem 5.1.1 can be interpreted as follows: it can be shown that (p_3, p_3, p_2) represents the payoff for both players being honest with preference order (5.1), (5.2) and (5.3) respectively. On the other hand, the average payoff of a successful cheating can be shown as q for all three orders, i.e. one player cheating without any retaliation. Thus, the strict inequality in all three orders implies that honesty is stable if both players being honest has strictly higher payoff than the successful stealing of other's secret. Theorem 5.1.1 is important because making p_3 high by providing and paying for high-quality services, and providing state-of-the-art theft tracking technology such as watermarking to make q small are both reasonable mechanisms in maintaining a viable market. There are situations when thefts of secrets are hard to prove because the stolen secret is never resold, but merely provides knowledge to the thief, such as the case of a software vendor trying to find out new technology from a competitor. Such undetectable thefts make it difficult to keep q small. Additional mechanisms to tackle such scenarios will be described in Section 6.1.

As the payoffs are real values, all the equality cases are of marginal interest. However, due to the symmetric nature of the game, the conditions become very important if cheating behavior is already rampant:

Theorem 5.1.2. *(cheat, cheat) is a Nash Equilibrium if*

1. $qp_2 > p_1$, or $qp_2 = p_1$ and $q > p_3$ for (5.2);

2. $qp_3 > p_1$, of $qp_3 = p_1$ and $q > p_2$ for (5.3).

(Cheat, cheat) is not a Nash Equilibrium for (5.1).

For (5.2) and (5.3), qp_2 and qp_3 respectively represent the average payoff of mutual theft while p_1 represents the payoff of both players losing their secrets. Thus, cheating is a NE if mutual theft has strictly higher utility than losing one's secret. If the two utilities are equal, cheating is still a NE if a successful theft has higher utility than both being honest.

For a poorly developed and managed market, it is quite possible that the majority of the population has already engaged in dishonest behaviors. Theorem 5.1.2 shows that it is very difficult to turn things around because the condition to maintain the cheating behaviors is easily satisfied: the sole reason of the existence of a marketplace in providing privacy-preserving computation is that the participants value the privacy of their data. This means that losing those data can cause significant harm and p_1 must be very small. Despite efforts of making q small, the low to medium costs of collusion could keep p_3 for (5.2) or p_2 for (5.3) significantly higher than p_1. There are two lessons to be learned from the point of view of mechanism design: first, it is important to maintain honesty as majority by growing the initial market with substantial subsidy. Second, additional mechanisms are required to make collusion harder and they will be discussed in Sections 6.1 and 6.2.

Finally, when none of the above conditions are met, as in the case when the value of a secret cannot be a-priori determined, there could be a robust fraction of cheating behaviors in the population based on the following Corollary.

Corollary 5.1.3. *If none of the conditions in Theorems 5.1.1 and 5.1.2 are met, the NE is a mixed strategy.*

The proofs of Theorems 5.1.1 and 5.1.2 as well as Corollary 5.1.3 can be found in the Appendix. Simulation results demonstrating different NE's can be found in Section 7.2.

Proofs of Theorems 5.1.1, 5.1.2, and 5.1.3

Preference order (5.1)

We first consider the case of C $\succ$ D $\succ$ E $\succ$ B $\succ$ A. The normal form game can be described by the payoff matix 5.3.

Table 5.3: Payoff matrix for order (5.1)

		V	
		Honest	Cheat
U	Honest	(p_3, p_3)	$(1-q)(p_0, p_0)$ $+q(p_2, p_4)$ $= (qp_2, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_2)$ $= (q, qp_2)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_1, p_1)$ $= (q^2p_1, q^2p_1)$

The two-tuple in each entry indicates the average payoffs of U and V when adopting the row and column strategies respectively. In the context of a population game, cheating would be a NE if (a) $q^2p_1 > qp_2$ or (b) $q^2p_1 = qp_2$ and $q > p_3$. As $0 \leq p_1, q \leq 1$, neither condition is valid and cheating can never be a NE. Honesty would be a NE if (c) $p_3 > q$ or (d) $p_3 = q$ and $qp_2 > q^2p_1$. As $qp_2 > q^2p_1$ is always true, we have the following conclusion: **Honesty is a NE for both U and V if $\mathbf{p_3 \geq q}$.** When the theft is *undetectable*, i.e. $p_3 < q$, it can be shown that the following mixed strategy constitutes a NE:

$$h_u = h_v = \frac{1}{\frac{q-p_3}{q(p_2-qp_1)} + 1} \tag{5.4}$$

where h_u and h_v are the honest fraction of U and V respectively. Unfortunately, this situation will undoubtedly occur in real life. It is thus important to incorporate additional mechanisms to deter cheating behaviors.

Preference order (5.2)

In this subsection we examine the case of C $\succ$ D $\succ$ B $\succ$ E $\succ$ A. Actually we need to examine only the interpretation of B $\succ$ E, as others remain the same. The motivation behind such an order is that U now gets more information – the secret of V – in outcome "B" than in outcome "E", although in both cases U loses his/her own secret. The payoff matrix is described in Table 5.4.

Table 5.4: Payoff matrix for order (5.2)

		V	
		Honest	Cheat
U	Honest	(p_3, p_3)	$(1-q)(p_0, p_0)$ $+q(p_1, p_4)$ $= (qp_1, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_1)$ $= (q, qp_1)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_2, p_2)$ $= (q^2p_2, q^2p_2)$

To make the honest strategy a NE, we must have either (a) $p_3 > q$, or (b) $p_3 = q$ and $qp_1 > q^2p_2$, i.e. $p_1 > qp_2 = p_2p_3$. For cheating to be a NE, either (c) $q^2p_2 > qp_1$, i.e. $qp_2 > p_1$, or (d) $q^2p_2 = qp_1$ and $q > p_3$, i.e. $p_1 > p_2p_3$ and $q > p_3$. While the equality constraints may be hard to achieve in real-life, condition (c) is possible with a high enough q. Thus, cheating is still possible. If none of the above conditions are satisfied, the NE is a mixed strategy with the honest fraction as follows:

$$h_u = h_v = \frac{1}{\frac{q-p_3}{q(p_1-qp_2)} + 1} \tag{5.5}$$

Preference order (5.3)

In this subsection we examine the case of C ≻ B ≻ D ≻ E ≻ A. With B ≻ D, we have the situation that both cheating is preferred over both being honest. The payoff matrix is shown in Table 5.5.

Table 5.5: Payoff matrix for order (5.3)

		V	
		Honest	Cheat
U	Honest	(p_2, p_2)	$(1-q)(p_0, p_0)$ $+q(p_1, p_4)$ $= (qp_1, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_1)$ $= (q, qp_1)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_3, p_3)$ $= (q^2p_3, q^2p_3)$

To make the honest strategy a NE, either (a) $p_2 > q$, or (b) $p_2 = q$ and $qp_1 > q^2p_3$, i.e. $p_1 > qp_3 = p_2p_3$, which are possible to obtain. For cheating to be a NE, either (c) $q^2p_3 > qp_1$, i.e. $qp_3 > p_1$, or (d) $q^2p_3 = qp_1$ and $q > p_2$, i.e. $p_1 = qp_3 > p_2p_3$. Note that conditions (b) and (d) are the same except that (b) requires $p_2 = q$ which is difficult to sustain, so the situation is that both populations of honesty and cheating coexist. The honest fraction is expressed in the general solution:

$$h_u = h_v = \frac{1}{\frac{q-p_2}{q(p_1-qp_3)} + 1} \tag{5.6}$$

5.2 User-Vendor Bayesian Games

In the previous section, we assume both players share the same preference order. This assumption enables us to use relatively straightforward analysis to compute the Nash Equilibrium. However, in many situations, customers U and V may not be able to tell the other player's preferences. The preference order of a player may change depending on the secret data used in the computation. A player open to the possibility of cheating may also adopt a different preference order, and wants to keep this information private. Such unknown private information of the opponent can be viewed as the opponent's *type* used in the Bayesian game as described in Section 4.1. Of course, a player knows his/her own type, or in this case, the preference order. Interactions between players with private types known only to him/herself should be taken into account in designing anti-collusion mechanisms. This is especially necessary during the initial stage of the secure MPC marketplace in question where customers are mostly new.

In this section, we analyze all possible Bayesian Games between U and V derived from our framework from the perspective of U, since V's conclusion is analogous. There are a total of three Bayesian Games according to U's three different possible preference orders, i.e. (5.1), (5.2), and (5.3). Instead of Nash Equilibrium, the analysis will focus on strongly dominant equilibria as they are independent of the prior belief of the opponent's type. Recall that $0 = p_0 < p_1 < p_2 < p_3 < p_4 = 1$ stands for the five different utilities for the five outcomes in a specific preference order. Using the solution concept of dominant strategy equilibrium introduced in Definition 4.1.4 and 4.1.5, we obtained the following results.

Theorem 5.2.1. *For the Bayesian User-Vendor game, the conditions for honesty to be a strongly dominant strategy for a player regardless of his preference order are*

$$\begin{cases} p_1 > qp_3 \\ p_2 > q. \end{cases} \tag{5.7}$$

The detailed proof of Theorem 5.2.1 can be found in Appendix 5.2. Simulation results demonstrating different Bayesian Games can be found in Section 7.2. Compared with the symmetric case in Theorem 5.1.1, we can see that the Bayesian result is dominated by the conditions required by the preference order 5.3. The reason is that this preference order represents the lowest collusion cost and the conditions of a strongly dominant strategy must consider the worst case scenarios. Once again, the key to deterring collusion is to keep q as low as possible or to make detection of a theft highly robust. However, as pointed out earlier, some collusion attacks may not be detectable at all and as such, retaliation mechanism alone is insufficient. Thus, we have designed further mechanisms to reinforce the anti-collusion efforts, which are described in the following sections. Finally, we skip the Bayesian analysis for cheating as a dominant strategy because it is very similar but of less interest in building a sustainable market.

Proofs of User-Vendor Bayesian Games

We describe the first User-Vendor Bayesian Games in details, since the other two Bayesian Games are to be analyzed in a very similar way. Let us denote preferences

(5.1), (5.2), and (5.3) as $x, y,$ and, z in this section. Denote Θ_U as the private type of player U. The private type in our framework is the choice of one's preference order.

U has type $\Theta_U = x$

First, suppose the type of U is $\Theta_U = x$. To U, V's type set is $\Theta_V = \{x, y, z\}$, which records all three possible types of V. The remaining problem for U is to guess *a priori* how likely each type of V will be. Given U's own type $\Theta_U = x$, denote U's belief probabilities over V's possible types as $\alpha(x|x) = r$, $\alpha(y|x) = s$, and $\alpha(z|x) = 1 - r - s = t$, where $0 \leq r, s \leq 1$. The utility functions for Bayesian Games when U has type x are defined by three type games, formulated in Tables 5.6, 5.7, 5.8. They represent interactions between $\Theta_U = x$ and $\Theta_V = x$, $\Theta_U = x$ and $\Theta_V = y$, and $\Theta_U = x$ and $\Theta_V = z$, respectively.

