EFFICIENT INCENTIVE COMPATIBLE SECURE DATA SHARING

by

Robert Nix

APPROVED BY SUPERVISORY COMMITTEE:

Dr. Murat Kantarcioglu, Chair

Dr. Bhavani Thuraisingham

Dr. Latifur Khan

Dr. Kevin Hamlen

To Mom, Dad, Phillip, Bryan, James, Matt, and FNSG.

EFFICIENT INCENTIVE COMPATIBLE SECURE DATA SHARING

by

ROBERT NIX, B.S., M.S.

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December 2012

ACKNOWLEDGMENTS

This work was supported by the Air Force Office of Scientific Research MURI-Grant FA-9550-08-1-0265 and Grant FA9550-12-1-0082, National Institutes of Health Grant 1R01LM009989, National Science Foundation (NSF) Grant Career-CNS-0845803, NSF Grants CNS-0964350, CNS-1016343, CNS-1111529, and CNS-1228198, and Army Research Office Grant 58345-CS.

I would like to thank my advisor, Dr. Kantarcioglu, for his patience and his guidance throughout my graduate school process. I would like to thank my dissertation committee for their service, and my collaborators for their invaluable support.

October 2012

PREFACE

This dissertation was produced in accordance with guidelines which permit the inclusion as part of the dissertation the text of an original paper or papers submitted for publication. The dissertation must still conform to all other requirements explained in the "Guide for the Preparation of Master's Theses and Doctoral Dissertations at The University of Texas at Dallas." It must include a comprehensive abstract, a full introduction and literature review, and a final overall conclusion. Additional material (procedural and design data as well as descriptions of equipment) must be provided in sufficient detail to allow a clear and precise judgment to be made of the importance and originality of the research reported.

It is acceptable for this dissertation to include as chapters authentic copies of papers already published, provided these meet type size, margin, and legibility requirements. In such cases, connecting texts which provide logical bridges between different manuscripts are mandatory. Where the student is not the sole author of a manuscript, the student is required to make an explicit statement in the introductory material to that manuscript describing the student's contribution to the work and acknowledging the contribution of the other author(s). The signature of the Supervising Committee which precedes all other material in the dissertation attest to the accuracy of this statement.

EFFICIENT INCENTIVE COMPATIBLE SECURE DATA SHARING

Publication No. _____

Robert Nix, Ph.D.
The University of Texas at Dallas, 2012

Supervising Professor: Dr. Murat Kantarcioglu

As time goes on, humans collect more and more data. This data is used to create useful information which allows us to make decisions. Not all data is in one database, however. Humanity's data is split among millions of different owners: companies, individuals, and governments. In order to create better information for better decisions, data sharing can be employed. However, the different motivations of the owners can cause problems in the data sharing process. This dissertation examines the different motivations involved in data sharing, and proposes methods to both expedite and assure the data sharing process's completion. In particular, it considers approximations in secure data mining, enforcing honesty in data sharing through game theory, and improvements on the verification of outsourced data queries.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

The human race produces huge amounts of data per day. A small fraction of this data is processed by computers every day. This small fraction, however, is a large amount of data. For example, Google, Inc., as of 2008, processed 20 petabytes (20000 terabytes) of data per day (Dean and Ghemawat 2008). According the the McKinsey Institute's report on big data (Manyika, Chui, Brown, Bughin, Dobbs, Roxburgh, and Byers 2011), there is approximately $900 billion in untapped value in medical and location data alone. A large chunk of raw data, however, does not have this value. In order to create this value, we have to extract useful information from the raw data, such as models, graphs, or theorems. For example, a large collection of webpages is not particularly useful to the average user, but a search result listing the most relevant pages is. Thus, the ability to efficiently and correctly extract useful information from data is paramount.

At times, companies or other entities may wish to expand their information generation potential by combining data with the data of other companies or entities. In fact, the 9/11 commission (National 2004) concluded that combining data to "connect the dots" is *necessary* to prevent acts of terror. This leads us to the concept of *data sharing*. Data sharing is the process by which multiple entities come together to create new information from their raw data. In a perfect world, this could be as simple as combining the raw data into one large database, and extracting the information. However, in the real world, entities are self-interested, and have different motivations in play.

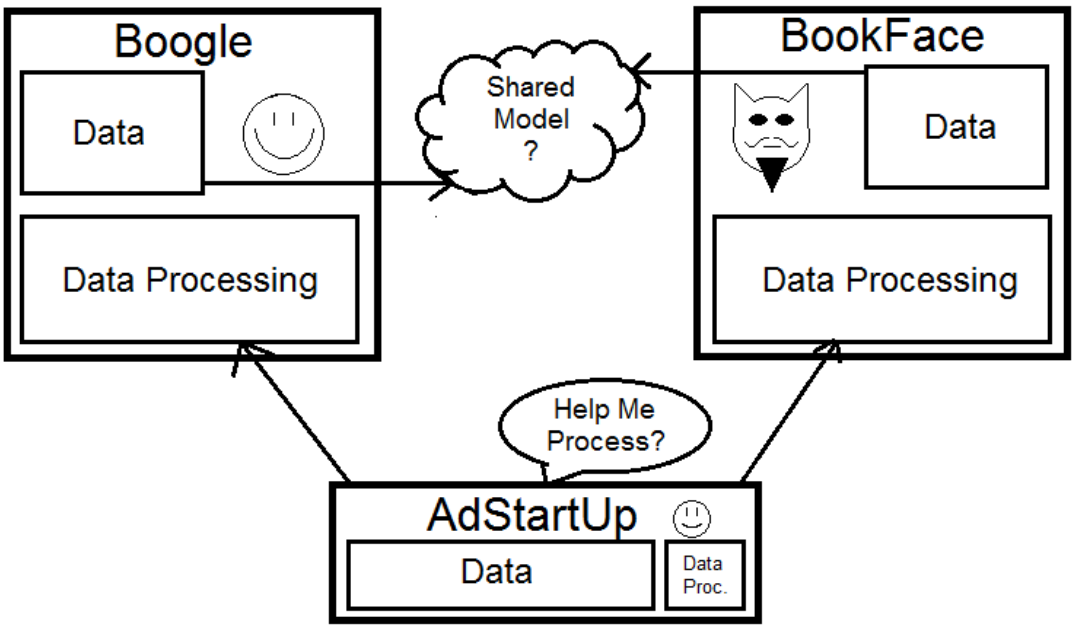


Figure 1.1. Dot Product Approximation Concept

1.1 A Motivating Example

In order to outline some of these motivations, we first discuss an example. A diagram of the forces at play in this example can be found in figure 1.1. Suppose there are two large companies, which we will call *Boogle* and *BookFace*, each of which has large amounts of data and the capability to process large amounts of data. These two companies have different types of data, however. Boogle, for the most part, has search query data, while BookFace has social networking data. It is entirely possible that, in order to better serve their customers or increase their revenue, that they might want to combine this data into a larger model. Neither company particularly wants to divulge their own data to another company, however, since it is proprietary data and is quite valuable. For this, they would want a way to compute this model while keeping their data secret.

Now, suppose this combined model could be updated unilaterally. This means that either Boogle or BookFace could remove data from the model, then apply a new set of data to get

an entirely new model. This would allow a selfishly motivated BookFace to provide fake data to the model-making process, and then, after the computation, construct the true result from the false one. Thus, we would need some method to keep the players honest, no matter what their motivations.

Finally, let's consider the case of the tiny AdStartUp company. This company has generated large amounts of data, but does not have the infrastructure necessary to process it. Boogle and BookFace, however, do have this infrastructure, and AdStartUp would like to ask one of them to help process this data. The two companies would be willing to do this for a price. Boogle and BookFace, to an outside observer, both seem to be good companies with no ulterior motives. This is because Boogle and BookFace have both spent a good deal of money advertising themselves as such. AdStartUp, then does not know whether the company it chooses will faithfully process the data, or attempt to cut corners and do as little work as possible while still getting away with it. Thus, we would need some way for AdStartUp to know whether its data is processed correctly.

In this dissertation, we develop tools to deal with all of these issues. We examine privacy-preserving data mining techniques, and find ways to improve their efficiency. We construct economic mechanisms to enforce honesty, both in the data sharing and outsourcing cases.

1.2 Motivations in Data Sharing

In our example above, Boogle and BookFace were depicted as "good" and "bad," respectively. However, for our purposes, we do not assume that there are good and bad companies (as the example diagram depicts), but merely *rational* (self-interested) companies. This is realistic, since all for-profit companies exist to make a profit. These interests, however, can be informed by several different motiviations. We now proceed with a more formal discussion of the motivations at play in data sharing.

Information Utility. The utility of the information is the intrinsic value of the information itself. It could be a measurement of accuracy, or a measure of general usefulness.

For example, a web search result for "IBM" whose first result was not International Business Machines would likely have a low information utility. Note that the information utility for one entity might not be the same as the information utility for another. This could be due to two reasons. First, if the two entities extract different information, the utility would likely be different. Otherwise, the entities might have different valuations of the same information. A medical doctor would have a greater use for diagnostic information than an advertiser. If the advertiser were a pharmaceutical representative, however, they might have great interest in diagnostic information. In our example, the intrinsic utility would be the accuracy of the model created, or the usefulness of the query result requested by AdStartUp. In short, different information is worth different amounts to different entities.

Correctness. An entity motivated by correctness wishes to receive the correct result of the data sharing process. Most of the time, the utility of correct information is higher. Thus, correctness and utility are highly linked. It is possible for an incorrect result to provide higher utility, however. For example, when creating a model from observed data, sometimes the model becomes so tuned to the previously observed data that it suffers in its ability to predict future data. This phenomenon, known as overfitting, can be combatted through altering the model or removing outliers from the data. Thus, we denote correctness as separate from the utility of the information.

Privacy. Privacy is the prevention of the disclosure of an entity's sensitive data. There are various reasons why an entity might wish to keep its data private. First, a company more than likely does not wish to disclose its data to its competition, because this would be a loss of competitive advantage. This is the case in our motivating example, as the large companies do not wish to reveal their proprietary data. Also, certain data might be sensitive, causing massive losses if leaked, such as customers' credit card information or Social Security numbers. Finally, there are laws in place that prevent some data from being disclosed, such as HIPAA (Annas 2003). Nevertheless, there are ways of extracting information from shared data without disclosing data to the other parties. These methods are collectively known as *privacy-preserving data mining*, and will be discussed shortly.

Voyeurism. Voyeurism is the motivation to violate the privacy of others. The prospect of learning proprietary data can be quite appealing to companies, and the existence of the practice of "corporate espionage" is a testament to this. While Boogle and BookFace above might wish to keep their proprietary data private, each of them would love to have a peek at the other's data. Naturally, in any privacy-preserving process, voyeurism by the different parties is minimized. Thus, voyeurism could be considered the inverse of privacy.

Obfuscation. Obfuscation is the motivation to keep other players from learning correct information from the data sharing process. One way an entity can achieve obfuscation is by providing false data as input to the process, thus rendering the result worthless. We usually assume that entities are more interested in learning the correct result than they are in obfuscating the result from others.

Exclusivity. Exclusivity, as defined by (Shoham and Tennenholtz 2005), is the combination of correctness and obfuscation. In other words, an entity motivated by exclusivity wishes to learn the correct result while ensuring that others do not receive the correct result. One way exclusivity can be achieved is by providing a false input to the process, such that the final result of the process, combined with the true data, can yield the correct result. As another example, consider the process where at the end of the data sharing process, a given entity receives the result, and is then expected to send the result to the remaining entities. This is the scenario outlined in our example above. If this entity was motivated by exclusivity, it would have incentive to send a false result to the remaining entities.

Efficiency. Efficiency is the motivation to get the data sharing process done with as few resources (time and computation) as possible. As the old adage goes, time is money. Companies and other entities are always interested in getting the result as quickly as possible. Often the data sharing process must strike a delicate balance between efficiency and privacy or efficiency and the utility of the information retrieved. It is possible that an *approximation* of the correct result might be much faster to compute.

Malice. A malicious entity is motivated purely by the suffering of other parties involved. The entity derives utility from the losses of others. A malicious party in a privacy-preserving

scenario would be motivated by voyeurism. A malicious party when correctness is desired would be motivated by obfuscation. In general, a malicious party is out to ruin the data sharing process.

1.3 Varieties of Data Sharing

These different motivations give rise to several different types of data sharing. These methods range from simple to complex, and we enumerate some of them now.

Distributed Data Mining. Distributed data mining is simply the act of using raw data to produce meaningful information in a distributed setting. If the only goal is to produce meaningful information, then the task of distributed data mining can be done in a manner not dissimilar from local data mining. However, this is often not the case, and entities may wish to keep their data private while still getting meaningful information, or they may wish to keep communication and computation cost down.

Privacy-Preserving Data Mining. Several methods have been developed for doing distributed data mining while maintaining privacy. These include anonymization techniques, perturbation techniques, and cryptographic (secure multiparty computation) techniques. Anonymization strips sensitive and identifying elements from the data before performing the data mining. Perturbation adds noise to the data before the data mining process, resulting in similar information learned, but without revealing the original data. Perturbation sacrifices utility for privacy. Cryptographic techniques use homomorphic encryption, secret sharing, virtual circuits, and other tools to exactly compute the correct result without revealing any of the data to other parties. These cryptographic protocols invariably take longer to run than the original data mining process, however. Thus, the cryptographic protocols provide a tradeoff between privacy and efficiency.

Secure Multiparty Computation. Data mining is a subset of multiparty computation, which is the general computing of a function between multiple parties. Secure multiparty computation is the process of computing such a function without revealing the inputs to the

other parties. There are general methods for performing secure multiparty computation, but these again require very expensive cryptographic protocols.

Data Outsourcing. Due to lack of infrastructure or resources, an entity may wish to outsource its data to another entity, who we will call the provider, for the purposes of processing. Thus, in order to make use of the provider's increased computing resources, the entity has the provider hold its data. Then, the owner of the data sends queries to the provider, which the provider will use its considerable resources to run. The result is then returned to the owner. In the above example, this is what AdStartUp wishes to do. This can also be done in a privacy-preserving manner.

1.4 Our Contributions

This dissertation discusses several methods in which the processes of data sharing can be improved. We first, in chapter 2, review the existing literature on secure data sharing. Then, in chapter 3, we discuss some background knowledge in game theory and cryptography that will be necessary to understand the results.

Game theory is a branch of economics which studies competitive behavior. The discipline of game theory deals in incentives. Thus, we find it natural to make use of game theory in order to deal with the various incentives involved in data sharing. In recent years, game theory has been garnering more attention in the area of computer security, as outlined in the survey paper by Katz (Katz 2008). Through the use of game-theoretic principles, we can improve the efficiency, utility, or privacy of different data sharing methods.

In chapter 4, we look at the use of approximations in privacy-preserving data mining. We propose a framework for implementing efficient privacy-preserving secure approximations of data mining tasks in the vertically partitioned setting. In particular, we implement three sketching protocols for the scalar (dot) product of two vectors which then can be used in larger data mining tasks. These approximations have high accuracy, low data leakage, and one to two orders of magnitude better efficiency. We show these accuracy and efficiency

results through extensive experimentation. We also analyze the security properties of these approximations, and propose a revised notion of secure approximations which allows for much more efficient approximations.

While we can, using clever technical means, keep sensitive data private in data mining, these means do nothing to ensure that the people involved provide correct data. Some entities may have incentive to lie. In chapter 5, we propose game-theoretic mechanisms to ensure truthful data sharing in distributed data mining. We use the Nobel Prize winning Vickrey-Clarke-Groves (VCG) mechanism to ensure truthfulness in the non-cooperative case, and the Shapley value in the cooperative case. To implement these mechanisms, we do not rely on the ability to check a given party's input data for truthfulness. Instead, we incentivize truthfulness based solely on the data mining result. This is useful when privacy is important. Under a reasonable assumption, we show that these mechanisms are incentive compatible, and, through experimentation, we show that they are applicable in practice.

In chapter 6, we extend the results from chapter 5 to present some general theoretical results regarding multiparty computation and mechanism design. In particular, we show that if a result's utility can be evaluated, then we can use this evaluation to incentivize truthful behavior using the VCG mechanism. Furthermore, we show that the intermediate results in the VCG calculation need not be evaluated by an external party. The only result which might need external evaluation is the final result.

One commonly used technology in data sharing is the use of cloud data services to do computationally intensive tasks. These cloud services, however, have their own incentives at play. In chapter 7, we examine the problem of verifying the processing of queries in data outsourcing. We use the principles of game theory and economics to vastly reduce the complexity of query verification on outsourced data. We consider two cases: First, we consider the scenario where multiple non-colluding providers exist, and then, we consider the case where only one provider exists. Using a game theoretic model, we show that given the proper mechanism and incentive structure, we can effectively deter dishonest behavior on the part of the providers with very few computational and monetary resources. We prove that

the incentive for a provider to cheat can be reduced to zero. Finally, we show that a simple verification method can achieve this reduction through experimental evaluation.

Finally, we summarize our results in chapter 8.

CHAPTER 2
RELATED WORK

In this chapter, we enumerate the various works related to the topic of this dissertation.

2.1 Prior Work in Approximate Data Mining

Privacy-preserving data mining (PPDM) is a vast field with hundreds of publications in many different areas. The two landmark papers by Agrawal and Srikant (Agrawal and Srikant 2000) and Lindell and Pinkas (Lindell and Pinkas 2000) began the charge, and soon many privacy preserving techniques emerged for computing many data mining models (Kantarcioglu and Clifton 2004; Vaidya and Clifton 2002; Clifton, Kantarcioglu, Vaidya, Lin, and Zhu 2002; Pinkas 2002). Other techniques can be found in the survey (Aggarwal and Yu 2008). For our purposes, we will focus on those works which are quite closely related to approximate data mining.

There are quite a few protocols previously proposed for the secure computation of the dot product. The protocol proposed by (Vaidya and Clifton 2002) is quadratic in the size of the vector (times a security parameter). It does, however, have some privacy concerns accoring to (Goethals, Laur, Lipmaa, and Mielikainen 2005). This same work, along with several others (Du and Atallah 2001; Ioannidis, Grama, and Attallah 2002) propose other protocols which are based on very slow public key cryptography. (Ravikumar, Cohen, and Feinberg 2004) proposes a sampling-based algorithm for secure dot product computation which relies on secure set intersection as a sub-protocol. However, the secure set intersection problem is also nontrivial. It either relies on a secure dot product protocol (Vaidya and Clifton 2002) (which would lead to a circular dependency with (Ravikumar, Cohen, and Feinberg 2004)), or a large amount of extremely expensive cryptographic operations (Vaidya and Clifton 2005b).

The sketching primitives used in this work have been applied to data mining in several different capacities. The Johnson-Lindenstrauss theorem is employed for data mining by (Liu, Kargupta, and Ryan 2006), however, they employ the Johnson-Lindenstrauss theorem as the sole means of preserving privacy, whereas we are using it as part of a process. Other works (Fradkin and Madigan 2003; Wang, Garofalakis, and Ramchandran 2007) use Johnson-Lindenstrauss projection as an approximation tool. These, however, do not make use of the projection in a privacy-preserving context, and are merely concerned with fast approximations.

The work of (Kantarcioglu, Nix, and Vaidya 2009) presents a sketching protocol for the scalar product based on Bloom filters. However, its experimentation and discussion of actual data mining tasks was insufficient. Our protocols perform better on real data mining tasks, especially at high compression ratios.

2.2 Prior Work in Incentive Compatible Classification

Cryptography and game theory have a great deal in common, in terms of the goals they try to achieve. The problems tackled by cryptography generally seek to assure that participants in certain activites are forbidden to deviate (profitably) from the prescribed protocol by rendering such actions detectable, impossible, or computationally infeasible. Similarly, mechanism design seeks to forbid deviations, but it does so by rendering the deviations unprofitable. It is understandable, therefore, that a fair amount of work has been done to use the techniques of one to solve the problems of the other. Most of this work is not directly related to ours, since a fair amount of the game theoretic security work deals with specific functions, and the individual steps of the computations of those functions.

Shoham and Tennenholtz (Shoham and Tennenholtz 2005), define the class of *NCC*, or *non-cooperatively computable* functions, and define specifically the boolean functions which are NCC. In addition, the paper defined two additional classes, p-NCC and s-NCC, which stand for probabilistic-NCC and subsidized-NCC, respectively. p-NCC are the functions

which are computable with some probability non-cooperatively, and s-NCC are the functions which are computable when external monetary motivation is allowed. This was expanded to consider different motivations (McGrew, Porter, and Shoham 2003), and coalitions (Ashlagi, Klinger, and Tenneholtz 2007). While our work does involve making functions computable in a competitive setting, it involves more complicated functions, and specifies mechanisms to ensure computability.

In addition to this, much work seeks to include a game-theoretic model in standard secure multi-party computation. Instead of considering players which are honest, semi-honest, or malicious, these works simply consider players to be rational, in the game theoretic sense. Much of this work concentrates on the problem of *secret sharing*, that is, dividing a secret number among players such that any quorum (sufficiently large subset) of them can reconstruct the secret number. This was first studied by Halpern and Teague (Halpern and Teague 2004), and later re-examined by Gordon and Katz, (Gordon and Katz 2006). Other protocols for this problem were outlined in (Abraham, Dolev, Gonen, and Halpern 2006) and (Lysyanskaya and Triandopoulos 2006). The paper by Ong, et al.(Ong, Parkes, Rosen, and Vadhan 2009), hybridizes the two areas, within the realm of secret sharing, by considering some players honest and a majority of players rational. Other work seeks a broader realm of computation, such as (Izmalkov, Micali, and Lepinski 2005), and (Kol and Naor 2008), which build their computation model on a secret sharing model. There is other work that attempts to combine game theoretic and cryptographic methodologies, many of which are surveyed in (Katz 2008). Many of these rational secure computation systems could be used to ensure privacy in our mechanism. However, like other secure computation systems, they make no guarantees about the truthfulness of the inputs.

More closely related to the work in this paper, several works have attempted to enforce honest behavior among the participants in a data sharing protocol. This paper builds on the work of Agrawal and Terzi(Agrawal and Terzi 2006), who present a model which enforces honesty in data sharing through use of auditing mechanisms. Layfield, et al., in (Layfield, Kantarcioglu, and Thuraisingham 2009), present strategies which enforce honesty in a dis-

tributed computation, without relying on a mediator. Jiang, et al., in (Jiang, Clifton, and Kantarcıoğlu 2008) integrate the auditing mechanism with secure computation, to convert existing protocols into rationally secure protocols. Dekel, et al.(Dekel, Fischer, and Procaccia 2008), create a mechanism-based framework for regression learning using risk minimization. This work says nothing about privacy, and solely focuses on regression learning. Finally, the work of Kargupta, et al.(Kargupta, Das, and Liu 2007), analyzes each step of a multi-party computation process in terms of game theory, with the focus of preventing cheating withing the process, and removing coalitions from gameplay. Each of these deals with the problem of ensuring truthfulness in data mining. However, each one requires the ability to verify the data after the calculation. Our mechanisms have no such requirement.