Table 5.6: Bayesian Game for $\Theta_U = x$ against $\Theta_V = x$

		V	
		Honest	Cheat
U	Honest	(p_3, p_3)	$(1-q)(p_0, p_0)$ $+q(p_2, p_4)$ $= (qp_2, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_2)$ $= (q, qp_2)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_1, p_1)$ $= (q^2p_1, q^2p_1)$

Table 5.7: Bayesian Game for $\Theta_U = x$ against $\Theta_V = y$

		V	
		Honest	Cheat
U	Honest	(p_3, p_3)	$(1-q)(p_0, p_0)$ $+q(p_2, p_4)$ $= (qp_2, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_1)$ $= (q, qp_1)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_1, p_2)$ $= (q^2p_1, q^2p_2)$

The utilities for U are denoted as $U_{U,\Theta_U=x}(S_U; S_V)$. It is the total expected utility for U under his/her type x when U plays his/her strategy S_U against V's various strategy profile S_V. S_V has three component strategies $(S_{V_x}, S_{V_y}, S_{V_z})$, each representing a strategy for a type of V. For example, $U_{U,\Theta_U=x}(H; H, H, H)$ means U plays

Table 5.8: Bayesian Game for $\Theta_U = x$ against $\Theta_V = z$

		V	
		Honest	Cheat
U	Honest	(p_3, p_2)	$(1-q)(p_0, p_0)$ $+q(p_2, p_4)$ $= (qp_2, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_1)$ $= (q, qp_1)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_1, p_3)$ $= (q^2 p_1, q^2 p_3)$

H and V plays strategy H for type x, H for type y, and H for type z. Then the utility of U under the above strategies is calculated as follows,

$$\begin{aligned} U_{U,\Theta_U=x}(H; H, H, H) = {} & ru_U(x, x; H, H) \\ & + su_U(x, y; H, H) \\ & + tu_U(x, z; H, H) \\ = {} & (r + s + t)p_3 = p_3 \end{aligned} \tag{5.8}$$

where $u_U(x, x; H, H)$ means the utility for U in the type game $\Theta_U = x$ against $\Theta_V = x$, when the matching strategies are (H, H).

Similarly, we can proceed to calculate all utilities for the Bayesian Games of $\Theta_U = x$, summarized in Table 5.9.

Table 5.9: Utilities in Bayesian Games of $\Theta_U = x$

Strategy Profile	$U_{U,\Theta_U=x}(S_U = H; S_V)$	$U_{U,\Theta_U=x}(S_U = C; S_V)$
$(S_U; H, H, H)$	p_3	q
$(S_U; H, H, C)$	$rp_3 + sp_3 + tqp_2$	$rq + sq + tq^2p_1$
$(S_U; H, C, H)$	$rp_3 + sqp_2 + tp_3$	$rq + sq^2p_1 + tq$
$(S_U; H, C, C)$	$rp_3 + sqp_2 + tqp_2$	$rq + sq^2p_1 + tq^2p_1$
$(S_U; C, H, H)$	$rqp_2 + sp_3 + tp_3$	$rq^2p_1 + sq + tq$
$(S_U; C, H, C)$	$rqp_2 + sp_3 + tqp_2$	$rq^2p_1 + sq + tq^2p_1$
$(S_U; C, C, H)$	$rqp_2 + sqp_2 + tp_3$	$rq^2p_1 + sq^2p_1 + tq$
$(S_U; C, C, C)$	$rqp_2 + sqp_2 + tqp_2$	$rq^2p_1 + sq^2p_1 + tq^2p_1$

The solution concept we are to use is the strong dominant strategy equilibrium, introduced in Definitions 4.1.4 and 4.1.5. The solution is in such a strong sense that it results in a stable choice of strategy of one player regardless of what the other player chooses to play. Specifically for our User-Vendor Bayesian Games, the ideal strong dominant strategy would be staying honest, summarized in Table 5.10.

Table 5.10: Ideal Solutions for Bayesian Games for All Possible Θ_U

Honesty	vs	**Cheating**
$U_{U,\Theta_U}(H; H, H, H)$	$>$	$U_{U,\Theta_U}(C; H, H, H)$
$U_{U,\Theta_U}(H; H, H, C)$	$>$	$U_{U,\Theta_U}(C; H, H, C)$
$U_{U,\Theta_U}(H; H, C, H)$	$>$	$U_{U,\Theta_U}(C; H, C, H)$
$U_{U,\Theta_U}(H; H, C, C)$	$>$	$U_{U,\Theta_U}(C; H, C, C)$
$U_{U,\Theta_U}(H; C, H, H)$	$>$	$U_{U,\Theta_U}(C; C, H, H)$
$U_{U,\Theta_U}(H; C, H, C)$	$>$	$U_{U,\Theta_U}(C; C, H, C)$
$U_{U,\Theta_U}(H; C, C, H)$	$>$	$U_{U,\Theta_U}(C; C, C, H)$
$U_{U,\Theta_U}(H; C, C, C)$	$>$	$U_{U,\Theta_U}(C; C, C, C)$

Recall $0 \leq r, s \leq 1$, $t = 1 - r - s$, and $0 = p_0 < p_1 < p_2 < p_3 < p_4 = 1$. To achieve the goal of making the pure honest strategy dominant expressed in Table 5.10, the condition would be to have $p_3 > q$. This can be verified by comparing the utilities of the two strategies, "H" and "C", for U under all possible opponent strategy profiles in the two columns in Table 5.9.

Analogously, we can find conditions for the honest strategy under U's two remaining Bayesian Games when U's types are y and z.

U has type $\Theta_U = y$

For the Bayesian Games of $\Theta_U = y$, the three type of games are listed in Tables 5.11, 5.12, and 5.13.

Table 5.11: Bayesian Game for $\Theta_U = y$ against $\Theta_V = x$

		V	
		Honest	Cheat
U	Honest	(p_3, p_3)	$(1-q)(p_0, p_0)$ $+q(p_1, p_4)$ $= (qp_1, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_2)$ $= (q, qp_2)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_2, p_1)$ $= (q^2p_2, q^2p_1)$

The utilities for U under $\Theta_U = y$ are summarized in Table 5.14.

Table 5.12: Bayesian Game for $\Theta_U = y$ against $\Theta_V = y$

		V	
		Honest	Cheat
U	Honest	(p_3, p_3)	$(1-q)(p_0, p_0)$ $+q(p_1, p_4)$ $= (qp_1, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_1)$ $= (q, qp_1)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_2, p_2)$ $= (q^2p_2, q^2p_2)$

Table 5.13: Bayesian Game for $\Theta_U = y$ against $\Theta_V = z$

		V	
		Honest	Cheat
U	Honest	(p_3, p_2)	$(1-q)(p_0, p_0)$ $+q(p_1, p_4)$ $= (qp_1, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_1)$ $= (q, qp_1)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_2, p_3)$ $= (q^2p_2, q^2p_3)$

Table 5.14: Utilities in Bayesian Game, $\Theta_U = y$

Strategy Profile	$U_{U,\Theta_U=y}(S_U = H; S_V)$	$U_{U,\Theta_U=y}(S_U = C; S_V)$
$(S_U; H, H, H)$	p_3	q
$(S_U; H, H, C)$	$rp_3 + sp_3 + tqp_1$	$rq + sq + tq^2p_2$
$(S_U; H, C, H)$	$rp_3 + sqp_1 + tp_3$	$rq + sq^2p_2 + tq$
$(S_U; H, C, C)$	$rp_3 + sqp_1 + tqp_1$	$rq + sq^2p_2 + tq^2p_2$
$(S_U; C, H, H)$	$rqp_1 + sp_3 + tp_3$	$rq^2p_2 + sq + tq$
$(S_U; C, H, C)$	$rqp_1 + sp_3 + tqp_1$	$rq^2p_2 + sq + tq^2p_2$
$(S_U; C, C, H)$	$rqp_1 + sqp_1 + tp_3$	$rq^2p_2 + sq^2p_2 + tq$
$(S_U; C, C, C)$	$rqp_1 + sqp_1 + tqp_1$	$rq^2p_2 + sq^2p_2 + tq^2p_2$

It is when $p_3 > q$ and $p_1 > qp_2$ that the honest strategy becomes dominant against all possible strategy profiles of V. As such, the goal of making the pure honest strategy dominant expressed in Table 5.10 is fulfilled. This can be verified by comparing the utilities of the two strategies, "H" and "C", for U under all possible opponent strategy profiles in the two columns in Table 5.14.

U has type $\Theta_U = z$

Lastly, let's examine the Bayesian Games of $\Theta_U = z$. The three type games are presented in Tables 5.15, 5.16, and 5.17.

Table 5.15: Bayesian Game for $\Theta_U = z$ against $\Theta_V = x$

		V	
		Honest	Cheat
U	Honest	(p_2, p_3)	$(1-q)(p_0, p_0)$ $+q(p_1, p_4)$ $= (qp_1, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_2)$ $= (q, qp_2)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_3, p_1)$ $= (q^2p_3, q^2p_1)$

Table 5.16: Bayesian Game for $\Theta_U = z$ against $\Theta_V = y$

		V	
		Honest	Cheat
U	Honest	(p_2, p_3)	$(1-q)(p_0, p_0)$ $+q(p_1, p_4)$ $= (qp_1, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_1)$ $= (q, qp_1)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_3, p_2)$ $= (q^2p_3, q^2p_2)$

Table 5.17: Bayesian Game for $\Theta_U = z$ against $\Theta_V = z$

		V	
		Honest	Cheat
U	Honest	(p_2, p_2)	$(1-q)(p_0, p_0)$ $+q(p_1, p_4)$ $= (qp_1, q)$
	Cheat	$(1-q)(p_0, p_0)$ $+q(p_4, p_1)$ $= (q, qp_1)$	$(1-q^2)(p_0, p_0)$ $+q^2(p_3, p_3)$ $= (q^2p_3, q^2p_3)$

The utilities for U under $\Theta_U = z$ are summarized in Table 5.18.

Table 5.18: Utilities in Bayesian Game, $\Theta_U = z$

Strategy Profile	$U_{U,\Theta_U=y}(S_U = H; S_V)$	$U_{U,\Theta_U=y}(S_U = C; S_V)$
$(S_U; H, H, H)$	p_2	q
$(S_U; H, H, C)$	$rp_2 + sp_2 + tqp_1$	$rq + sq + tq^2p_3$
$(S_U; H, C, H)$	$rp_2 + sqp_1 + tp_2$	$rq + sq^2p_3 + tq$
$(S_U; H, C, C)$	$rp_2 + sqp_1 + tqp_1$	$rq + sq^2p_3 + tq^2p_3$
$(S_U; C, H, H)$	$rqp_1 + sp_2 + tp_2$	$rq^2p_3 + sq + tq$
$(S_U; C, H, C)$	$rqp_1 + sp_2 + tqp_1$	$rq^2p_3 + sq + tq^2p_3$
$(S_U; C, C, H)$	$rqp_1 + sqp_1 + tp_2$	$rq^2p_3 + sq^2p_3 + tq$
$(S_U; C, C, C)$	$rqp_1 + sqp_1 + tqp_1$	$rq^2p_3 + sq^2p_3 + tq^2p_3$

The conditions $p_2 > q$ and $p_1 > qp_3$ makes the honest strategy the dominant one against all possible strategy profiles of V. Hence, the goal of making the pure honest strategy dominant expressed in Table 5.10 is fulfilled. This can be verified by comparing the utilities of the two strategies, "H" and "C", for U under all possible opponent strategy profiles in the two columns in Table 5.18.

In conclusion, for all three possible types of U and all possible strategy profiles of V, the conditions for honesty to be dominant are

$$\begin{cases} p_1 > qp_2 \\ p_1 > qp_3 \\ p_3 > p_2 > q \end{cases} \tag{5.9}$$

It can be simplified as

$$\begin{cases} p_1 > qp_3 \\ p_2 > q. \end{cases} \tag{5.10}$$

Chapter 6 Collusion Deterrence Mechanisms for Computing Agents

6.1 Customer-Agent Game

For U to be successful in stealing V's secret, U must be able to convince t or more agents to collude with him/her. A collusion attack can thus be avoided if the agents refuse to collude. Such a collusion avoidance tactic is highly desirable as it does not rely on after-the-fact retaliation that hinges on the detection of the theft. To deter agents from colluding with customers, we introduce honest undercover customers (police) that attempt to collude with agents. A cheating agent who is reported by either a police customer or an honest customer will be paid nothing and will be banned from the system. This worst outcome is denoted by v_0. Let λ be the conditional probability of encountering police given a colluding request from the customer. For the case when there is no colluding request from the customer, the conditional probability of encountering police is 0 as a police customer is assumed to always attempt to collude with the agents. The payoff matrix for the *customer-agent* game is given in Table 6.1.