There is one work, by Zhang and Zhao (Zhang and Zhao 2005) which does not make use of an auditing mechanism to encourage truthfulness. However, this work does not actually encourage truthful sharing by all parties. The game theoretic strategies proposed for a non-malicious player actually encourage the player to falsify his data, although not completely, in the face of a malicious adversary. This strategy results in reduced accuracy, but greater privacy. Interestingly enough, in the strategy presented, the malicious adversary has no incentive to change his input. Our work does not consider parties to be malicious or otherwise. Our work only assumes parties are rational. In addition, Zhang and Zhao focus on data integration rather than data mining.

The Shapley value (Shapley 1952) has been applied to many things, from fair division (Moulin 1992) to power cost allocation (Tan and Lie 2002), but has not been applied in this way to data sharing.

2.3 Prior Work in Query Verification

Several works have outlined query verification methods. The vast majority of these works focus on specific types of queries. Some focus only on selection (Atallah, Cho, and Kundu 2008; Chen, Ma, Hsu, Li, and Wang 2008; Mykletun, Narasimha, and Tsudik 2006; Xie, Wang,

Yin, and Meng 2007; Yang, Papadias, Papadopoulos, and Kalnis 2009), while others focus on relational queries such as selection, projection, and joins (Pang, Zhang, and Mouratidis 2009; Pang, Jain, Ramamritham, and Tan 2005). Still others focus only on aggregation queries like sum, count, and average (Haber, Horne, Sander, and Yao 2006; Xu and Chang 2010; Yi, Li, Cormode, Hadjieleftheriou, Kollios, and Srivastava 2009). Some of these processes (Sion 2005; Yi, Li, Cormode, Hadjieleftheriou, Kollios, and Srivastava 2009) require different verification schemes for each type of query, or even each individual query, requiring that the subscriber knows which queries will be asked in advance.

Many of the aforementioned schemes require complex cryptographic protocols. Some encrypt the data itself, relying on homomorphic schemes to allow the cloud provider to perform the computation (Gennaro, Gentry, and Parno 2010; Xu and Chang 2010). A homomorphic operation will always be less efficient than the operation on the unencrypted data, rendering the overhead of these protocols greater by orders of magnitude. Others, such as (Sion 2005), rely on relatively simpler cryptographic primitives, like secure hash functions. To maintain integrity, our scheme will also use hash functions. Our verification framework is, however, simpler than these cryptography-based protocols, and can be used to improve the expected runtime of any of these verification schemes.

The work of Canetti, Riva, and Rothblum (Canetti, Riva, and Rothblum 2011) also makes use of multiple outsourcing services for query verification. However, they make use of all the services all the time, and require a logarithmic number of rounds to ensure verifiability of computation. In addition, they assume that at least one of the cloud providers is in fact honest. We, in contrast, do not assume that any provider is honest, merely that they are *rational* (meaning that the provider wishes to maximize his profits), and we only use additional providers a fraction of the time. In addition, we only require one round of computation.

CHAPTER 3
BACKGROUND

In order to keep this dissertation self-contained, we now give some necessary background information, most notably in the fields of game theory and cryptography. We will make extensive use of the game theory and some use of the cryptography.

Before proceeding with the background discussion, it is convenient to define a common notation used within the literature and within this dissertation.

Given a vector, $X = (x_1, x_2, ..., x_n)$, we define:

$$X_{-i} = (x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n)$$

Or, intuitively, X_{-i} is the vector X without the ith element.

3.1 Game Theoretic Background

Game theory is the study of competitive behavior among multiple parties. A game contains four basic elements: *players*, *actions*, *payoffs*, and *information* (Rasmusen 2007). Players have actions which they can perform at designated times in the game, and as a result of the actions in the game, players receive payoffs. The players have different pieces of information, on which the payoffs may depend, and it is the responsibility of the player to use a profitable *strategy* to increase his or her payout. A player who acts in such a way as to maximize his or her payout is termed *rational*. Games take many forms, and vary in the four attributes mentioned above, but all games deal with them. The specific games we describe in this dissertation are finite player, single round, incomplete information games, with payouts based on the final result of players' actions.

A game is said to be at *equilibrium* when no single player can unilaterally increase his or her payoff by changing his or her strategy. In such a scenario, no players have any incentive to

choose a different strategy. It was shown by Nash (Nash 1951) that all finite player games have an equilibrium, although the equilibrium might require *mixed strategies*. A mixed strategy is a strategy in which players choose each of the available actions with a certain probability. For example, consider the game with two players, A and B. During the game, the players can choose either action X or action Y, and both players choose their actions simultaneously. If both players choose the same action, player A receives a utility of 1, while player B recieves a utility of 0. Otherwise, player B recieves a utility of 1, and player A recieves a utility of 0. This game can be represented by the table in figure 3.1.

Suppose player A's strategy is to always choose action X. Player B could then choose his action to be Y, and guarantee himself a payout of 1. However, if this was the case, then player A could simply alter his strategy to choose action Y, thwarting player B's strategy. Suppose, however, that player A's strategy is to flip a fair coin, choose X if it comes up heads, and tails if it comes up Y. In this scenario, no matter what player B chooses, player B's expected payout is $\frac{1}{2}$. Player B can also choose to use this strategy. If both players use this strategy, then the game is in equlibrium, since neither player has any incentive to unilaterally change strategy. This equilibrium is the only equilibrium of the game, and since the strategies are probabilistic, the equilibrium is a *mixed strategy* equilibrium.

We can also frame the above game as a game with a pure strategy equilibrium, but with continuous actions. Instead of having the actions be X and Y, we allow each player to select, as his action, a probability between zero and one that they would select X. Let the A's chosen probability be α, and let B's chosen probability be β. As before, the equilibrium is $\alpha = \beta = \frac{1}{2}$. However, this equilibrium is in pure strategies, since the action is now to choose the probability, not the action as before. This could be considered an irrelevant distinction. However, it will prove useful in the models we present.

For behavior at an equilibrium to be considered rational, it must not only be *incentive compatible*, meaning that no player has any incentive to unilaterally deviate from that strategy, but it must also be *individually rational*. Individual rationality means that each player is expected to be no worse off than they were before they chose to participate in the game.

$\frac{A\rightarrow}{B\downarrow}$	X	Y
X	1,0	0,1
X	0,1	1,0

Figure 3.1. A simple game with a mixed strategy equilibrium

More formally, it means that the payoffs for each player in the equilibrium have an expected value greater than or equal to zero.

3.1.1 Mechanism Design for Non-Cooperative Games

Mechanism design is a sub-field of game theory, and deals with the construction of games for the purpose of achieving some goal, when players act rationally. A *mechanism* is defined, for our purposes[1], as:

Definition 1 *Given a set of n players, and a set of outcomes, A, let V_i be the set of possible valuation functions of the form $v_i(a)$ which player i could have for an outcome $a \in A$. We then define a mechanism as a function $f : V_1 \times V_2 \times ... \times V_n \rightarrow A$, which given the valuations claimed by the players, selects an outcome, and n payment functions, $p_1, p_2, ..., p_n$, where $p_i : V_1 \times V_2 \times ... \times V_n \rightarrow \Re$, that is, given the valuations claimed by the players, selects an amount for player i to pay (Nisan 2007).*

Thus, the overall payout to a player in this mechanism is his valuation on the outcome, $v_i(a)$, minus the amount he is required to pay, $p_i(v_i, v_{-i})$. A mechanism is said to be *incentive compatible* if rational players would prefer to give the true valuation rather than any false valuation. Or, more formally:

Definition 2 *If, for every player i, every $v_1 \in V_1$, $v_2 \in V_2$, ..., $v_n \in V_n$, and every $v_i' \in V_i$, where $a = f(v_i, v_{-i})$ and $a' = f(v_i', v_{-i})$, then $v_i(a) - p_i(v_i, v_{-i}) \geq v_i(a') - p_i(v_i', v_{-i})$, then the mechanism in question is incentive compatible (Nisan 2007).*

[1]Technically, this is only a *direct revelation* mechanism, but we will have no need to generalize this.

Thus, a player would prefer to reveal his true valuation rather than any other valuation, assuming all other players are truthful.

Another important term is *individual rationality*, which is intuitively whether a player would desire to participate in a game in the first place. The utility a player receives in the event that they choose not to participate is called the *reservation utility*. In order for a strategy to be considered an equilibrium, for all players, it must be individually rational and incentive compatible.

The specfic mechanism used in our data mining is the Vickrey-Clarke-Groves (VCG) mechanism. The VCG mechanism, in general, seeks to maximize the social welfare of all participants in a game. The social welfare can be defined as the sum of the valuations of all players. Thus, VCG wishes to cause rational players to act in such a way that the sum of the valuations each player has of the outcome is maximized. In mathematical notation, this is where the outcome chosen is $\text{argmax}_{a \in A} \sum_i v_i(a)$, where A is the set of possible actions, and v_i is the valuation function for player i. The VCG mechanism is defined as follows:

Definition 3 *A mechanism, consisting of payment functions $p_1, p_2, ..., p_n$ and a function f, for a game with outcome set A, is a Vickrey-Clarke-Groves mechanism if f maximizes the social welfare, by satisfying the following equation:*

$$f(v_1, v_2, ..., v_n) = argmax_{a \in A} \sum v_i(a)$$

and for some functions $h_1, h_2, ..., h_n$, where $h_i : V_{-i} \to \Re$ (h_i does not depend on v_i), for all $(v_1, v_2, ..., v_n) \in V, p_i(v_1, v_2, ..., v_n) = h(v_{-i}) - \sum_{j \neq i} v_j(f(v_1, v_2, ..., v_n))$ (Nisan 2007).

Since p_i is the amount paid by player i, this ensures that each player is paid an amount equal to the valuation of all the other players. This means that each player would have incentive to make actions to maximize the social welfare. The formal proof that the VCG mechanism is incentive compatible can be found in (Nisan 2007).

3.1.2 Cooperative Game Theory

Cooperative games, first formalized by von Neumann and Morgenstern (Von Neumann and Morgenstern 1967) use a different setup than the standard non-cooperative game scenario. Cooperative games consist of a set of players N (usually called the *grand coalition*) and a valuation function v which maps subsets of N to the amount the subset of players can gain by cooperating, with $v(\emptyset) = 0$.

A non-cooperative game can be translated into the cooperative scenario in a few ways, assuming that coalitions can enforce coordinated behavior. The most common methods are to associate with each coalition the max-min or min-max sum of the gains its members can guarantee by cooperating.

One important mechanism designed for use in cooperative games is the *Shapley value* (Shapley 1952), which is defined for each player i as:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!}(v(S \cup \{i\}) - v(S))$$

This function can also be defined as:

$$\phi_i = \frac{1}{|N|!} \sum_{R} v(P_i^R \cup \{i\}) - v(P_i^R)$$

where R is taken over the possible orderings of N, and P_i^R is defined as the elements of R which precede i in R. Informally, this value is formed by taking the contribution brought to the coalition by the player at each possible time the player could have been added to the coalition. This overall sum gives a "fair" value for the player's contribution to the grand coalition.

The Shapley value is considered individually rational, that is, players will choose to join the coalition if offered their Shapley value, if the game is *superadditive*. In a superadditive game, for any disjoint coalitions $S, T \subseteq N$, we have:

$$v(S \cup T) \geq v(S) + v(T)$$

For other games, the Shapley value is defined, but not necessarily individually rational.

3.2 Cryptographic Background

In order to maintain the integrity of our outsourced data, we will need to employ some basic cryptographic primitives. We will need to employ a scheme that allows the owner to make sure that tuples he receives from the server are legitimate, and were not added or altered by the server. We can use a simple message authentication code protocol known as HMAC (Pub 2002) (Hash-based Message Authentication Code) to do this. HMAC requires the use of *cryptographic hash functions*.