Table 6.1: Customer-Agent Game

		A	
		Honest	Cheat
U	Honest	(v_1, v_1)	(v_0, v_0)
	Cheat	$\lambda \cdot (v_1, v_1)+$ $(1-\lambda)(v_1, v_1)$ $= (v_1, v_1)$	$\lambda \cdot (v_0, v_0)+$ $(1-\lambda)(v_2, v_2)$ $= (1-\lambda, 1-\lambda)$

The normalized payoffs are represented as $0 = v_0 < v_1 < v_2 = 1$, and are assumed, for simplicity, to be the same for both user and agents. (honest, honest) is clearly a NE of this game. In fact, honesty becomes a strongly dominant strategy for the agent if $v_1 > 1 - \lambda$. This is the most desirable outcome, brought on by a large λ or a significant presence of police. On the other hand, $v_1 \leq 1 - \lambda$ will make (cheat, cheat) another NE of the game. Such an unfortunate situation will occur when there are not enough police or the payoff for an honest agent is significantly smaller than that of collusion.

The suggested strategy of inserting police to track dishonest agents finds its parallel in real life. In network security, honeypot servers are routinely used to decoy network attacks towards a well-isolated and monitored area so as to collect information that may lead to ultimate apprehension of the attackers. In fact, game theory has been routinely used in analyzing the strategic use and placement of honeypots [55, 139, 130, 79, 26]. Using the hierarchical structure of today's internet, some researchers argued that only a small number of honeypot servers at network core can thwart most network attacks [139]. This points to the interesting possibility of incorporating network topology into our agent-based secure MPC framework to further reduce the number of police customers, a subject worthy of further investigation.

6.2 Censorship

The collusion deterrence games in Chapter 5 focuses on strategies to promote honest behavior among users. However, even if all the users are honest, the computing agents themselves can collude to steal secrets as described the *A3* attacks scenario in Section 4.2. During reconstruction or renormalization, agents are supposed to exchange information with each other. Unlike the *A1* attacks, agents do not have any preexisting side-channels and as such, they might try to collude by sending subliminal messages within the protocol. A trivial solution is to simply prohibit communication among agents. This solution works for only simple computation protocols in which renormalization is not necessary. Verifiable secret sharing [32] does not work for renormalization either, as the dishonest agent has the freedom in setting a few semantically meaningful new shares (say to its IP address) while maintaining perfect reconstruction of its original share.

In this section, we propose a simple censorship scheme to delegate the task of renormalization to a more "trusted" entity. With a properly designed game-based mechanism as described in Sections 5.1 and 5.2, we assume that the customers U and V are deterred from colluding with the agents and use the semi-honest model for their possible adversarial behaviors. However, they still have a strong incentive to safeguard their secrets and keep all of the agents honest. As such, U (or V) can carry out the task of renormalization by injecting fresh noise into the shares to destroy any subliminal messages. The proposed censorship scheme requires processing and routing messages of agents through U and V. Suppose the underlying protocol is the (t, n) SSS where n is the number of computing agents and t is the original threshold of the SSS. The censorship scheme is described in Protocol 1.

Protocol 1 $f_{renormalize}$

```
1: Each agent $A_i$ for $i = 1, \ldots, n$ has a secret share $[u]_i^p$ based on a random $p$-degree
   polynomial with $t - 1 \leq p \leq n - 1$.
2: $U$ selects a uniformly random number $r_U$ and sends its shares $[r_U]_i^t$ to $A_i$.
3: $V$ selects a uniformly random number $r_V$ and sends its shares $[r_V]_i^t$ to $A_i$.
4: $A_i$ computes $[u]_i^p + [r_U]_i^t + [r_V]_i^t \mod m = [u + r_U + r_V]_i^p$ and sends to $U$.
5: For each $i$, $U$ computes $m_{i,j} = [[u + r_U + r_V]_i^p]_j^t$ and sends them to $A_j$ for $j =
   1, \ldots, n$.
6: $A_i$ receives $m_{i,j}$ from $U$ for $j = 1, \ldots, n$ and compute $s_i = \sum_{j=1}^{n} \gamma_j m_{i,j} - [r_U]_i^t -
   [r_U]_i^t \mod m$ where $\gamma_j$'s are the Lagrange interpolation coefficient for polynomial
   of degree $p$.
```

Before presenting the security proof, we discuss the complexity of this protocol. Our protocol has $4n$ invocations of communication in each renormalization compared to $n(n-1)$ in the original one without any collusion-deterrence. The reduction in the number of invocations (for $n > 5$) is due to the fact that the centralization of the responsibility to U enables multiple messages to the same agents be combined in Step 5. Our scheme is different from the mediator solution in [3] as we are using a building block that is different from a secure two-party protocol. While it does involve the participation of a secret holder, our scheme is of much lower complexity as it is only needed for steps like renormalization and output.

Correctness

The correctness of Protocol 3 can be shown as follows: by our covert adversary assumption, the agents will only deviate from the protocol if there is a non-zero probability of launching a successful collusion attack. We will demonstrate later in our security proof that it is impossible to do so. As such, we focus here on the correct execution of the protocol. In step 4, the equation $[u]_i^p + [r_U]_i^t + [r_V]_i^t \mod m = [u + r_U + r_V]_i^p$ holds because $p \geq t - 1$. Thus, the polynomials used to share r_U and r_V can be considered to have degree p with zero leading terms so the additive homomorphism holds. In step 5, U carries out the renormalization step to reduce the degree of the polynomial back to t. In step 6, the first term reconstructs the noisy share based on the renormalization formula (4.2):

$$\begin{aligned}\sum_{j=1}^{n} \gamma_j m_{i,j} &= \sum_{j=1}^{n} \gamma_j [[u + r_U + r_V]_i^p]_i^t \\ &= \left[\sum_{j=1}^{n} \gamma_j [u + r_U + r_V]_i^p\right]_i^t = [u + r_U + r_V]_i^t\end{aligned}$$

Then, using additive homomorphism, A_i obtains $s_i = [u + r_U + r_V]_i^t - [r_U]_i^t - [r_U]_i^t \mod m = [u]_i^t$, which is the desired output of renormalization.

Security

The security proof of Protocol 1 is as follows: we assume that U and V are semi-honest with the explicit goal of detecting any irregularities in agent traffic. The agents hope to achieve collusion by deviating from the protocol and changing the outbound messages, but will only do so if there is a non-zero probability of success. We assume the worst case scenario in that *all agents are interested in a collusion attack and are aware of a common preamble to signal subliminal communication of information such* *their own IP addresses*. The proof follows the well-known simulation paradigm [60] in which the distribution of any inbound messages at each party can be computed based on the party's knowledge obtained through the *ideal functionality*. Since U and V are not involved in the ideal functionality of renormalization, they should not gain any additional information about the underlying secret. As for the agents, the ideal functionality should result in uniformly random shares of the same secret hidden inside a $t-1$ degree polynomial, or termination as a result of malicious behaviors from some agents. Let us analyze the inbound messages received by each party as described in Protocol 3.

V does not receive any messages so there is no security concern. All the agents receive two rounds of independent uniformly random shares - first round are the random shares from U and V and the second round are the renormalized shares from U. They are independent uniformly random shares because no agent receives more than t shares of any secret and it is assumed that both U and V carry out the protocol faithfully. Lastly, U receives inbound messages from all agents. There are two scenarios. For the first scenario, we assume all agents carry out the protocol faithfully and U receives all the random shares of $u + r_U + r_V$. While U can certainly carry out the reconstruction, U cannot learn anything about u due to the presence of uniformly random r_V unknown to U. For the second scenario, all the agents may decide to replace their messages with subliminal messages. While preambles known to all agents can certainly be sent undetected by U, it is impossible for a single agent to send an unique message because U will be able to detect inconsistency among reconstructions based on different subgroups of t shares received. Even if U does not check for any inconsistency, subliminal messages will be destroyed as they are replaced by uniformly generated random shares in Step 5. In either scenario, U does not gain any new knowledge about V's secret. As all the inbound messages to U and the agents are statistically indistinguishable from those from the ideal functionality, we conclude that Protocol 3 is secure.

Chapter 7 Experiments

In this chapter, we first present a comparison in computation efficiency between our Collusion-Deterred SSS (CD-SSS) and other state-of-the-arts computationally secure SMC techniques. Then, we simulate how different strategies might evolve under different conditions in the user-vendor games and the customer-agent game.

7.1 Computational Efficiency of CD-SSS versus GC

We first test the hypothesis that our CD-SSS system provides a more computationally-efficient secure MPC system than other state-of-the-arts GC implementations, including both TASTY [68] and ObliVM [83]. The choice of TASTY and ObliVM is based on their performance and the availability of software. GC is primarily a 2-party secure MPC scheme while our CD-SSS system requires at least 3 agents and 2 customers, user and vendor. As the extra computing resources are not used for parallelization but rather for matching the security access structure, we measure the performance based on the actual wall clock time needed to complete the entire computation.

For the benchmark operations, we have chosen addition, multiplication, and comparison of two encrypted numbers. These three operations are universal and commonly used in literature as benchmarks. For our CD-SSS system, we assume that the input numbers are already in shares and the operations complete with a reconstruction. The implementations of addition and multiplication are straightforward and based on the GNU MP library [64]. While we only use one-level deep multiplication, we have included a renormalization step so that the number is representative for arbitrary number of levels.

Comparison is more complicated and our algorithm is based on our earlier work in [116]. Here we briefly review the procedure. Suppose v and w are two l-bit numbers to be compared and we rely on the radix-2 representations of the numbers. So the bits of v and w are denoted as v_i and w_i for $i = 0, 1, ..., l - 1$. The computation is performed bit by bit from the most significant bit. All the bits are already in shares and the share computation is performed in the prime field $\mathbb{F}_5$. The algorithm uses two state variables b and c to accumulate the intermediate result. Initially, $b := 1$ and $c := v_0 - w_0 + 1$, which would be 2 if $v_0 > w_0$, 0 if $v_0 < w_0$, and 1 otherwise. For the subsequent bits, we perform $b := b \cdot (1 - v_{j-1} \oplus w_{j-1})$ followed by $c := c \cdot (v_j - w_j + 2 - b)$. b is 1 until the corresponding bits from v and w begin to differ, and all the subsequent b's will be 0. $v_j - w_j + 2 - b$ is 1 until the bits start to differ – it will be 0 if $v < w$ and 2 if $w > v$. This leads to the final output $c = 0$ if and only if $v < w$. The above protocol involves multiple random number generations, multiplications, renormalization, and additions.

In the proposed CD-SSS, the user or vendor is responsible for renormalization and reconstructions as discussed in Section 6.2, while the agents perform all the remaining computation. We adopt a number of strategies to expedite the calculations. First, all shared random numbers are pre-generated and distributed among the agents. Second, as communication and synchronization are needed after comparing each bit, we amortize the measurements over a large number of comparison operations in a bitwise fashion so as to minimize the communication overhead. To promote reproducible research, we have made our CD-SSS implementation publicly available at our website.