A *cryptographic hash function* or *one-way hash function* is a function mapping a large, potentially infinite, domain to a finite range. This function is simple to compute (taking polynomial time), but is difficult to invert. Equivalently, we can say that, for a cryptographic hash function f, it is difficult to find an x and y such that $x \neq y$ and $f(x) = f(y)$. Examples of cryptographic hash functions include MD5 (Rivest 1992), SHA-1, and SHA-256 (National Institute of Standards and Technology 2002).

The HMAC system creates a *keyed* hash function from an existing cryptographic hash function. Let m be the message for which we wish to create a code, and k be the key we wish to use. Let f be our cryptographic hash function, and let its required input size be n. If k has a length smaller than n, we pad k with zeroes until it has size n. If k is larger, we let k be $f(k)$ for the purposes of calculating the HMAC function. We define the HMAC function as follows:

$$HMAC(m, k) = f((k \oplus outpad)||f(k \oplus inpad)||m)$$

where outpad and inpad are two constants which are the length of f's block size (in practice, 0x5c...5c and 0x36...36, respectively).

Given a message m and its HMAC value h, if we have the key k, we can simply check to see if HMAC(m,k) matches h. If it does, then the probability that the message is not legitimate (i.e., fabricated or altered) is negligible. Someone who does not have the key k, however, will be unable to compute HMAC(m,k), and will therefore be unable to forge a correct message.

Some more sophisticated methods of verifying data exist, such as Merkle hash trees(Merkle 1979), which allow larger and smaller granularities of the message to be authenticated without authenticating the rest. These other methods of verification could be used to ensure data integrity if desired. In practice, any method of ensuring data integrity once it is in the hands of the outsourced servers will suffice. We will use the simple HMAC protocol to do this. Data integrity will be a critical component of our second solution.

It is also of interest to note the traditional models of secure multiparty computation. In multiparty computation, adversarial models include the semi-honest and malicious models. A semi-honest adversary will follow the protocol we prescribe, but will attempt to learn whatever information it can about others' private data. A malicious adversary, however, can act arbitrarily. A protocol can be considered secure against a malicious adversary if the malicious adversary learns nothing but the function result, no matter what it does. This does not, however, prevent a malicious adversary from lying about its input. Even the most sophisticated cryptographic algorithms cannot ensure truthfulness.

CHAPTER 4
APPROXIMATE PRIVACY-PRESERVING DATA MINING

Privacy is a growing concern among the world's populace. As social networking and cloud computing become more prevalent in today's world, questions arise about the safety and confidentiality of the data that people provide to such services. In some domains, such as medicine, laws such as HIPAA and the Privacy Act of 1979 step in to make certain that sensitive data remains private. This is great for ordinary consumers, but can cause problems for the holders of the data. These data holders would like to create meaningful information from the data that they have, but privacy laws prevent them from disclosing the data to others. In order to allow such collaboration between the holders of sensitive data, *privacy-preserving* data mining techniques have been developed.

In privacy-preserving data mining, useful models can be created from sensitive data without revealing the data itself. One way to do this is to perturb the data set using anonymization or noise addition (Dwork 2008) and perform the computation on that data. This approach was first pioneered by Agrawal and Srikant (Agrawal and Srikant 2000). These methods can suffer from low utility, since the data involved in the computation is not the actual data being modeled. In addition, these protocols can suffer from some security problems(Kargupta, Datta, Wang, and Sivakumar 2003; Huang, Du., and Chen 2005; Liu, Giannella, and Kargupta 2006), which can lead to the retrieval of private data from the perturbed data given.

The other way to do this is using secure multiparty computation techniques to compute the exact data mining result, on the actual data. Secure computation makes use of encryption schemes to keep the data secret, but relies on other tactics, such as encrypting the function itself, or homomorphic properties of the encryption, to perform the computation. This approach was first used by Lindell and Pinkas (Lindell and Pinkas 2000). These schemes generally rely on very slow public key encryption, which results in a massive decrease in infor-

mation output. The exact computation of data mining models can take thousands of times longer when using these public key cryptosystems.

While many functions are very difficult to compute using secure multiparty computation, some of these functions have approximations which are much easier to compute. This is especially true in those data mining tasks that deal with aggregates of the data, since these aggregates can often be easily estimated. Approximating the data mining result, however, can lead to some data leakage if the approximation is not done very carefully. The security of approximations has been analyzed by Feigenbaum, et al., (Feigenbaum, Ishai, Malkin, Nissim, Strausse, and Wright 2006), but the results of their analysis showed that to make an approximation fully private, the process of the computation must be substantially more complex. Sometimes, this complexity can make computing the approximation more difficult than computing the function itself!

Here, we present another security analysis that, while allowing some small, parameter defined data leakage, creates the opportunity to use much simpler and less computationally expensive approximations securely. We then use this model of security to show the security of three approximation methods for a sub-protocol of many vertically partitioned data mining tasks: the two-party dot product. The dot product is used in association rule mining, classification, and other types of data mining. We prove that our approximations are secure under our reasonable security definitions. These approximations can provide one to two orders of magnitude improvement in terms of efficiency, while sacrificing very little in accuracy.

4.0.1 Summary of Contributions

A summary of our contributions are as follows:

- We outline a practical security model for secure approximation which allows simple protocols to be implemented securely.
- We showcase three sketching protocols for the dot product and prove their security under our model.

- Through experimentation, we show the practicality of these protocols in vertically partitioned privacy-preserving data mining tasks. These protocols can lead to a two order of magnitude improvement in efficiency, while sacrificing very little in terms of accuracy.

Section 4.1 provides the standard definitions of secure approximations, and our minor alteration thereof. Section 4.2 outlines the approximation protocols we use. Section 4.3 gives the proof that these simple approximation protocols are secure under our definition of secure approximation. In section 4.4, we give experimental results for different data mining tasks using the approximations. Finally, we offer our overall conclusions in 4.5.

The work in this chapter is based on the works "An Efficient Approximate Protocol for Privacy-Preserving Association Rule Mining (Kantarcioglu, Nix, and Vaidya 2009)," and "Approximate Privacy-Preserving Data Mining on Vertically Partitioned Data (Nix, Kantarcioglu, and Han 2012)." The author of this dissertation was heavily involved in the publication of both, and was principal author on the second. The author of this dissertation has permission to use this material.

4.1 Secure Approximations

Much has been written about secure computation, and the steps one must go through in order to compute functions without revealing anything about the data involved. Securely computing the *approximation* of a function poses another challenge. In addition to not revealing the data through the computation process, we must also assure that the function we use to approximate the actual function must not reveal anything about the data! To this end, we outline a definition of secure approximations given by (Feigenbaum, Ishai, Malkin, Nissim, Strausse, and Wright 2006), and then propose an alteration to this framework. This alteration, while allowing a very small amount of data leakage, allows for the use of very efficient approximation protocols, which can improve the efficiency of exact secure computation by orders of magnitude.

4.1.1 A secure approximation framework

The work of Feigenbaum, et. al. (Feigenbaum, Ishai, Malkin, Nissim, Strausse, and Wright 2006) gives a well-constructed and thorough definition of secure approximations. In the paper, they first define a concept called *functional privacy*, then use this definition to define the notion of a secure approximation. First, we examine the definition of functional privacy, as follows:

Definition 1 *Functional Privacy*: Let $f(\mathbf{x})$ be a deterministic, real valued function. Let $\hat{f}(\mathbf{x})$ be a (possibly randomized) function. $\hat{f}$ is *functionally private* with respect to f if there exists a probabilistic, expected polynomial time sampling algorithm $\mathbf{S}$ such that for every input $\mathbf{x} \in X$, the distribution $\mathbf{S}(f(\mathbf{x}))$ is indistinguishable from $\hat{f}(\mathbf{x})$.

Note that the term "indistinguishable" in the definition is left intentionally vague. This could be one of the standard models of perfect indistinguishability, statistical indistinguishability, computational indistinguishability (Menezes, Van Oorschot, and Vanstone 1997), or any other kind of indistinguisability. In these cases, the adjective applied to the indistinguishablity is also applied to the functional privacy (i.e., statistical functional privacy for statistical indistinguishability).

Intuitively, this definition means that the result of $\hat{f}$ yields no more information about the input than the actual result of f would. Note, however, that this does not claim that there is any relation between the two outputs, other than the privacy requirement. This does not require that the function $\hat{f}$ be a good approximation of f. Feigenbaum, et al., therefore, also provide a definition for approximations, which is also used in the final concept of a secure approximation.

Definition 2 *P-approximation*: Let $P(f, \hat{f})$ be a predicate for determining the "closeness" of two functions. A function $\hat{f}$ is a *P-approximation* of f if $P(f, \hat{f})$ is satisfied.

Now, for this definition to be useful, we need to define a predicate P to use for the closeness calculation. The most commonly used predicate P is the $\langle \epsilon, \delta \rangle$ criterion, in which $\langle \epsilon, \delta \rangle (f, \hat{f})$ is satisfied if and only if $\forall \mathbf{x}, Pr[(1-\epsilon)f(\mathbf{x}) \leq \hat{f}(\mathbf{x}) \leq (1+\epsilon)f(\mathbf{x})] > 1 - \delta$. We do not refer

to any other criterion in our work, but the definition is provided with a generic closeness predicate for the sake of completeness.

Finally, we present the liberal definition of secure two party approximations as outlined in Feigenbaum, et al.

Definition 3 *Secure Approximation (2-parties)*: Let $f(\mathbf{x}_1, \mathbf{x}_2)$ be a deterministic function mapping the two inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ to a single output. A protocol p is a secure P-approximation protocol for f if there exists a functionally private P-approx-imation $\hat{f}$ such that the following conditions hold:

Correctness The outputs of the protocol p for each player are in fact equal to the same $\hat{f}(\mathbf{x}_1, \mathbf{x}_2)$.

Privacy There exist probabilistic polynomial-time algorithms $\mathbf{S}_1, \mathbf{S}_2$ such that

$$\{(\mathbf{S}_1(\mathbf{x}_1, f(\mathbf{x}_1, \mathbf{x}_2), \hat{f}(\mathbf{x}_1, \mathbf{x}_2)), \hat{f}(\mathbf{x}_1, \mathbf{x}_2))\}_{(\mathbf{x}_1, \mathbf{x}_2) \in X} \stackrel{c}{\equiv}$$

$$\{(\text{view}_1^p(\mathbf{x}_1, \mathbf{x}_2), \text{output}_2^p(\mathbf{x}_1, \mathbf{x}_2))\}_{(\mathbf{x}_1, \mathbf{x}_2) \in X},$$

$$\{(\hat{f}(\mathbf{x}_1, \mathbf{x}_2), \mathbf{S}_2(\mathbf{x}_1, f(\mathbf{x}_1, \mathbf{x}_2), \hat{f}(\mathbf{x}_1, \mathbf{x}_2)))\}_{(\mathbf{x}_1, \mathbf{x}_2) \in X} \stackrel{c}{\equiv}$$

$$\{(\text{output}_1^p(\mathbf{x}_1, \mathbf{x}_2), \text{view}_2^p(\mathbf{x}_1, \mathbf{x}_2))\}_{(\mathbf{x}_1, \mathbf{x}_2) \in X}$$

where $A \stackrel{c}{\equiv} B$ means that A is computationally equivalent to B. Note that in the above definition all instances of $\hat{f}(\mathbf{x}_1, \mathbf{x}_2)$ have the same value, as opposed to being some random value from the distribution of $\hat{f}$. This limits the application of the simulators to a single output. This definition essentially says that we have a functionally private function $\hat{f}$ which is a P-approximation of f which itself is computed in a private manner, such that no player learns anything else about the input data.

4.1.2 Our definition

Having defined the essential notions of functional privacy, approximations, and secure approximations, we now define another notion of functional privacy, which, while less secure than the above model, allows for vastly more efficient approximations.