We compare different techniques in two testbeds. The first testbed is on a virtual 1-Gbps LAN with up to 5 Linux nodes (1G Hz Dual-Core AMD Opteron with 2GB RAM) on Deterlab [17]. The available TASTY software is unable to support the

Table 7.1: Deterlab Experiment

Time per operation	CD-SSS	ObliVM	TASTY
Addition (1024 bits)	0.32 ms	806.4 ms	25.56 ms
Multiplication (1024 bits)	1.17 ms	1.587 s	8.602 s
Compare (16384 bit)	2.50 ms	7.671 s	-

comparison operations on such a wide operand. Table 7.1 shows that CD-SSS is 2 to 3 orders of magnitude faster than the two other GC methods. We also notice that our measurements for ObliVM are significantly slower than the ones reported in the original paper. As such, we attempt to normalize the platform by running CD-SSS on Amazon EC2 computing nodes of types c4.8xlarge, the same computation platform used in ObliVM. The results of ObliVM are directly cited from their original paper [83]. The results are summarized in Table 7.2. CD-SSS remains significantly faster than ObliVM in addition and multiplication. However, the heavy network cost on a shared network, as opposed to the dedicated network in Deterlab, has taken a toll on comparison and the two schemes are much closer. As such, a key goal to design a practical CD-SSS system is the minimization of the number of network invocations.

Table 7.2: Amazon EC2 Experiment

Time per operation	CD-SSS	ObliVM
Addition (1024 bits)	8.1 μs	1.7 ms
Multiplication (1024 bits)	19.9 μs	833 ms
Comparison (16384 bits)	57.3 μs	26 ms

7.2 Simulations of Strategic Behaviors

In this section, we validate the conditions of different games studied in Section 5.1. Instead of Nash Equilibria (NE), we focus our experiments on simulating the evolutionary stable strategy (ESS). While these two solution concepts are based on different conditions, an ESS, if it exists, must coincide with a NE. Also, we believe that this is an appropriate approach because, for the distributed computing platform to be economically viable, it will need to have a large number of customers. Among the customers, those that provide proprietary software services are likely to be reviewed by their clients and dishonest behaviors can result in poor customer ratings, leading to their ultimate demise. Typical consumers who want computing services on their private information will need to use their credit cards to pay for the services.

Again, any suspected cheating behaviors can ruin their credit scores. In other words, while we model cheating versus staying honest as rational behaviors, bad behaviors on a highly-social, well-connected distributed computing marketplace can affect the decisions of other players. As such, we use EGT to model the population of all the customers in this marketplace that are seeking and providing privacy-protected computing services. The goal is to study how pervasive cheating behaviors can be in such a marketplace. In our simulation, we use the replicator dynamic (RD) to simulate the evolution from a given population profile under different conditions. Under RD, the growth rate of the agents using each strategy is proportional to the excess of the strategy's payoff over the average payoff [136]. Our implementations are based on the GameBug simulator [137].

We first illustrate how the system evolves over time for different User-Vendor Games. Each user in the system is randomly matched with a vendor from the same population for cooperations. Recall that q is the non-retaliation probability and p_i is the i-th ranked payoff. At the beginning of the simulation, 90% of the populations are honest. We first test the scenario of preference ranking (5.1), with $q \leq p_3$. The initial population profile evolves very quickly toward the pure-honesty ESS as depicted in Fig. 7.1. In sharp contrast, $q > p_3$ leads to a mixed ESS with $h_u = h_v = 0.8$. Fig. 7.2 shows this evolution in the population whose profile gradually converges to the mixed ESS as marked by the black solid line. Second, we test the scenario using preference ranking (5.2). Under the condition $p_3 > q$, the honest strategy prevails quickly as depicted in Fig. 7.3. Under the alternative honest condition $p_1 > p_2p_3$ and $p_3 = q$, however, the honest behavior evolves very slowly as depicted in Fig. 7.4. Both the conditions $qp_2 > p_1$ and $p_1 > p_2p_3$ and $q > p_3$ lead to the population evolve to cheating, as depicted in Fig. 7.5. Third, we simulate preference ranking (5.3). The honest condition $p_2 > q$ yields a relatively slow system evolution compared to the previous two games, and the condition $p_1 > p_2p_3$ and $p_2 = q$ has a even much slower evolution, as depicted in Fig. 7.6 and 7.7 respectively. The cheating conditions $qp_3 > p_1$ or $p_1 > p_2p_3$ and $q > p_2$ have very similar effects, as depicted in Fig. 7.8.

The Bayesian Game when U is with preference order (5.1) is illustrated in Figure 7.9 under the condition $p_3 > q$. The initial population is divided 50% against 50%, each playing "H" and "C". The belief probabilities are set as $r = s = t = 1/3$. They can be set arbitrarily and do not much affect the result. The system evolves quickly to honest. For the other possible types of U, the honest conditions yields very similar evolution.

Next, we simulate the *Customer-Agent Game*. Consider the case when $v_1 > 1-\lambda$. Setting 50% of the users and agents as honest initially, Fig. 7.10 shows that agents evolve to honesty while users stay at a mixed strategy over time as predicted by the non-strict NE. Second, for the case of $v_1 < 1-\lambda$, Fig. 7.11 shows that the same initial population composition of half cheating and half honest is gradually taken over by cheating which is an ESS.

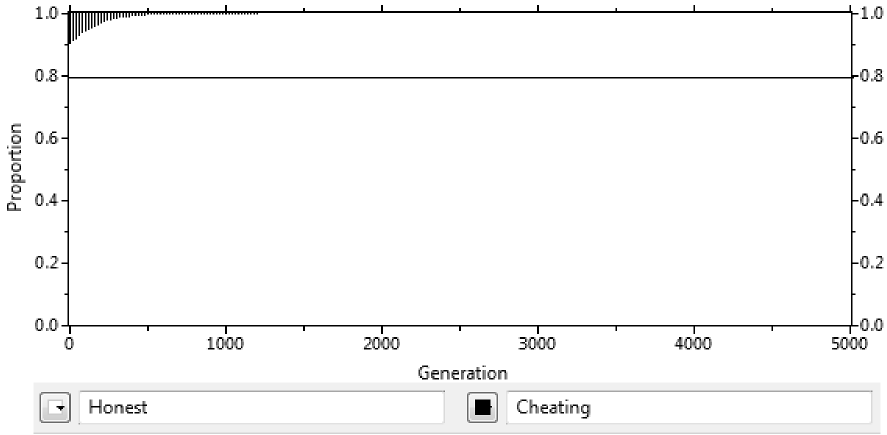

Figure 7.1: Emergent Behavior in User-Vendor Game One when $q < p_3$

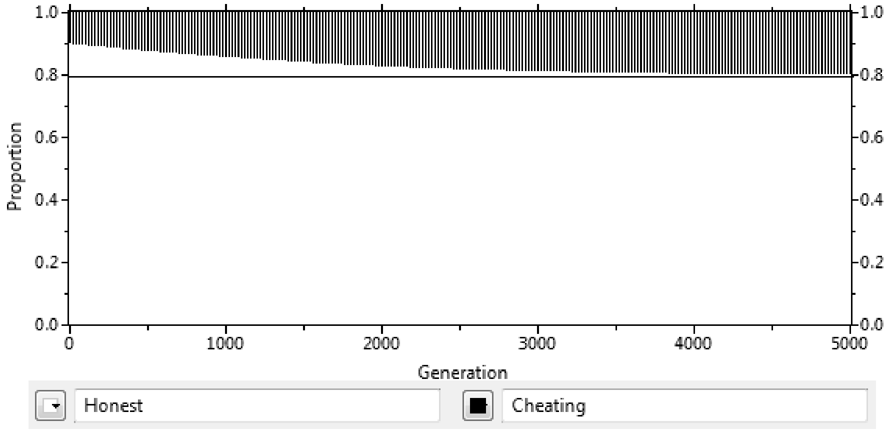

Figure 7.2: Emergent Behavior in User-Vendor Game One when $q > p_3$

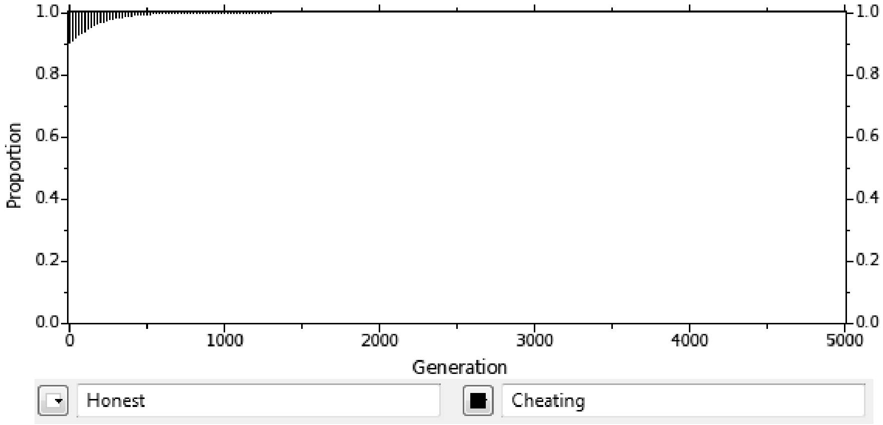

Figure 7.3: Emergent Behavior in User-Vendor Game Two when $q < p_3$

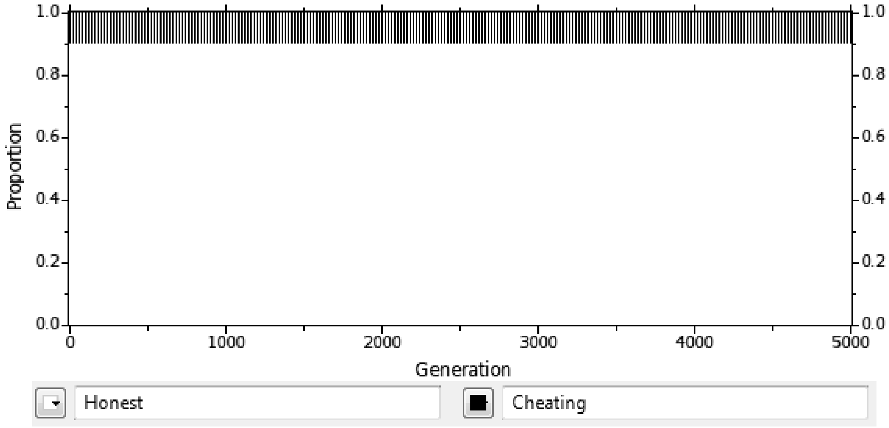

Figure 7.4: Emergent Behavior in User-Vendor Game Two when $p_1 > p_2p_3$ and $p_3 = q$

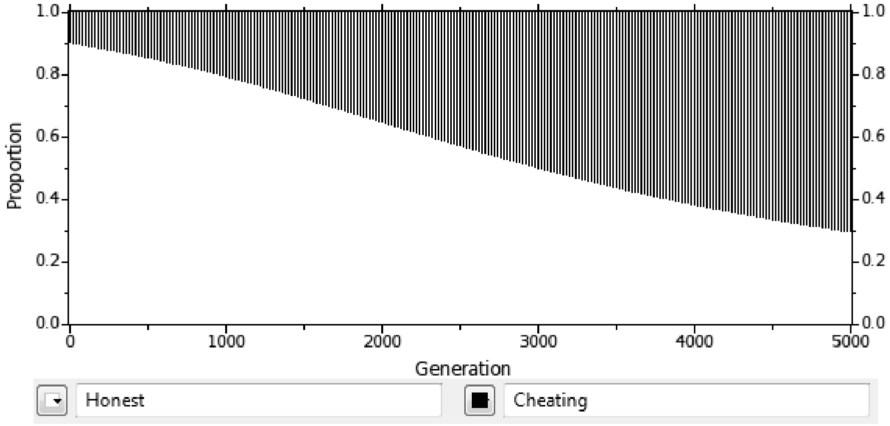

Figure 7.5: Emergent Behavior in User-Vendor Game Two when either $qp_2 > p_1$ or $p_1 > p_2p_3$ and $q > p_3$