Definition 4 $\langle \epsilon, \delta \rangle$-*functional privacy*: A function $\hat{f}$ is $\langle \epsilon, \delta \rangle$-*functionally private* with respect to f if there exists a polynomial time simulator $\mathbf{S}$ such that $Pr[|\mathbf{S}(f(\mathbf{x}), R) - \hat{f}(\mathbf{x})| < \epsilon] > 1 - \delta$, where R is a shared source of randomness involved in the calculation of $\hat{f}$.

Intuitively, this definition allows for a non-negligible but still small acceptable information loss of at most ϵ, while still otherwise retaining security. In practice, the amount of information revealed could be much smaller, but this puts a maximum bound on the privacy of the function. In addition, we allow the simulator access to the randomness function used in computing $\hat{f}$, which allows the simulator to more accurately produce similar results to $\hat{f}$.

The acceptable level of loss ϵ can vary greatly with the task at hand. For example, if the function is to be run on the same data set several times, the leakage from that data set would increase with each computation. Thus, for applications with higher repetition, we would want a much smaller ϵ. The ϵ can be adjusted by using a more accurate approximation.

In their work describing the original definition above, Feigenbaum, et al. (Feigenbaum, Ishai, Malkin, Nissim, Strausse, and Wright 2006) dismissed a simple, efficient approximation protocol based on their definition of functional privacy. This approximation was a simple random sampling based method for approximating the hamming distance between two vectors. The claim was that even if the computation was done entirely securely, some information about the randomness used in the computation would be leaked into the final result. Thus, we simply explicitly allow the randomness to be used by the simulator in our model. We feel this is realistic, as the randomness is common knowledge to all parties in the computation.

In short, the previous definition of (Feigenbaum, Ishai, Malkin, Nissim, Strausse, and Wright 2006) aims to eliminate data leakage from the approximation result. Our definition simply seeks to quantify it and reduce it to acceptable levels. In return, we can use much simpler approximation protocols securely. For example, the eventual secure hamming distance protocol given by (Feigenbaum, Ishai, Malkin, Nissim, Strausse, and Wright 2006) has two separate protocols (one which works for high distance and one for low distance) each of which requires several rounds of oblivious transfers between the two parties. Under our definition,

protocols can be used which use only a single round of computation and work for any type of vector, as we will show in the next section.

4.2 Scalar Product Approximation Techniques for Distributed Data Mining

Data mining is, in essence, the creation of useful models from large amounts of raw data. This is typically done through the application of machine learning based model building algorithms such as association rules mining, naive bayes classification, linear regression, or other model creation algorithms. Distributed data mining, then, is the creation of these models from data which is distributed (partitioned) across multiple owners. The dot product of two vectors has many applications in vertically partitioned data mining. Many data mining algorithms can be reduced to one or more dot products between two vectors in the vertically partitioned case. Vertical partitioning can be defined as follows:

Let X be a data set containing tuples of the form $(a_1, a_2, ..., a_k)$ where each a is an attribute of the tuple. Let S be a subset of $\{1, 2, ..., k\}$. Let X_S be the data set where the tuples contain only those attributes specified by the set S. For example, $X_{\{1,2\}}$ would contain tuples of the form (a_1, a_2). The data set X is said to be vertically partitioned across n parties if each party i has a set S_i, and the associated data X_{S_i}, and

$$\bigcup_{i=1}^{n} S_i = \{1, 2, ..., k\}$$

In previous work, it has been shown that the three algorithms we test in this paper can in fact be reduced to the dot product of two zero-one vectors in the vertically partioned case. These algorithms are association rules mining(Kantarcioglu, Nix, and Vaidya 2009), naive Bayes classification(Vaidya and Clifton 2004), and C4.5 decision tree classification(Vaidya and Clifton 2005a).

We developed three sketching protocols for the approximation of the dot product of two zero-one vectors. These protocols are used to provide smaller input to an exact dot product protocol, which is then used to estimate the overall dot product, as outlined in figure 4.1.

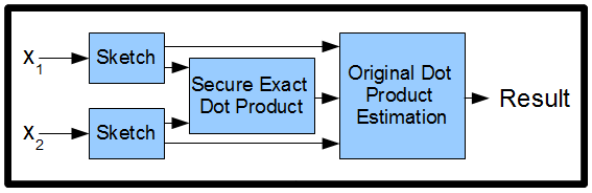


Figure 4.1. Dot Product Approximation Concept

First, we present a protocol based on Bloom filters(Bloom 1970). Second, we present a sketching protocol based on the Johnson-Lindenstrauss theorem (Johnson and Lindenstrauss 1984) and the work of (Achlioptas 2003) and (Li, Hastie, and Church 2006). Then, we present a simple sampling algorithm which is also secure under our model. Finally, we present a proof of the security of these approximations in our security model.

4.2.1 Bloom Filter Sketching

A set of items X can be represented as a bit vector of length n, where n is the total number of possible items, where the bit at position i is 1 if the item i is in set X. A Bloom filter is a lossy method of representing the set using a vector f of m bits, where $m < n$. First, we create k independent hash functions $(h_1(x), ..., h_k(x))$, each of which has a range of $\{1, ..., m\}$. Then, for each element $x \in X$, and for all $i = 1...k$, we set $f_{h_i(x)}$ to one. To test if an element x is an element of X given f, we simply test to see if $f_{h_i(x)} = 1$ for all $i = 1...k$. Now, it is possible that an element might be falsely identified as being in X, if other elements hash to $h_1(x), ..., h_k(x)$. Thus, it is important that the hash functions are chosen carefully.

Now, suppose we want to calculate the intersection of two sets X_1 and X_2 represented by Bloom filters F_1, F_2. The following formula is given in (Broder and Mitzenmacher 2004):

$$\frac{1}{m}\left(1 - \frac{1}{m}\right)^{-k|X_1 \cap X_2|} \approx \frac{Z_1 + Z_2 - Z_{12}}{Z_1 Z_2}$$

where Z_1 is the number of zeroes in F_1, Z_2 is the number of zeroes in F_2, and Z_{12} is the number of zeroes in the inner product of Z_{12}, equal to $m - (F_1 \cdot F_2)$. Solving algebraically,

we can come up with an expression for the approximation of $|X_1 \cup X_2|$:

$$|X_1 \cup X_2| \approx \frac{\ln(m(Z_1 + Z_2 + Z_{12})) - \ln(Z_1) - \ln(Z_2)}{-k\ln(1 - \frac{1}{m})}$$

Now, if the sets X_1 and X_2 were originally represented as bit vectors of length n, then this provides an approximation of the dot product of the original vectors. This algorithm is shown below as Algorithm 4.2.1.

Algorithm 4.2.1 Bloom Filter Dot Product Protocol

```
Sketch(Vector v,m,k):
sketch ← [0, ..., 0]_m
for i ← 1...|v| do
  if v_i = 1 then
    for j ← 1...k do
      sketch_{h_j(i)} ← 1
    end for
  end if
end for
return sketch
```

```
DotProductApproximation(Vector u,Vector v,m,k):
u' ← Sketch(u, m, k)
v' ← Sketch(v, m, k)
Z_1 ← countZeroes(u')
Z_2 ← countZeroes(v')
Z_12 ← m − SecureDotProduct(u', v')
return (ln(m(Z_1+Z_2+Z_12))−ln(Z_1)−ln(Z_2)) / (−k ln(1−1/m))
```

4.2.2 Johnson-Lindenstrauss (JL) Sketching

The Johnson-Lindenstrauss theorem (Johnson and Lindenstrauss 1984) states that for any set of vectors, there is a random projection of these vectors which preserves Euclidean distance within a tolerance of ϵ. More formally, for a given ϵ, there exists a function $f : \mathbb{R}^d \to \mathbb{R}^k$ such that for all u and v in a set of points,

$$(1 - \epsilon)||u - v||^2 \leq ||f(u) - f(v)||^2 \leq (1 + \epsilon)||u - v||^2$$

It is shown in that because of this property, the dot product is also preserved within a tolerance of ϵ. As with any sketching scheme, the probability of being close to the correct answer increases with the size of the sketch.

As outlined in (Achlioptas 2003) and (Li, Hastie, and Church 2006), to do our random projection, we generate a $k \times n$ matrix R, where n is the number of rows in the data set, and k is the number of rows in the resultant sketch. Each value of this matrix has the value 1, 0, or -1, with probabilities set by a sparisity factor s. The value 0 has a probability of $1 - \frac{1}{s}$, and the values 1 and -1 each have a probability of $\frac{1}{2s}$. In order to sketch a vector a of length n, we do $\frac{\sqrt{s}}{\sqrt{k}}Ra$, which will have a length of k. This preserves the dot product to within a certain tolerance. So, to estimate the dot product of two vectors a and b, we merely compute $\frac{\sqrt{s}}{\sqrt{k}}Ra \cdot \frac{\sqrt{s}}{\sqrt{k}}Rb$. Note that this will be equal to $s\frac{Ra \cdot Rb}{k}$, and in practice, we typically omit the $\frac{\sqrt{s}}{\sqrt{k}}$ term from the sketching protocol, and simply divide by the length of the sketch and multiply by the sparsity factor after performing the dot product. This yields the same result. This is shown below as Algorithm 4.2.2.

According to (Li, Hastie, and Church 2006), the sparsity factor s can be as high as $\frac{n}{logn}$ before significant error is introduced, and as s increases, the time and space requirements for the sketch decrease. Nevertheless we still used relatively low sparsity factors, to show that even in the slowest case, we still have an improvement.

4.2.3 Random Sampling

In addition to the more complicated method above, to estimate the dot product of two vectors, one could simply select a random sample of both vectors, compute the dot product, then multiply by a scaling factor to estimate the total dot product. Note that this works fairly well on vectors where the distribution of values is known, such as zero-one vectors, but can work quite poorly on arbitrary vectors. The sampling algorithm is shown below in Algorithm 4.2.3.

Algorithm 4.2.2 Johnson-Lindenstrauss(JL) Dot Product Protocol

RandomMatrixGeneration(n,k):
Matrix R
for $i \leftarrow 1...n$ **do**
  for $j \leftarrow 1...k$ **do**
    $R_{j,i} \xleftarrow{\$} \{\frac{1}{2s} : -1, 1 - \frac{1}{s} : 0, \frac{1}{2s} : 1\}$
  end for
end for
return R

DotProductApproximation(Vector u,Vector v, k):
Matrix $R \leftarrow$ RandomMatrixGeneration($|u|, k$)
$u' \leftarrow Ru$
$v' \leftarrow Rv$
return $\frac{s \cdot \text{SecureDotProduct}(u',v')}{k}$

Algorithm 4.2.3 Sampling Protocol

Sketch(Vector v, $samplingFactor \in [0...1]$):
$sketch \leftarrow []$
for $i \leftarrow 1...n$ **do**
  $r \xleftarrow{\$} [0...1]$
  if $r < samplingFactor$ **then**
    $sketch.append(v_i)$
  end if
end for
return $sketch$

DotProductApproximation(u,v,$samplingFactor$)
$u' \leftarrow$ Sketch(u)
$v' \leftarrow$ Sketch(v)
return $\frac{\text{SecureDotProduct}(u',v') \cdot |u|}{|u'|}$

4.3 Approximation Protocol Security

We now provide a proof that each of the above protocols provides a secure approximation in the sense outlined above. We first show the $\langle 2\epsilon, \delta^2 \rangle$-functional privacy of the protocols, then show that the protocols are secure under the liberal definition of secure approximations.

Theorem The protocols outlined in section 4.2 are both $\langle 2\epsilon, \delta^2 \rangle$-functionally private, and meet the liberal definition for secure approximations (definition 3).