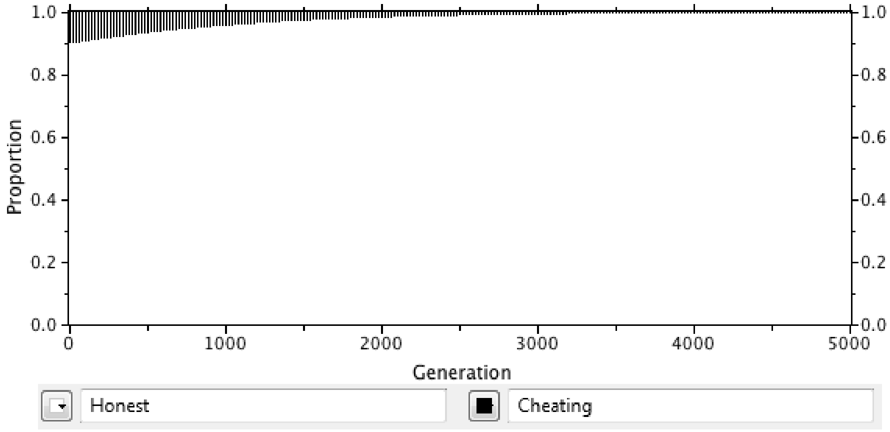

Figure 7.6: Emergent Behavior in User-Vendor Game Three when $q < p_2$

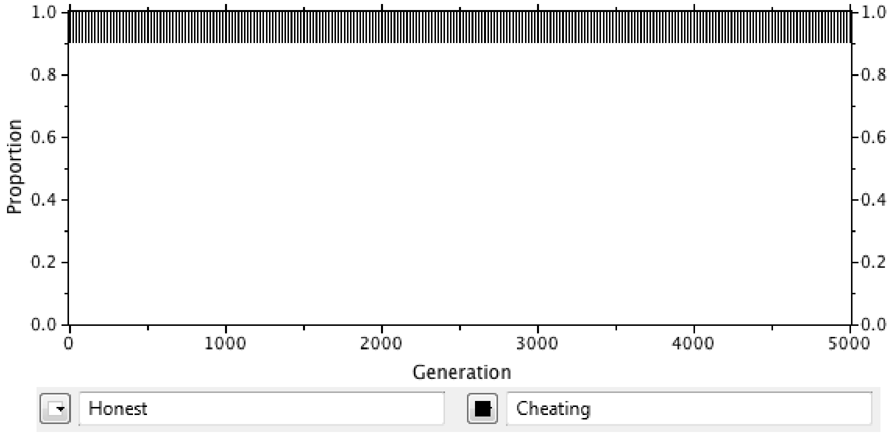

Figure 7.7: Emergent Behavior in User-Vendor Game Three when $p_1 > p_2p_3$ and $p_2 = q$

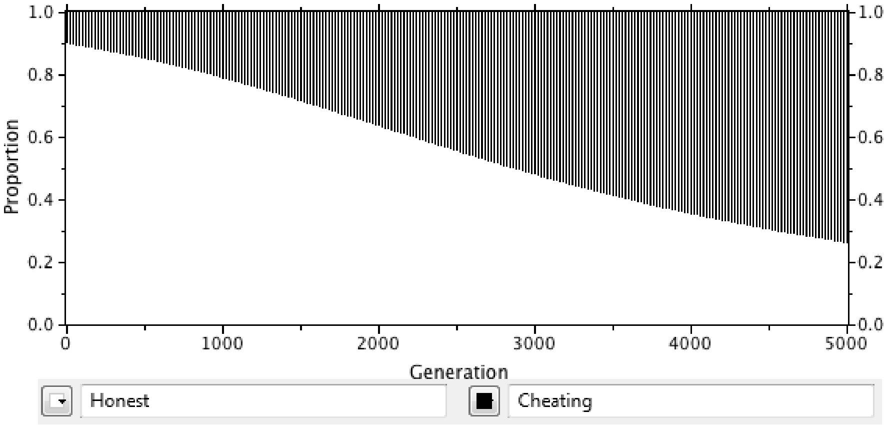

Figure 7.8: Emergent Behavior in User-Vendor Game Three when either $qp_3 > p_1$ or $p_1 > p_2p_3$ and $q > p_2$

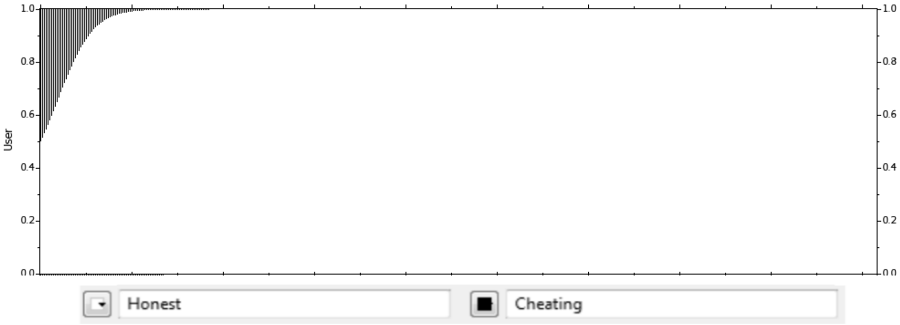

Figure 7.9: Emergent Behavior in Bayesian Game Three when $p_3 > q$

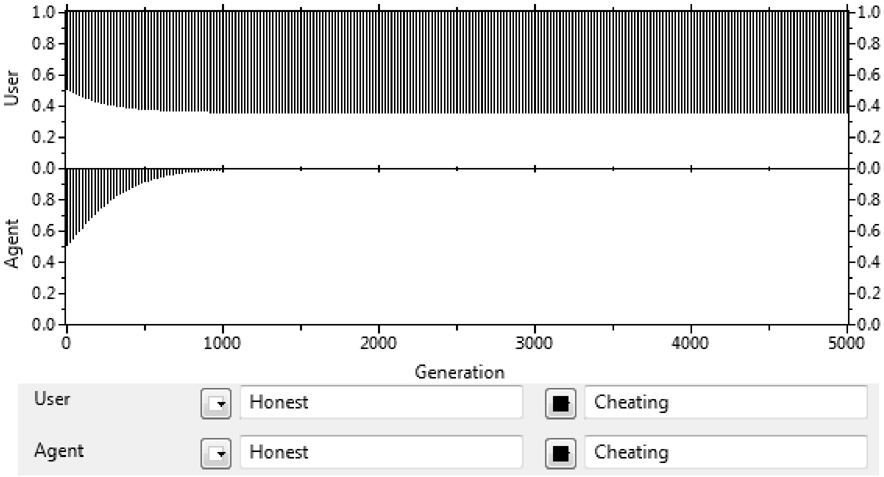

Figure 7.10: Emergent Behavior in the User-Agent game when $p_1 > 1 - \lambda$

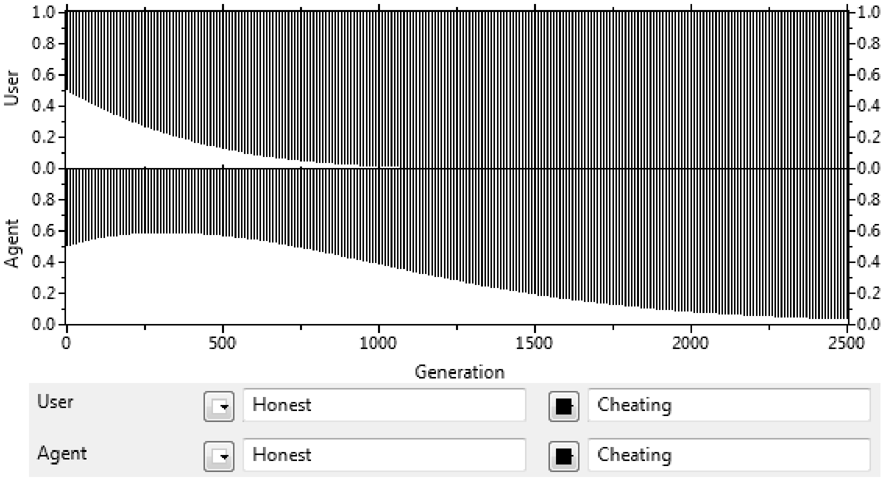

Figure 7.11: Emergent Behavior in the User-Agent game when $p_1 < 1 - \lambda$

Chapter 8 Conclusion and Future Directions

In our construction of the censorship scheme, one of the customers, U or V, is responsible for censoring the messages among computing agents. While the proposed scheme achieves information-theoretic security, it relies on U or V to carry out the part of censorship that is necessary to the whole task. We would like to further reduce the computing load of U and V in an outsourced scenario. This motivates our investigations for new mechanisms that deter collusion among agents while further relieving the burden of U and V. Specifically, we are interested in the outsourced computation where there is only one secret owner U who gives the input to a computing platform.

In this chapter, we briefly describe our initial results of new collusion-deterrence mechanisms as the starting point of future explorations. The basic idea is to have multiple layers of computing agents. The upper level agents can communicate with the immediate lower level agents, but the communication in the reverse direction should be very difficult, thus making all communications one directional. We call it Message Routing Mechanism and analyze its collusion deterrence using the coalition game theory. Finally, we summarize this dissertation and point out future directions.

8.1 Message Routing Mechanism

In order to destroy collusions, we break down the collusion channels between agents. The idea is to isolate agents so that whoever has the input secret shares cannot communicate directly with another agent who also holds a secret input share. Specifically, those who have secret shares do not know each other and whoever routes the communication messages does not know anything of the secret shares. We designed two schemes, called “Proxy” and “Relay”, from the above principle, in the following two subsections respectively. We are interested in the task of $\mathcal{F}_{output}$ (output a reconstructed value as described in (3.2)) and $\mathcal{F}_{renorm}$ (renormalization as described in (4.2)). Since these two tasks involves communications among agents, they directly correspond to the possible attacks described in Section 4.2. The reason is that by the completeness theorems from [16] there are only three types of gates necessary for SSS-based protocols: addition, scaling, and multiplication. All of the three gates would lead to an output of a value, captured in $\mathcal{F}_{output}$, and for the multiplication gate a reduce of degree is necessary which is captured by $\mathcal{F}_{renorm}$. We now describe two types of future Message Routing Mechanisms. The first is called Proxy and the second is called Relay.

Proxy

The Proxy scheme has a set of computing agents S_i for $i = 1, ..., n$ and two other agents P_1 and P_2 acting as proxies. Initially, U distributes input secret shares $[a]_i$ to S_i and P_1 and P_2 have no secret inputs but are responsible for routing messages between S_i whenever needed. Only two proxies are sufficient. Suppose we use a (t, n) SSS. A well-known fact is that $t \leq \lceil n/2 \rceil$. In whatever step involving the proxies, P_1 gets $(t-1)$ shares and P_2 gets the remaining shares, so neither of them can learn the secret. First, whenever the agents need an intermediate public value to be revealed, they call Protocol of Output, described in Protocol (2). Second, suppose at each S_i some intermediate answer $[x]_i^d$ has reached a threshold $d > \lceil \frac{n}{2} \rceil$, so a renormalization is needed. The computation of the final answer cannot be performed by these two protocols. In fact, when all the computation is done, each S_i sends back the share of result $[r]_i$ to U who then use (3.2) to reconstruct the result. Then agents run Protocol of Renormalization, described in Protocol (3). The computational case of three agents with two proxies is depicted in Fig 8.1.

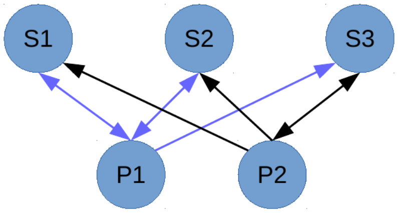

Figure 8.1: Network topology for the Proxy scheme in the operation of renormalization: $S_{1,2}$ send shares to P_1 while S_3 sends to P_2; then $P_{1,2}$ distribute new shares back to all agents.