Proof:

Proof of Functional Privacy. Let ϵ, δ be the approximation guarantees granted by the above protocols. That is, $Pr[|u \cdot v - \text{DotProductApproximation}(u, v)| > \epsilon] < 1 - \delta$. For Bloom filters, these bounds are given by the estimation formula found in (Broder and Mitzenmacher 2004). For JL, these bounds are provided by the Johnson-Lindenstrauss theorem itself, as shown by the work of (Liu, Kargupta, and Ryan 2006). For sampling, we can use the Hoeffding inequality (Hoeffding 1965) to establish a bound on the error:

$$Pr[|\hat{f}(\mathbf{x}) - f(\mathbf{x})| \geq \epsilon] \leq 2e^{-2\epsilon^2 n^2}$$

Where n is the sample size. As $\hat{f}$ can be taken to be an estimate of the mean of the product of the random variables, the Hoeffding inequality holds for the dot product of the samples. So, we set our δ to $2e^{-2\epsilon^2 n^2}$.

Note that, with both of these approximation protocols, adjusting the size (for JL, the matrix size, and for sampling, the sample size), allows us to adjust the ϵ of the functional privacy requirement. This would allow us to adjust the ϵ value to be as low as we deemed necessary for our purposes.

Now, let our simulator $\mathbf{S}(f(\mathbf{x}), R)$ generate two random zero-one vectors u and v such that $f(u, v) = u \cdot v = f(\mathbf{x})$. We then apply the randomness given to perform a calculation of the dot product approximation $\beta = \hat{f}(u, v)$. Now, the probability that $|f(\mathbf{x}) - \hat{f}(\mathbf{x})| \geq \epsilon$ is $1 - \delta$. The probability that $|f(\mathbf{x}) - \beta| \geq \epsilon$ is also $1 - \delta$, since $f(\mathbf{x}) = f(u, v)$. As these are independent events, the probability that neither occurs is δ^2. In the case this occurs,

we have $|f(\mathbf{x}) - \hat{f}(\mathbf{x})| \leq \epsilon$ and $|f(\mathbf{x}) - \beta| \leq \epsilon$, which means that $-\epsilon \leq f(\mathbf{x}) - \hat{f}(\mathbf{x}) \leq \epsilon$ and $-\epsilon \leq f(\mathbf{x}) - \beta \leq \epsilon$. Because of this, the difference between the two quantities $(f(\mathbf{x}) - \hat{f}(\mathbf{x})) - (f(\mathbf{x}) - \beta) = \beta - \hat{f}(\mathbf{x})$ can be no more than 2ϵ. If our simulator returns β, then we have shown that $\hat{f}$ is $\langle 2\epsilon, \delta^2 \rangle$-functionally private with respect to f.

Proof of Secure Approximation (under Definition 3). For the approximation to be considered secure, it must compute the same value for both players (which is trivially true for all protocols), and be private with respect to the views of each player. Now, consider, in each case, what each player sees. Player 1 sees his input, a sketch of that input, and the inputs and outputs of a secure dot product protocol. Our simulator can take that input, sketch it, and simulate the secure dot product protocol, altering its output to be $\hat{f}(\mathbf{x})$ to player 1. Since this output is all player 1 sees outside of the secure dot product protocol, it cannot distinguish this from the true output. Player 2 sees the same thing, his input, a sketch of that input, and the operations of a secure dot product protocol on the inputs. Since the subprotocol is secure, neither player can learn anything about the inputs that the sketches would not tell them.

Having shown that the sketching protocols are $\langle \epsilon, \delta \rangle$-functionally private, and that the computation protocol is secure under definition 3, we now claim that the entire protocols are secure under our model. ∎

4.4 Experiments

In order to determine the efficiency and effectiveness of the algorithms proposed, we conducted several experiments. Each of the sketching protocols presented were inserted into the data mining process for three different data mining tasks: association rules mining, naive Bayes classification, and C4.5 decision tree classification. For Bloom filters, since previous experimental results (Kantarcioglu, Nix, and Vaidya 2009) showed that an increase in the number of hash functions (k) in the Bloom filter protocol did not result in an increase in accuracy, we chose to keep $k = 1$. We used three separate sparsity values for JL sketching:

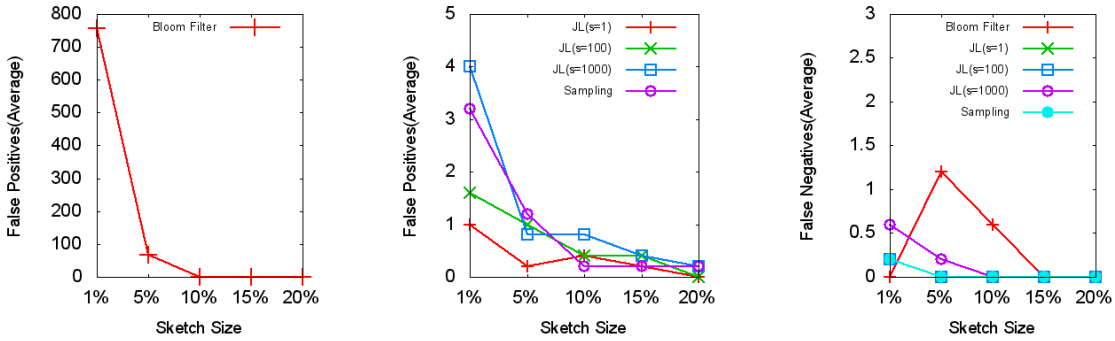
$s = 1$, which results in a matrix completely full of 1 and -1, $s = 100$, and $s = 1000$. The efficiency of JL increases with s, and these values are much lower than what is required to achieve good accuracy (Li, Hastie, and Church 2006).

For association rules mining, we used the retail data set found at (Goethals 2005), which lists transactions from an anonymous Belgian retail store. We considered three variables in the association rules experiments: the required support, the required confidence, and the compaction ratio of the sketching protocol. For testing the required support, we used 2%, 3%, 4%, 5%, and 6%, while holding the confidence constant at 70% and the compaction ratio constant at 10%. For the confidence, we used 60%, 65%, 70%, 75%, and 80% while holding the support constant at 4% and the compaction ratio constant at 10%. Finally, for the compaction ratio, we used 1%, 5%, 10%, 15%, and 20%, holding the support constant at 4% and the confidence constant at 70%. For naive Bayes and C4.5 decision tree classification, we used the Adult data set from the UC Irvine Machine Learning Repository (Asuncion and Newman 2007), which consists of data from the 1993 US Census. As there were no paramaters to set for naive Bayes or the decision tree, we varied only the compaction ratio as above. We did, however, discretize each attribute of the data set before performing the data mining, as continuous data would not function under our model. For each task and variable set, we ran ten separate experiments, using different initialization values for the inherent randomness in the sketching protocols. We employed ten-fold cross-validation for the classification tasks. The accuracy results were then averaged over all ten trials to come up with the final result.

4.4.1 Accuracy

Association Rules Mining

To assess the accuracy of the algorithms on association rules mining, we look at both the number of false positives (that is, the number of invalid associations returned by the algorithm) and false negatives (the number of valid assocations not returned by the algorithm). For the association rules mining, this is a better picture of the accuracy than overall accu-

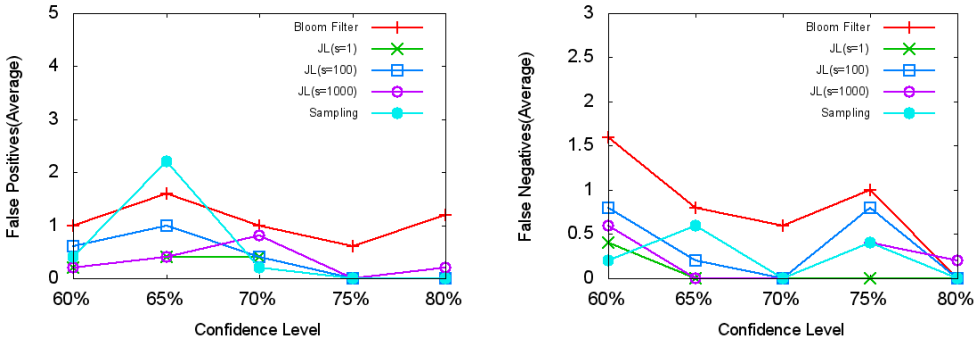


False Positives(Bloom) False Positives(JL,Sampling) False Negatives

Figure 4.2. Association Mining Results Varying Sketch Size

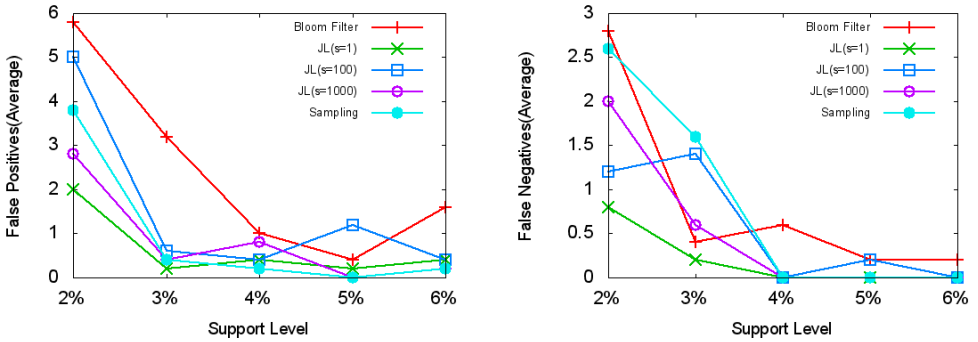
racy, since the true positives are so much rarer than the true negatives. Figure 4.2 shows the results when we varied the compaction ratio. Note that at very low compaction, the Bloom filter method almost always fills the filter, resulting in a dot product estimation of infinity. Now, since this makes no sense, we set any dot product estimation larger than the original vector to be equal to the size of the original vector. This, in turn, results in an incredibly high false positive rate at low compaction ratios for Bloom filters. JL and sampling are very similar in terms of accuracy, with a slight overall edge to JL. Note that by the time we reach a compression ratio of 10%, no more false negatives arise in any JL sketching (regardless of sparsity), or in the sampling protocol.

Figures 4.3 and 4.4 show the results varying the required confidence and required support, respectively. As one might expect, there is no discernable correlation between these variables and the accuracy of the approximation for it. A larger error rate generally indicates that there are more itemsets near the exact required value, which means a smaller error in the dot product might result in the incorrect rejection or acceptance of an itemset. This is especially true for a support value of 2%, since below 2%, the number of supported itemsets increases dramatically. In any case, JL and sampling again had similar performance, while Bloom filters had more errors most of the time.



False Positives False Negatives

Figure 4.3. Association Mining Results Varying Confidence



False Positives False Negatives

Figure 4.4. Association Mining Results Varying Support

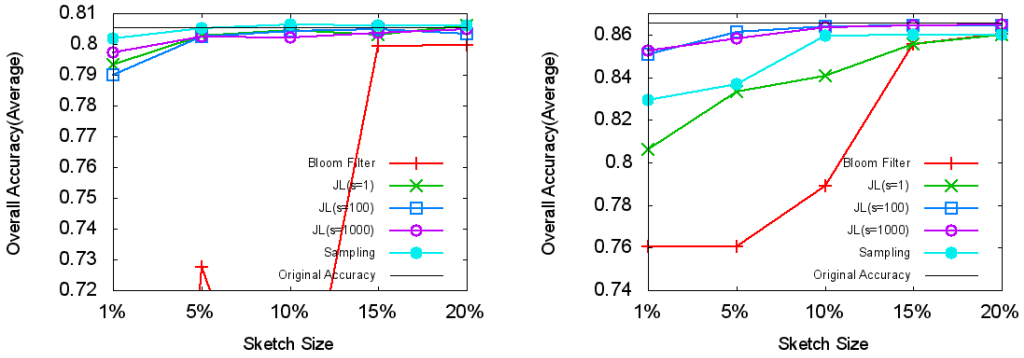
Naive Bayes Classification

Figure 4.5 (left side) shows the results for naive Bayes classification. Two of the data points for Bloom filters are not shown on the graph. The 1% filter resulted in an overall accuracy of 23%, since, as stated before, a very high compaction ratio results in filling the filter every time. Thus, each partial probability is computed to be $\frac{dataLength}{classSize}$, which, by the formula above, results in the algorithm choosing the *least* likely class. The 10% value is roughly 58%. The 5% value is higher because at 5% we only fill the filter some of the time, which leads to the algorithm simply guessing the most likely class each time. However, once we reach a 15% compaction ratio, the Bloom filter method begins to perform well.

JL and sampling, again, perform quite similiarly. The accuracy, as expected, increases with the sketch size. The thin black line on the graph represents the accuracy of the naive Bayes classification on the original, uncompacted data. The accuracy of the approximation for both JL and sampling hovers right around the original accuracy, and in some cases performs better. This is understandable due to the machine learning phenomenon of *overfitting*. When a model is built on some data, it performs quite well on the data it was trained with, but the model will not perform as well on test data. When this happens, the model is said to overfit the training data. Often some noise is added to the model to remove the overfitting problem. The approximation of the dot product can provide such noise. Thus, the approximations can achieve higher accuracy than the exact result.

C4.5 Decision Tree

Figure 4.5 (right side) shows the results for C4.5 decision tree classification. The results are consistent with our findings in other tasks. The Bloom filters provide lower accuracy, while JL and sampling are very similar. Interestingly enough, the more sparse versions of JL outperformed the unabridged ($s = 1$) version. This is likely due to the fact that the sparse vectors provided slightly less distortion in the multiplication, resulting in a closer approximation for the dot product. In this case, as opposed to the naive Bayes case, the original tree



Naive Bayes C4.5 Decision Tree

Figure 4.5. Naive Bayes and C4.5 Results Varying Sketch Size

provides a higher degree of accuracy, mainly because the C4.5 algorithm implements noise introduction by pruning the tree after building it.

4.4.2 Efficiency

In order to gauge the efficiency of our sketching protocols, we ran several timing experiments. The machine used was an AMD Athlon(tm) 64X2 dual core processor 4800T at 2.5 GHz with 2GB of RAM, running Windows Vista, and running on the Java 6 Standard runtime environment update 24. As our sub-protocol for exact dot product computation, we use the protocol of Goethals, et al (Goethals, Laur, Lipmaa, and Mielikainen 2005), as it is provably secure, and lends itself well to improvement from our sketching protocols.

First, we ran several timing experiments computing the complete dot product of zero-one vectors of size 1000. The average time for the computation was 105 seconds. To ensure that the algorithm scaled linearly, we then ran it on vectors of size 2000, and the average computation time was 211 seconds. So, we determined the time-per-element in the dot product protocol to be .105 seconds. From this point forward, we computed the runtime of the approximate protocol in terms of the run time of the exact protocol by counting the time not involved in the computation of dot products, then adding it to the estimated dot product

Mining Task	Sketching Protocol	Compaction Ratio 1%	5%	10%	15%	20%
Assoc. Mining	Bloom Filter	1.1043%	5.0671%	10.0794%	15.0824%	20.0854%
	JL(s=1)	1.2330%	5.9753%	12.0635%	17.9322%	23.9004%
	JL(s=100)	1.1019%	5.5068%	11.0154%	16.6250%	22.1959%
	JL(s=1000)	1.0968%	5.4391%	10.9785%	16.4922%	22.0116%
	Sampling	1.0798%	5.0972%	10.0880%	15.0792%	20.0882%
Naive Bayes	Bloom Filter	1.0394%	5.03990%	10.0406%	15.0408%	20.0411%
	JL(s=1)	1.1039%	5.5099%	11.0198%	16.5268%	22.0332%
	JL(s=100)	1.0902%	5.3681%	10.8624%	16.2493%	21.2520%
	JL(s=1000)	1.0888%	5.3322%	10.7194%	15.9864%	21.0516%
	Sampling	1.0132%	5.0134%	10.0139%	15.0144%	20.0147%
C4.5 Tree	Bloom Filter	0.0014%	0.0062%	0.7244%	2.3581%	2.9318%
	JL(s=1)	0.2036%	0.2284%	0.6555%	1.8956%	2.7209%
	JL(s=100)	0.1893%	0.1945%	0.5923%	1.7182%	2.6438%
	JL(s=1000)	0.1759%	0.1962%	0.5841%	1.6502%	2.6122%
	Sampling	0.0220%	0.1915%	0.8003%	1.4445%	2.5389%

Figure 4.6. Efficiency Results: Percent of the Exact Algorithm Runtime

calculation time based on the previous timing experiments. The actual formula used was:

$$\frac{t_i + .105s \cdot n_d \cdot compactionRatio \cdot n}{.105s \cdot n_d \cdot n}$$

Where t_i is the time involved in the sketching, n_d is the number of dot products performed, n is the length of the vectors involved, and *compactionRatio* is the fraction of the original vector's size which is retained by the sketching protocol. The results for three different sketching algorithms and five different compaction ratios are be are seen in figure 4.6.

In all cases, the algorithms are much faster than the exact algorithm. Because it produces a matrix with 1 or -1 for every value, JL with $s = 1$ has a large amount of pre-processing before it can apply the projection to each vector, which again, takes time. This runtime can be improved by using the sparsity factor. We chose, however, to present the worst case, as it is still much better than the original runtime. The association rules mining process involved the fewest number of dot products computed. Therefore, the preprocessing and other portions of the algorithms took up a greater percentage of the time in association rules mining. The

Naive Bayes process had orders of magnitude more dot product calculations, so the overall time was dominated by the number of dot product calculations necessary.

In the decision tree case, the number of dot products computed varied with the algorithm involved. This is because we use the dot products to determine if a node is to be split. If a split is found to be not useful, the split will not occur. The compaction introduced enough error into the calculation that splits with very little information gain were not even attempted, resulting in much fewer dot products being calculated. The different algorithms all calculated far fewer dot products at every compaction level, resulting in a much greater efficiency increase.

4.5 Conclusions

We have presented several interesting approximation techniques for the secure compuation of the dot product of two vectors. These protocols can be applied to many different data mining tasks, and can provide an efficiency increase to any protocol that uses a secure dot product as a sub-protocol. Depending on the protocol invoked, and the parameters used, these protocols can increase efficiency by up to two orders of magnitude without sacrificing much in the way of accuracy.

CHAPTER 5
INCENTIVE COMPATIBLE PRIVACY-PRESERVING DISTRIBUTED CLASSIFICATION

Information has become a power currency in our society. As such, people treat information with care and secrecy. There are times, however, that information needs to be shared among owners for the betterment of society, or simply for their own profit. Data mining seeks to take information and aggregate it into models that are more useful than the original information. Since people are cautious and do not wish to give up their private information the need for *privacy-preserving* data mining has arisen. In addition to the simple desire for privacy, certain government regulations, such as the Health Insurance Portability and Accountability Act (HIPAA) (Annas 2003) require that certain data be kept private.

Techniques for privacy preserving data mining are many in number. Some of these include anonymizing data (Sweeney 2002; Machanavajjhala, Kifer, Gehrke, and Venkitasubramaniam 2007; Xiao and Tao 2007), noise addition techniques (Islam and Brankovic 2003; Dwork, Kenthapadi, McSherry, Mironov, and Naor 2006), and cryptographic techniques (Pinkas 2002; Clifton, Kantarcioglu, Vaidya, Lin, and Zhu 2002), in addition to countless others. The cryptographic techniques have the distinction of being able to compute models based on unperturbed data, since the cryptography ensures that the data will not be revealed. However, they make no guarantees that participants will not use false data anyway.

Consider the following scenario: Suppose that the different intelligence agencies around the world wish to share their information on terrorist networks, in order to increase global knowledge about terrorists and terrorist organizations. This, of course, is a noble goal, and would benefit mankind as a whole. Intelligence agencies, however, wish to receive credit for capturing terrorists, and to this end, may provide false information in hopes of having the best information to themselves. However, several agencies could have this plan. Even if the

agencies compute the overall terrorist information model securely and privately, this would not change the fact that the end result would not be an accurate model based on real data. Because of this, the intelligence agencies get no closer to finding terrorists, potentially causing danger to ordinary citizens.

Granted, this is a rather extreme example, but it illustrates the failure of traditional cryptographic secure multi-party computation to ensure that players use truthful data. The discipline of cryptography can be used to create provably secure protocols which guarantee the privacy of the data of all parties in data mining. What then does this say about the correctness of the result of the calculation? It is true that in many situations, it can be proved that the calculation will be correct with respect to the data supplied by the players for the calculation. This is usually based on commitments that must be made by each player, ensuring that no player can change their input at any time during the calculation. However, this does not ensure that the player would provide true data for the calculation! In particular, if the data mining function is *reversible*, that is, given two inputs, x and x', and the result $f(x)$, it is simple to calculate $f(x')$, then a player might wish to provide false data in order to exclusively learn the correct data mining result! (Shoham and Tennenholtz 2005) One simple example of a reversible data mining function in practice is the Naive Bayes classifier in the vertically partitioned case, which takes the form

$$p(C) \cdot \prod_{i=1..n} p(F_i|C)$$

where $p(C)$ is the probability of a given class, and $p(F_i|C)$ is the probability of an attribute F_i given that the instance is a member of that class. If a player j wished to cheat, and provided $p'(F_j|C)$ instead, the calculation would become

$$p(C) \cdot p'(F_j|C) \cdot \prod_{i=1..n|i\neq j} p(F_i|C)$$

To retrieve the correct result, player j can multiply the above by $\frac{p(F_j|C)}{p'(F_j|C)}$, yielding the original formula. This is merely an example of the many useful data mining functions which are reversible.

In order to combat this problem, scholars have attempted to mesh game theory with cryptography to deal with the actions of players who act in their own self interest. Given that one can verify, after the fact, albeit with some cost, that a player used their true data, it is quite simple to ensure that players use true data. We simply audit the process with a high enough frequency, and stiff enough penalty, that players will think twice about lying about their data. The classic IRS game (Rasmusen 2007) is a typical example of this: a taxpayer can be motivated to be truthful on his return by both the magnitude of the penalty for cheating, and the frequency of audits. The higher the penalty, the less frequent audits need to be. However, in most cases, the ability to audit the data defeats the purpose of privacy-preserving data mining, in that it requires a trusted auditor to be able to access each player's data. The main question we address in this paper is: What guarantees can we make about the truthfulness of players' data when we have no way of verifying the data used by a given player?

We tackle this problem by using a monetary mechanism to encourage players to be truthful about their data without being able to verify the truthfulness of the data that players provide. It is important to be able to do this without verifying data, because the verification of the data could violate privacy! To illustrate the effectiveness of an after-the-fact mechanism, consider the following scenario: Several passengers are flying on a chartered cross-country flight, and the flight passes on fuel costs to the passengers. In order to board, the charter airline requires all passengers to report their weight, so that the airline can calculate the necessary fuel to get to their destination. In this case, passengers have the incentive to tell the turth about their weights, since if they under-report, the plane could crash from lack of fuel, and no amount of money (or embarrassment) saved is worth their lives. In addition, if they over-report, they are simply increasing their cost. Therefore, there is no reason to verify each passenger's weight by means of a scale, since each passenger will give their correct weight (unless, of course, they do not *know* their weight).