Protocol 2 $\mathcal{F}_{output}$ implemented in Proxy

```
1: $S_i$ has a share of a public value $[c]_i$, $i \in \{1, ..., n\}$;
2: $S_i$, $i \in \{1, ..., t\}$, sends $[c]_i$ to $P_1$
3: $P_1$ outputs $c$ using Eq. (3.2)
4: $P_1$ sends $c$ to every $S_i$;
5: $S_{1,...,n}$ continue to run the remaining program if any.
```

Protocol 3 $\mathcal{F}_{renorm}$ implemented in Proxy

```
1: $[x]_i^d$ for $i = 1, \dots, \lceil \frac{n}{2} \rceil$ are sent to $P_1$ and the rest to $P_2$;
2: $P_1$ and $P_2$ generate the renormalized shares as follows:
       $$[[x]_i^d]_j^t = \sum_{k=1}^{t-1} \alpha_k j^k + [x]_i^d \bmod m, \quad (8.1)$$
   where $\alpha_k$ are random numbers unknown to any $S_i$;
3: $[[x]_i^d]_j^t$ are then sent to $S_j$ for $i, j = 1, 2, \dots, n$;
4: $S_j$ computes the new $[x]_j^t$ using Eq. (3.2);
5: $S_{1,...,n}$ continue to run the remaining program if any.
```

In the *Proxy* process, S_i knows the shares $[a]_i$ but does not know IP of any S_j for $i \neq j$. P_i does not have the initial $[a]_i$. By virtue of the underlying SSS scheme, all data at any participants are random numbers.

Relay

The scheme of Proxy is suitable for cloud computing where three computing agents and two proxies are sufficient. If we deploy the P2P setting where millions of peers acting as computing agents, we can implement the Relay scheme. The following Relay deploys three agents in parallel at a time, and it can be extended to the situation of n agents at a time straightforwardly.

Suppose agents S_1, S_2, and S_3 get initial input secret shares $[a]_i$ for $i = 1, 2, 3$ from U. First, for outputting a public number c for the task $\mathcal{F}_{output}$, the agents call Protocol (4). The computational model and data flow is depicted in Fig. 8.2. Note that all S_4, S_5, and S_6 get are random shares $[c]_i$ but no initial secret shares $[a]_i$, hence the direct interactions among S_4, S_5, and S_6 do not matter.

Second, Protocol (5) describes computing $\mathcal{F}_{renorm}$, with the data flow and the computational model in Fig. (8.3). Here again S_1, S_2, and S_3 who has the initial secret shares cannot communicate with each other for collusion, nor can S_4, S_5, and S_6. Note that S_i for $i = 4, ..., 9$ does not have secret shares but only random numbers. Finally, U gets shares of the final result from a final set of agents and reconstructs the result. Note that if S_1, S_2, and S_3 do not possess any secret shares, they can do any renormalization and output without resorting to a next set of agents. This observation can be used to simply specific SOC protocols.

Protocol 4 $\mathcal{F}_{output}$ implemented in Relay

```
$S_i$ has a share of a public value $[c]_i$, $i \in \{1, 2, 3\}$;
$[c]_i$, $i \in \{1, 2, 3\}$, is sent to $S_i$ for $i \in \{4, 5, 6\}$ respectively;
$S_{4,5,6}$ communicate to reconstruct $c$ using Eq. (3.2);
$S_{4,5,6}$ continue to run the remaining program if any.
```

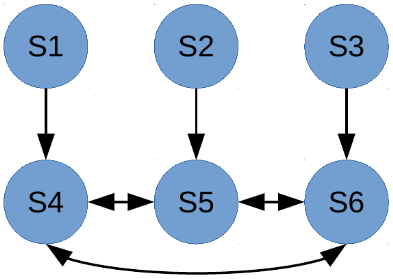

Figure 8.2: Network topology for the operation of output

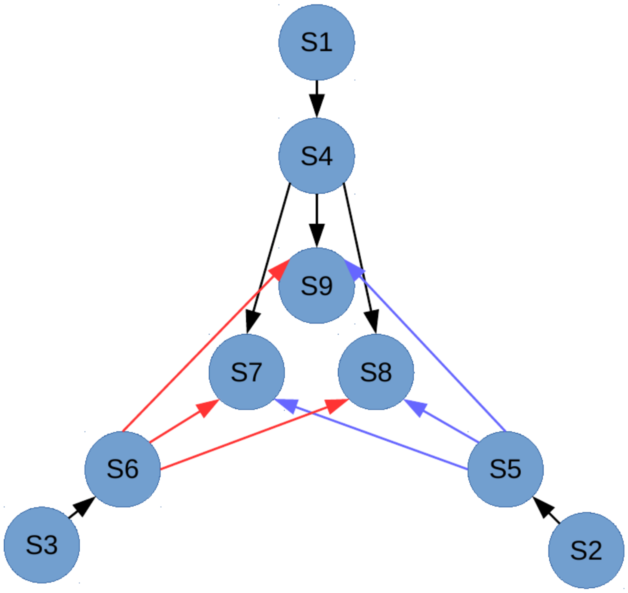

Figure 8.3: Network topology for the operation of renormalization

Protocol 5 $\mathcal{F}_{renorm}$ implemented in Relay

```
1: $S_{1,2,3}$ have $[x]_i^d$ for $i \in \{1,2,3\}$ and $d > \lceil \frac{n}{2} \rceil$;
2: $S_i$ sends $[x]_i^d$ for $i \in \{1,2,3\}$ to $S_{4,5,6}$ respectively;
3: $S_{4,5,6}$ use Eq. (8.1) to generate new shares which are received by $S_{7,8,9}$;
4: $S_{7,8,9}$ use Eq. (3.2) to get the renormalized $[x]_i^t$;
5: $S_{7,8,9}$ continue to run the remaining program if any.
```

Regarding the complexity of our schemes compared with the original non-collusion-deterred $n-$agent SSS, the Proxy will introduce $2n$ more invocations of communication in parallel in each round of renormalization, and two more invocations of communication in parallel in each round of output. For Relay, it will incur $2n$ more invocations of communication in parallel for each round of renormalization and n more in perallel for each output. Compared to SSS with collusion-deterrence [134], Proxy has the same communication complexity and Relay has $(n-2)$ more invocations in parallel in each output. At a first glance, Relay is more complex than Proxy. However, the anti-collusion analysis in Section 8.1 will show it is more difficult for

Table 8.1: Coalition Game in Proxy

Coalition	**Value**
$\{S_1, P_1, S_2\}$	1
All other $T \subseteq N$	0

agents in Relay to collude..

An example SOC task using our proposed schemes is to compute LTZ, the "comparison with zero" protocol [28]. Comparison is a performance bottleneck for SSS-based protocols [28]. The LTZ algorithm for the Relay is modified from [28] so as to adapt to our Relay. For Proxy, LTZ is the same as that for [134] except that the proxies now take place for renormalizations and reconstructions of intermediate values.

Coalition Game

First, we analyze the situation in Proxy. There is a game $G_P = (N, \mathbf{v})$ defined by a set of players $N = \{S_1, P_1, S_2\}$. Refer to Fig. 8.1 for the connections. Denote $T \subseteq N$ as any subset of players. $S_{1,2}$ have the secret shares but no idea of the IP of each other. If P_1 is ignorant of the collusion request from $S_{1,2}$ or refuses the request, then the collusion chain is broken. So P_1 is the veto player in G_P. The characteristic function $\mathbf{v}$ of G_P is defined as shown in Tab. 8.1. The value has been normalized where "1" means the highest possible value of collusion (for example, by selling the secret and then divide the revenue among colluders) and "0" means nothing is achieved. Regarding the solution of G_P, denote the payoff vector as $\upsilon = (\upsilon_{S_1}, \upsilon_{P_1}, \upsilon_{S_2})$. Hence, the *core* is $\upsilon_c = (0, 1, 0)$, the *nucleolus* is $\upsilon_n = (0, 1, 0)$, and the SV is $\upsilon_S = (1/3, 1/3, 1/3)$.

The interpretation of G_P from S_1's point of view includes: (1) S_1 must pay to P_1 what ever price P_1 asks as suggested by the *core* and the *nucleolus.* (2) Isolated from each other, S_1 has no information of the coordination of actions of S_2 and both are advised to be honest because of the zero payoff from the *core* and the *nucleolus.* (3)

Table 8.2: Coalition Game in Output of Relay

Coalition	**Value**
$\{S_1, S_2, S_4, S_5\}$	1
All other $I \subseteq J$	0

Table 8.3: Coalition Game in Renormalization of Relay

Coalition	**Value**
$\{S_1, S_2, S_4, S_5, S_7\}$	1
All other $H \subseteq K$	0

The only cooperation solution is SV. If S_2 also joins, it is fair to give P_1 only 1/3 of the value according to the SV. This thought will be a key point of future direction.

Second, let's turn to the Relay scheme. There are two games, one for *output* and another for *renormalization*.

Refer to Fig. 8.2 of *output* and consider the potential collusion between S_1, S_2, since the situation is the same for any pair of agents from $S_{1,2,3}$. Denote the game in *output* as $G_O = (J, \mathbf{w})$ where $J = \{S_1, S_2, S_4, S_5\}$. The S_4, S_5 are veto players because of their midway positions in the collusion chain. The characteristic function $\mathbf{w}$ is defined in Tab 8.2. Denote the payoff vector for G_O as $\tau = (\tau_{S_1}, \tau_{S_2}, \tau_{S_4}, \tau_{S_5})$. Hence, the *core* is $\tau_c = \{(0, 0, x, 1-x); 0 \leq x \leq 1\}$, the *nucleolus* is $\tau_n = (0, 0, 1/2, 1/2)$, and the SV is $\tau_S = (1/4, 1/4, 1/4, 1/4)$.

Refer to Fig. 8.3 for the *renormalization* in Relay. The collusion between S_1 and S_2 is even harder because now 3 veto players $\{S_4, S_5, S_7\}$ are in the set $K = \{S_{1,2,4,5,7,8}\}$ to which the game $G_R = (K, \mathbf{x})$ will be defined, as shown in Tab. 8.3. Denote the payoff vector for G_R as $\zeta = (\zeta_{S_1}, \zeta_{S_2}, \zeta_{S_4}, \zeta_{S_5}, \zeta_{S_7})$. Hence, the *core* is $\zeta_c = \{(0, 0, x, y, 1-x-y); 0 \leq x, y \leq 1 \text{ and } x+y = 1\}$, the *nucleolus* is $\zeta_n = (0, 0, 1/3, 1/3, 1/3)$, and the SV is $\zeta_S = (1/5, 1/5, 1/5, 1/5, 1/5)$.

The interpretation of G_O and G_R from S_1's point of view is very similar to that in G_P. The *core* and the *nucleolus* suggest that S_1 has nothing beneficial from collusion. The SV basically says that S_1 at most has a share of value of 1/4 and 1/5 for games

G_O and G_R respectively. The SV in games in Relay also show that cheating gets lower payoff than in Proxy. Intuitively this is because in Relay the collusion channel is "longer" than that in Proxy, so it is more difficult to cheat in Relay.

It is possible that S_1 may still try to persuade his/her next agent into collusion. This problem is one of our future investigations.

8.2 Conclusion

Secure multi-party computation (secure MPC) has been established as the de facto paradigm for protecting privacy in distributed computation. One of the earliest MPC primitives is the Shamir's secret sharing (SSS) scheme. SSS has many advantages over other secure MPC primitives like garbled circuits and homomorphic encryption – it provides an information-theoretic security (ITS) guarantee, requires no complex long-integer operations, and often leads to more efficient protocols. We have provided a summary of signal processing operations based on SSS as well as an example application of image denoising in the SSS domain to demonstrate the usefulness of SSS in protecting privacy in an outsourced environment. Nonetheless, ITS-MPC protocols built from SSS receive less attention in the signal processing community because SSS requires a larger number of honest participants, making it prone to collusion attacks.

In this dissertation, we have proposed an outsourced distributed computation framework on secret data based on Shamir's secret sharing. The key innovation is a comprehensive modeling of different collusion attacks and countermeasures using game-theoretic approaches. Two types of games, user-vendor and customer-agent, have been studied. We provide a detailed analysis of possible outcomes under different privacy preferences based on the relative cost of collusion attacks over loss of privacy. User-vendor games model the intention to commit collusion attacks with the possibility of retaliation from the perspective of a customer. Using both symmetric strategic form games and Bayesian games to model uncertainty in privacy preference, we have concluded that honesty is a stable strategy if it is possible to reliably detect thefts, thereby keeping the cost of collusion high. For undetectable theft, we have proposed a deception design to inject police customers into the framework. Using the customer-agent game, we have concluded that honesty is a stable strategy for agents under a significant presence of police. A cryptographic censorship protocol has also been proposed to sanitize traffic so as to eliminate any collusion among agents under a covert adversarial model. Our collusion deterred ITS-MPC framework based on SSS is called CD-SSS. Experimental results have been provided to demonstrate the efficiency of our proposed system over state-of-the-arts Garbled Circuit systems and the validity of our game-theoretic constructions. Further investigations are needed to confirm whether the proposed mechanisms accurately reflect practical applications through the use of social computing experiments. Another important area of extension is to incorporate suitable network topology of agents into the framework. We have already conducted preliminary studies on using network of agents in reducing customers' load in the censorship protocol [133]. Further extension of game-theoretic designs to network could potentially relax conditions for honesty strategy.

Bibliography

[1] Aol removes search data on vast group of web users (http://query.nytimes.com/gst/fullpage.html?res=9504e5d81e3ff93ba3575bc0a9609c8b63). *The New York Times*, Aug. 8, 2006; accessed Feb. 2, 2016.

[2] J. Alwen, J. Katz, Y. Lindell, G. Persiano, I. Visconti, et al. Collusion-free multiparty computation in the mediated model. In *Advances in Cryptology-CRYPTO 2009*, pages 524–540. Springer, 2009.

[3] J. Alwen, J. Katz, U. Maurer, and V. Zikas. Collusion-preserving computation. In *Advances in Cryptology–CRYPTO 2012*, pages 124–143. Springer, 2012.

[4] Amazon. https://aws.amazon.com/s3/, accessed Jan. 29, 2016.

[5] P. Amsel. Pokerstars restricting chinese players to one per table. *CalvinAyre.com*, Nov. 17, 201, accessed Sept.2, 2014 at "http://calvinayre.com/2010/11/17/poker/pokerstars-restricting-chinese-players/".

[6] G. Asharov, R. Canetti, and C. Hazay. Towards a game theoretic view of secure computation. In *Advances in Cryptology–EUROCRYPT 2011*, pages 426–445. Springer, 2011.

[7] G. Asharov and C. Orlandi. Calling out cheaters: Covert security with public verifiability. In *Advances in Cryptology–ASIACRYPT 2012*, pages 681–698. Springer, 2012.

[8] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of Cryptography*, pages 137–156. Springer, 2007.

[9] V. Balachandran and S. Emmanuel. Software code obfuscation by hiding control flow information in stack. In *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*, pages 1–6. IEEE, 2011.

[10] A. Beimel. Secret-sharing schemes: a survey. In *Coding and cryptology*, pages 11–46. Springer, 2011.

[11] 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*, pages 257–266. ACM, 2008.

[12] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.

[13] E. Ben-Sasson, S. Fehr, and R. Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In *Advances in Cryptology–CRYPTO 2012*, pages 663–680. Springer, 2012.

[14] J. Benaloh. Secret sharing homomorphisms: keeping shares of a secret secret. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 251–260, London, UK, UK, 1987. Springer-Verlag.

[15] J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In *Proceedings on Advances in cryptology*, pages 27–35. Springer-Verlag New York, Inc., 1990.

[16] J. C. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Advances in Cryptology—CRYPTO'86*, pages 251–260. Springer, 1987.

[17] T. Benzel and J. Wroclawski. The deter project: Towards structural advances in experimental cybersecurity research and evaluation. *Journal of Information Processing*, 20(4):824–834, 2012.

[18] T. Bianchi, A. Piva, and M. Barni. On the implementation of the discrete fourier transform in the encrypted domain. *Information Forensics and Security, IEEE Transactions on*, 4(1):86–97, 2009.

[19] T. Bianchi, A. Piva, and M. Barni. Composite signal representation for fast and storage-efficient processing of encrypted signals. *Information Forensics and Security, IEEE Transactions on*, 5(1):180–187, 2010.

[20] D. Bogdanov, L. Kamm, S. Laur, P. Pruulmann-Vengerfeldt, R. Talviste, and J. Willemson. Privacy-preserving statistical data analysis on federated databases. In *Privacy Technologies and Policy*, pages 30–55. Springer, 2014.

[21] D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In *Computer Security-ESORICS 2008*, pages 192–206. Springer, 2008.

[22] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, et al. Secure multiparty computation goes live. In *Financial Cryptography and Data Security*, pages 325–343. Springer, 2009.

[23] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu. Private database queries using somewhat homomorphic encryption. In *Applied Cryptography and Network Security*, pages 102–118. Springer, 2013.

[24] R. Canetti and M. Vald. Universally composable security with local adversaries. In *Security and Cryptography for Networks*, pages 281–301. Springer, 2012.

[25] Carbonite. http://www.carbonite.com/, accessed Jan. 29, 2016.

[26] T. E. Carroll and D. Grosu. A game theoretic investigation of deception in network security. *Security and Communication Networks*, 4(10):1162–1172, 2011.

[27] O. Catrina and S. De Hoogh. Improved primitives for secure multiparty integer computation. In *Security and Cryptography for Networks*, pages 182–199. Springer, 2010.

[28] O. Catrina and S. De Hoogh. Improved primitives for secure multiparty integer computation. In *Security and Cryptography for Networks*, pages 182–199. Springer, 2010.

[29] S. Chang, B. Yu, and M. Vetterli. Adaptive wavelet thresholding for image denoising and compression. *Image Processing, IEEE Transactions on*, 9(9):1532–1546, 2000.

[30] T.-H. Chen and K.-H. Tsao. Threshold visual secret sharing by random grids. *Journal of Systems and Software*, 84(7):1197–1208, 2011.

[31] Y. Chen, B. Wang, W. S. Lin, Y. Wu, and K. J. R. Liu. Cooperative peer-to-peer streaming: An evolutionary game-theoretic approach. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(10):1346–1357, 2010.

[32] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, 1985.

[33] A. Chowdhry. Apple and samsung drop patent disputes against each other outside of the u.s. *Forbes*, Aug. 6, 2014, accessed on Sept. 12 at "http://www.forbes.com/sites/amitchowdhry/2014/08/06/apple-and-samsung-drop-patent-disputes-against-each-other-outside-of-the-u-s/".

[34] R. Cramer. Introduction to secure computation. In *Lectures on Data Security*, pages 16–62. Springer, 1999.

[35] I. Damgård. Theory and practice of multiparty computation. In *Security and Cryptography for Networks*, pages 360–364. Springer, 2006.

[36] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Theory of Cryptography*, pages 285–304. Springer, 2006.

[37] I. Damgård, M. Geisler, and J. B. Nielsen. From passive to covert security at low cost. In *Theory of Cryptography*, pages 128–145. Springer, 2010.

[38] I. Damgård, M. Keller, E. Larraia, C. Miles, and N. P. Smart. Implementing aes via an actively/covertly secure dishonest-majority mpc protocol. In *Security and Cryptography for Networks*, pages 241–263. Springer, 2012.

[39] I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology–CRYPTO 2012*, pages 643–662. Springer, 2012.

[40] J. L. Dautrich and C. V. Ravishankar. Security limitations of using secret sharing for data outsourcing. In *Data and Applications Security and Privacy XXVI*, pages 145–160. Springer, 2012.

[41] U. de Vigo. Signal processing in the encrypted domain.

[42] C. Devet, I. Goldberg, and N. Heninger. Optimally robust private information retrieval. In *21st USENIX Security Symposium*, 2012.

[43] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[44] C. Doukas, T. Pliakas, and I. Maglogiannis. Mobile healthcare information management utilizing cloud computing and android os. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 1037–1040. IEEE, 2010.

[45] Dropbox. https://www.dropbox.com/, accassed Jan. 29, 2016.

[46] C. Dwork. Differential privacy. *Automata, languages and programming*, pages 1–12.

[47] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography*, pages 265–284. Springer, 2006.

[48] C. Dwork and A. Smith. Differential privacy for statistics: What we know and what we want to learn. *Journal of Privacy and Confidentiality*, 1(2):2, 2010.

[49] M. Economics. Healthcare data breaches decline, but aca could be increasing risks (http://medicaleconomics.modernmedicine.com/medical-economics/content/tags/affordable-care-act/healthcare-data-breaches-decline-aca-could-be-inc?page=full), accessed Jan. 29, 2016.

[50] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18. Springer, 1985.

[51] L. Fan and L. Xiong. Real-time aggregate monitoring with differential privacy. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2169–2173. ACM, 2012.

[52] G. Fanti, M. Finiasz, and K. Ramchandran. One-way private media search on public databases: The role of signal processing. *Signal Processing Magazine, IEEE*, 30(2):53–61, 2013.

[53] G. Fuchsbauer, J. Katz, and D. Naccache. Efficient rational secret sharing in standard communication networks. In *Theory of Cryptography*, pages 419–436. Springer, 2010.

[54] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 265–273. ACM, 2008.

[55] N. Garg and D. Grosu. Deception in honeynets: A game-theoretic analysis. In *Information Assurance and Security Workshop, 2007. IAW'07. IEEE SMC*, pages 107–113. IEEE, 2007.

[56] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, pages 101–111. ACM, 1998.

[57] C. Gentry. *A fully homomorphic encryption scheme.* PhD thesis, Stanford University, 2009.

[58] A. Giurgiu, R. Guerraoui, K. Huguenin, and A.-M. Kermarrec. Computing in social networks. In *Stabilization, Safety, and Security of Distributed Systems*, pages 332–346. Springer, 2010.

[59] A. Goldberg. Can the world of online poker chase out the cheats? *BBC News*, 2010. Accessed Sept. 1, 2014 at "http://www.bbc.co.uk/news/uk-11250835".

[60] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, volume 2. Cambridge university press, 2009.

[61] 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*, pages 218–229. ACM, 1987.

[62] 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*, pages 218–229. ACM, 1987.

[63] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006.

[64] T. Granlund. The gnu mp bignum library. *Hart, W., Harvey, D., et al.: Fast Library for Number Theory, Behnel, S., Bradshaw, R., Seljebotn, D.: Cython: C extensions for Python, http://www. cython. org*, 2010.

[65] J. Halpern and V. Teague. Rational secret sharing and multiparty computation. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 623–632. ACM, 2004.

[66] J. C. Harsanyi, R. Selten, et al. A general theory of equilibrium selection in games. *MIT Press Books*, 1, 1988.

[67] C. Hazay and Y. Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer, 2010.

[68] W. Henecka, A.-R. Sadeghi, T. Schneider, I. Wehrenberg, et al. Tasty: tool for automating secure two-party computations. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 451–462. ACM, 2010.

[69] J. Hofbauer and K. Sigmund. Evolutionary game dynamics. *Bulletin of the American Mathematical Society*, 40(4):479–519, 2003.

[70] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, 2011.

[71] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer–efficiently. In *Advances in Cryptology–CRYPTO 2008*, pages 572–591. Springer, 2008.

[72] M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989.

[73] S. Izmalkov, M. Lepinski, and S. Micali. Verifiably secure devices. In *Theory of Cryptography*, pages 273–301. Springer, 2008.

[74] K. Kaya and A. A. Selçuk. Threshold cryptography based on asmuth–bloom secret sharing. *Information Sciences*, 177(19):4148–4160, 2007.

[75] J. Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 20–31. ACM, 1988.

[76] G. Kol and M. Naor. Games for exchanging information. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 423–432. ACM, 2008.

[77] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. A systematic approach to practically efficient general two-party secure function evaluation protocols and their modular design. *Journal of Computer Security*, 21(2):283–315, 2013.

[78] A. Lathey and P. K. Atrey. Image enhancement in encrypted domain over cloud. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(3):38, 2015.

[79] W. Li-Feng and W. Xiao-Bin. Research on honeypot information fusion based on game theory. In *Computer Research and Development, 2010 Second International Conference on*, pages 803–806. IEEE, 2010.

[80] Q. Lian, Z. Zhang, M. Yang, B. Y. Zhao, Y. Dai, and X. Li. An empirical study of collusion behavior in the maze p2p file-sharing system. In *Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on*, pages 56–56. IEEE, 2007.

[81] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology—CRYPTO 2000*, pages 36–54. Springer, 2000.

[82] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of cryptology*, 15(3):177–206, 2002.

[83] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi. Oblivm: A generic, customizable, and reusable secure computation architecture. In *IEEE S & P*, 2015.

[84] J. Loftus and N. P. Smart. Secure outsourced computation. In *Progress in Cryptology–AFRICACRYPT 2011*, pages 1–20. Springer, 2011.

[85] logiworks.net. Obamacare goes live: Healthcare cloud computing in action (http://www.logicworks.net/blog/2013/10/obamacare-goes-live-healthcare-cloud-computing-action/), accessed Jan. 29, 2016.

[86] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, 2012.

[87] W. Lu, A. Swaminathan, A. L. Varna, and M. Wu. Enabling search over encrypted multimedia databases. In *IS&T/SPIE Electronic Imaging*, pages 725418–725418. International Society for Optics and Photonics, 2009.

[88] A. Lysyanskaya and N. Triandopoulos. Rationality and adversarial behavior in multi-party computation. In *Advances in Cryptology-CRYPTO 2006*, pages 180–197. Springer, 2006.

[89] F. McSherry and R. Mahajan. Differentially-private network trace analysis. *ACM SIGCOMM Computer Communication Review*, 41(4):123–134, 2011.

[90] MEGA. https://mega.nz/, accessed Jan. 29, 2016.

[91] M. Mohanty, P. Atrey, and W. T. Ooi. Secure cloud-based medical data visualization. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1105–1108. ACM, 2012.

[92] M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.

[93] M. Naor and A. Wool. Access control and signatures via quorum secret sharing. *Parallel and Distributed Systems, IEEE Transactions on*, 9(9):909–922, 1998.

[94] Y. Narahari. *Game Theory and Mechanism Design*. World Scientific Publishing Company, 2014.

[95] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 111 – 125. IEEE, 2008.

[96] I. Nargis, P. Mohassel, and W. Eberly. Efficient multiparty computation for arithmetic circuits against a covert majority. In *AFRICACRYPT*, pages 260–278. Springer, 2013.

[97] K. Nayak, X. S. Wang, S. Ioannidis, U. Weinsberg, N. Taft, and E. Shi. Graphsc: Parallel secure computation made easy. In *IEEE S & P*, 2015.

[98] T. H. News. Bitcoin cloud mining service hacked; database on sale for just 1 bitcoin (http://thehackernews.com/2015/07/bitcoin-mining-server.html), accessed Jan. 29, 2016.

[99] T. Nishide and K. Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *Public Key Cryptography–PKC 2007*, pages 343–360. Springer, 2007.

[100] D. Niyato and E. Hossain. Dynamics of network selection in heterogeneous wireless networks: an evolutionary game approach. *Vehicular Technology, IEEE Transactions on*, 58(4):2008–2017, 2009.

[101] D. Niyato, E. Hossain, and Z. Han. Dynamics of multiple-seller and multiple-buyer spectrum trading in cognitive radio networks: A game-theoretic modeling approach. *Mobile Computing, IEEE Transactions on*, 8(8):1009–1022, 2009.

[102] H. Ohtsuki, C. Hauert, E. Lieberman, and M. A. Nowak. A simple rule for the evolution of cooperation on graphs and social networks. *Nature*, 441(7092):502–505, 2006.

[103] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology—EUROCRYPT'99*, pages 223–238. Springer, 1999.

[104] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology—EUROCRYPT'99*, pages 223–238. Springer, 1999.

[105] M. A. Pathak, B. Raj, S. Rane, and P. Smaragdis. Privacy-preserving speech processing: cryptographic and string-matching frameworks show promise. *Signal Processing Magazine, IEEE*, 30(2):62–74, 2013.

[106] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):12–19, 2002.

[107] A. Piva and S. Katzenbeisser. Signal processing in the encrypted domain. *EURASIP Journal on Information Security*, 2007(1):082790, 2007.

[108] D. Proserpio, S. Goldberg, and F. McSherry. A workflow for differentially-private graph synthesis. In *Proceedings of the 2012 ACM workshop on Workshop on online social networks*, pages 13–18. ACM, 2012.

[109] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85. ACM, 1989.

[110] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 735–746. ACM, 2010.

[111] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[112] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[113] K. H. Rosen. *Elementary Number Theory.* Addison Wesley, 1988.

[114] I. Roy, S. T. Setty, A. Kilzer, V. Shmatikov, and E. Witchel. Airavat: Security and privacy for mapreduce. In *NSDI*, volume 10, pages 297–312, 2010.

[115] S. M. SaghaianNejadEsfahani, Y. Luo, and S.-c. S. Cheung. Privacy protected image denoising with secret shares. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 253–256. IEEE, 2012.

[116] S. M. SaghaianNejadEsfahani, Y. Luo, and S.-c. S. Cheung. Privacy protected image denoising with secret shares. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 253–256. IEEE, 2012.

[117] S. M. SaghaianNejadEsfahani, Y. Luo, and S.-c. S. Cheung. Privacy protected image denoising with secret shares. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 253–256. IEEE, 2012.

[118] A. D. Sarwate and K. Chaudhuri. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. *Signal Processing Magazine, IEEE*, 30(5):86–94, 2013.

[119] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[120] J. Shneidman and D. C. Parkes. Rationality and self-interest in peer to peer networks. In *Peer-to-Peer Systems II*, pages 139–148. Springer, 2003.

[121] SiliconAngle. Cloud-based password manager lastpass hacked (http://siliconangle.com/blog/2015/06/16/cloud-based-password-manager-lastpass-hacked-prompts-users-to-reset-master-passwords/), accessed Jan. 29, 2016.

[122] K. Singh and H. Schulzrinne. Peer-to-peer internet telephony using sip. In *Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 63–68. ACM, 2005.

[123] E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, T. Schneider, and F. Koushanfar. Tinygarble: Highly compressed and scalable sequential garbled circuits. In *IEEE S & P*, 2015.

[124] T. Tassa. Generalized oblivious transfer by secret sharing. *Designs, Codes and Cryptography*, 58(1):11–21, 2011.

[125] D. B. H. Tay. Rationalizing the coefficients of popular biorthogonal wavelet filters. *IEEE Trans. Circuits Syst. Video Techn.*, 10(6):998–1005, 2000.

[126] M. Upmanyu, A. M. Namboodiri, K. Srinathan, and C. Jawahar. Efficient privacy preserving video surveillance. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1639–1646. IEEE, 2009.

[127] J. Vaidya. Privacy-preserving linear programming. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 2002–2007. ACM, 2009.

[128] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008.

[129] A. Vinella et al. Bayesian-nash vs dominant-strategy implementation with countervailing incentives: the two-type case. *Economics Bulletin*, 30(1):636–649, 2010.

[130] G. Wagener, A. Dulaunoy, T. Engel, et al. Self adaptive high interaction honeypots driven by game theory. In *Stabilization, Safety, and Security of Distributed Systems*, pages 741–755. Springer, 2009.

[131] J. R. Wallrabenstein and C. Clifton. Equilibrium concepts for rational multiparty computation. In *Decision and Game Theory for Security*, pages 226–245. Springer, 2013.

[132] Z. Wang and S.-C. S. Cheung. On privacy preference in collusion-deterrence games for secure multiparty computation. In *Acoustics, Speech, and Signal Processing, 2016 41st IEEE International Conference on (ICASSP)*. IEEE, 2016.

[133] Z. Wang, T. Gu, and S. ching S. Cheung. A theoretical framework for distributed secure outsourced computing using secret sharing. In *(Work-in-progress) IEEE International Workshop on Information Forensics and Security*. IEEE, 2014.

[134] Z. Wang, Y. Luo, and S.-C. Cheung. Efficient multi-party computation with collusion-deterred secret sharing. In *Acoustics, Speech, and Signal Processing, 2014 39th IEEE International Conference on (ICASSP)*. IEEE, 2014.

[135] L. Wasserman and S. Zhou. A statistical framework for differential privacy. *Journal of the American Statistical Association*, 105(489):375–389, 2010.

[136] J. N. Webb. *Game theory: decisions, interaction and Evolution*. Springerverlag London Limited, 2006.

[137] R. Wyttenbach. Gamebug software.

[138] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[139] N. Zhao and X. Zhang. The model of the invulnerability of scale-free networks based on" honeypot". In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on*, pages 1–4. IEEE, 2008.

[140] W. Zhu, C. Luo, J. Wang, and S. Li. Multimedia cloud computing. *Signal Processing Magazine, IEEE*, 28(3):59–69, 2011.

Vita

Zhaohong Wang

Educational Background

- Master of Science in Measuring Instrumentation and Automation,
  Beijing University of Posts and Telecommunications, Beijing, China, 2010.
- Bachelor of Science in Measuring and Controlling Techniques and Instrumentation,
  Tianjin University, Tianjin, China, 2007.

Professional Experience

- Research Assistant, 01/2016 - 05/2016. Center for Visualization & Virtual Environments, University of Kentucky, Lexington, KY, USA.
- Research Assistant, 01/2012 - 08/2014. Center for Visualization & Virtual Environments, University of Kentucky, Lexington, KY, USA.
- Teaching Assistant, 08/2014 - 05/2016. Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY, USA.
- Teaching Assistant, 08/2010 - 12/2011. Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY, USA.

Awards

- Graduate Travel Award, May 2014 (for IEEE ICASSP in Florence, Italy)

- Graduate Travel Award, December 2014 (for IEEE WIFS/GlobalSIP in Atlanta, USA)

- Three A's Student of Tianjin University, 2005

Publications

- Zhaohng Wang and Sen-ching S. Cheung, "Information-Theoretic Secure Multi-Party Computation with Collusion Deterrence", *IEEE Transactions on Information Forensics and Security*, under review.

- Zhaohong Wang and Sen-ching S. Cheung, "Protecting Privacy in Signal Processing", *IEEE Potentials*, Vol. 33, No. 3. May, 2014.

- Zhaohong Wang and Sen-ching S. Cheung, "On Privacy Preference in Collusion-Deterrence Games for Secure Multi-Party Computation", accepted to *The 41st IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Shanghai, China, March 2016

- Zhaohong Wang, Ting Gu, and Sen-ching S. Cheung, "A Theoretical Framework for Distributed Secure Outsourced Computing Using Secret Sharing", *The IEEE International Workshop on Information Forensics and Security (WIFS, work-in-progress session), in conjuction with The IEEE GlobalSIP*, pages 1743–1747, Atlanta, USA, December 2014.

- Zhaohong Wang, Ying Luo, and Sen-ching S. Cheung, "Efficient Multi-Party Computation with Collusion-Deterred Secret Sharing", *The 39th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 7401–7405, Florence, Italy, May 2014