In a similar vein, our data mining mechanism does not require the verification of the data, it simply encourages truthfulness through extrinsic incentives. Namely, it provides monetary

incentives which subsidize the calculation, and these, in turn, motivate truthful behavior. We invoke a Vickrey-Clarke-Groves (VCG) mechanism based on the accuracy of the result itself in order to encourage correct data reporting. We show that, to the risk-averse player, the mechanism will encourage true data sharing, and for the risk-neutral player, the mechanism gives a close approximation that encourages minimal deviation from the correct data. In addition, we provide another mechanism based on the Shapley value which encourages truthful sharing in the cooperative setting. This is important since the non-cooperative setting only considers individuals and the lies that a single player can make. The cooperative solution considers what happens when players can collude in order to cheat the system, and creates incentives for entire groups of players to truthfully reveal their data.

For the purposes of this work, we focus on classification tasks. We choose to focus on classification tasks for three reasons. First, classification tasks have a widely accepted measure of utility: classification accuracy. This allows us to build our mechanisms on the common utility metric. Second, classification tasks are common in practice, used in association rules mining, recommender systems, and countless other applications. Finally, we focus on classification tasks because we feel the results generalize well to any task with a well-formed accuracy and utility metric.

Our contributions can be summarized as follows:

- We develop two mechanisms to encourage truthful data sharing which does not require the ability to audit or verify the data, one for the non-cooperative case, and one for the cooperative case.
- We prove that these mechanisms are incentive compatible under reasonable assumptions.
- We provide extensive experimental data which shows the viability of the mechanisms in practice.

In the next section, section 5.1, we describe the game theoretic model we use to represent the data mining process, the *assured information sharing game*. In section 5.2, we outline

our mechanisms, and prove their incentive compatibility. In section 5.3, we show experimental data on different kinds of data mining problems, indicating the practical use of this mechanism. Finally, in section 5.4, we give our conclusions.

The contents of this chapter are based on "Incentive Compatible Distributed Data Mining (Kantarcioglu and Nix 2010)," and "Incentive Compatible Privacy-Preserving Distributed Classification (Nix and Kantarcioglu 2012b)," the latter of which was published in the IEEE Transactions on Dependable and Secure Computing. The entire contents of the paper appear within this dissertation. IEEE does not require permission to include entire articles in dissertations. The author of this dissertation was principal author on both publications, and the co-authors have given consent for this material to appear in the dissertation.

5.1 Our Model: The Assured Information Sharing Game

In order to analyze data mining tasks in terms of game theory, we now describe a game scenario outlining the process for some data mining task. This model is a simple model in which a mediator does the data mining calculations. This may not be necessary, but for now, we use this to simplify our calculations. In terms of doing the calculation, the mediator can be removed using the **cryptographically secure** techniques outlined in (Kol and Naor 2008) or (Izmalkov, Micali, and Lepinski 2005), however, it may or may not be possible to remove the mediator for payments. We examine this further in chapter 6. We also consider only individual actions, rather than coalitions, for simplicity.

Definition 4 *Mediated Information Sharing Game*

Players: $P_1, P_2, ..., P_n$, *and a mediator* P_t.

Preconditions: Each player $P_i \in \{P_1, ..., P_n\}$ *has* x_i, *a piece of data which is to be shared for the purposes of computing some function of the data.* P_t *is another party who is bound to compute a data mining model from the players' data in a secure fashion.* P_t *is also in possession of a* ***small independent test data set****. It is reasonable that* P_t *could have such*

a set through observation of a small amount of public data, though this amount of data may not be enough to build an accurate model.

Game Progression:

1. Each player $P_i \in \{P_1, ..., P_n\}$, selects x'_i, which may or may not equal be equal to x_i, or chooses not to participate. These inputs are committed. Define X to be the vector of original values x_i, and X' to be the vector of chosen values x'_i.

2. Players send X' to P_t for secure computation of the data mining function. The function which builds the model will be referred to as D.

3. All players receive the function result, $m = D(X')$.

Payoffs: For each $P_1...P_n$, define the utility of a participating player as the following:

$$u_i(x_i, D(X')) = max\{v_i(m) - v_i(D(x_i)), 0\} - p_i(X', m) - c(D)$$

$v_i(m)$ is the intrinsic utility of the function result, which we approximate as the accuracy of a data mining function. Thus, $v_i(m) = acc(m)$ where acc is some accuracy metric applied to the data mining model. This will, of course, vary based on the truthfulness of each player. We normalize each player's reservation utility, that is, the utility received if the player chooses not to participate, to zero. This can be done without loss of generality by subtracting the reservation utility (which is $v_i(D(x_i))$, based on the accuracy of the model based only on one's own data), from the valuations in the mechanism. Note that a player will always recieve at least this much utility, so we obtain the expression $max\{v_i(m) - v_i(D(x_i)), 0\}$. $p_i(X', m)$ is the amount paid by P_i, based on the inputs and the results. Note that if p_i were to be negative, it would mean that P_i receives money instead. $c(D)$ is the computational cost of computing D. Since D is securely computed, there will be some cryptography involved in the computation of the model, hence computational cost should be considered.

5.1.1 The Cooperative Sharing Game

Using a very simple method, we can define the assured information sharing game in the context of a cooperative game. The players are already defined. The valuation function

$v_c(S)$ where S is a subset of the grand coalition of players (N), can be defined as the sum of the maximum valuations attainable by each player through collaboration among S. More formally:

$$v_c(S) = \sum_{i \in S} \max_{x_S} \min_{x_{-S}} v_i(x_S, x_{-S})$$

This maximum value that the coalition S can guarantee is called the max-min value, and the formulation of the v_c function is commonly called the α-effective form of the non-cooperative game (Aumann 1961). The β-effective form uses the min-max value, that is, the worst-case value for the maximum value which can be achieved by collaboration among S. Since each coalition's goal is to maximize their own payout, without regard for the payouts of others, players do not need to consider the worst case maximum, but the best case given any play by the other players. Therefore, we choose the α-effective form of the game.

Since we normalize the reservation utility for each player to zero, both the empty coalition $\emptyset$ and singleton coalitions $\{i\}$, for $i \in N$, have a valuation of 0. A two player coalition will have a valuation equal to twice the accuracy of the classifier created by both players' data, minus the accuracy of the classifiers of both players individually, as for players i and j where $i \neq j$, the gain experienced by i is $acc(D(x_i, x_j)) - acc(D(x_i))$ and the gain experienced by j is $acc(D(x_i, x_j)) - acc(D(x_j))$. In general,

$$v_c(S) = |S| acc(D(X_S)) - \sum_{i \in S} acc(D(x_i))$$

Our assumption, as before, is that the true data provides the best data mining model, for each subset of players. Therefore, this expression, over expectation, will be maximized when all members of S share truthful data. Any player who joins the coalition is best served by using truthful data. We assume players not joining the grand coalition will be attempting to disrupt the coalition in whatever way possible.

5.2 Our Solution

To motivate players to truthfully reveal the information, we propose the following:

1. In addition to computing the data mining model, P_t also computes $D(X'_{-i})$ for each P_i, that is, the data mining function without using the data provided by player i.

2. For each P_i, we let $p_i(X', m) = \sum_{j \neq i} v_j(D(X'_{-i})) - \sum_{j \neq i} v_j(m) - c(D)$, where v_i is determined by measuring the accuracy of the data mining model on the independent test set which P_t has. This pays each player an amount equal to the difference in accuracy between the overall data mining model and the data mining model without his input, essentially rewarding each player based on their own contribution to the model. We include the $-c(D)$ term in order to balance out the cost of the calculation. Figure 5.1 shows the process used to calculate the payment for a given player i.

Theorem 5.1 The above mechanism motivates players to truthfully reveal their inputs, under the following assumption:

Assumption: For each player i, the probability of an increase in the classifier's accuracy decreases significantly with the distance between the player's actual data and the data the player provides to the classifier building process. More formally, we may state that the expected value of the classifier's accuracy does not increase with said distance. Mathematically, for $X = x_i \cup X_{-i}$ and $X' = x'_i \cup X_{-i}$, this can be written as

$$E[acc(D(X))] \geq E[acc(D(x'))] + f(dist(X, X'))$$

where f is a non-negative, increasing function for all i, x_i, x'_i and X_{-i}.

This is essentially the implicit assumption used by any data miner: that deviating from the true data makes a bad classifier more likely. We feel that this assumption is, while not always true, always reasonable. Raw data mining processes, in practice, use true data unless they are trying to combat the problem known as "overfitting". Overfitting is when the data model is too well tuned to training data, and this causes accuracy on practical data to fall. In such instances, outliers are removed, or irrelevant dimensions are reduced away, but the data otherwise remains true. Usually, if the data is to be doctored in any way, it would be done before the data mining process would even take place. Another way to think of this

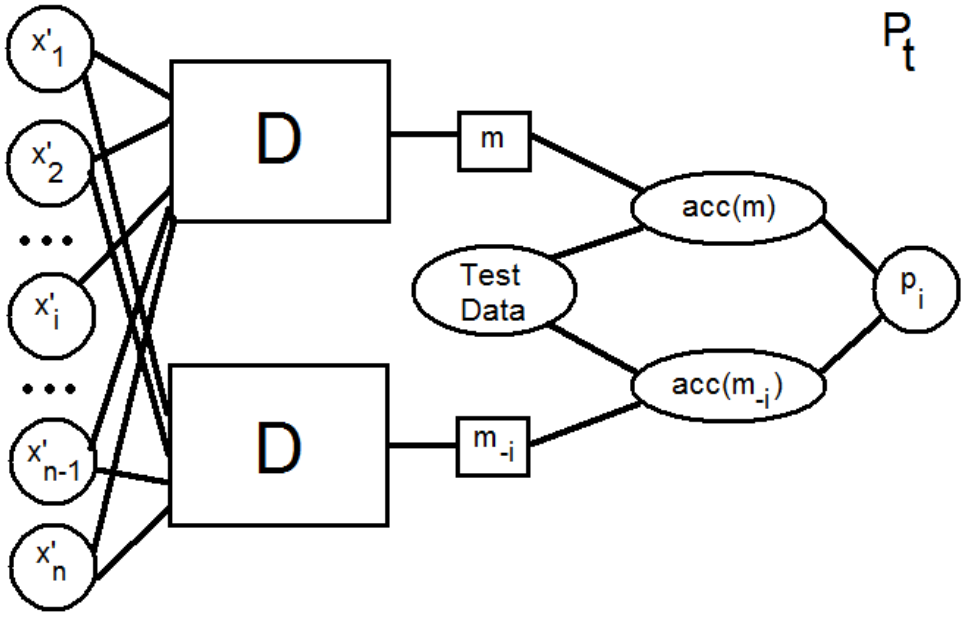


Figure 5.1. Payment calculation for player i

assumption might be to say that we assume all players' data is relevant to the data mining task.

Proof (Incentive Compatibility): We proceed in a similar fashion to the proof of VCG incentive compatibility. For any given i, x_i, X_{-i}, and x_i', we must show that

$$E[u_i(X = x_i \cap X_{-i})] \geq E[u_i(X' = x_i' \cap X_{-i})].$$

The utility of i for X is given by

$$\begin{aligned} u_i(x_i, D(X)) &= \max\{v_i(D(x)) - v_i(D(x_i)), 0\} \\ &\quad -p_i(X, D(X)) - c(D) \end{aligned}$$

where

$$p_i(X, D(X)) = \sum_{j \neq i} v_j(D(X_{-i})) - \sum_{j \neq i} v_j(D(X)) - c(D).$$

Likewise,

$$\begin{aligned} u_i(x_i', D(X')) &= \max\{v_i(D(X')) - v_i(D(x_i)), 0\} \\ &\quad -p_i(X', D(X')) - c(D) \end{aligned}$$

where

$$p_i(X', D(X')) = \sum_{j \neq i} v_j(D(X_{-i})) - \sum_{j \neq i} v_j(D(X')) - c(D).$$

Over expectation, in order for incentive compatibility to exist, this requires that

$$\begin{aligned} &E[\max\{v_i(D(X)) - v_i(D(x_i), 0\}] + E[\textstyle\sum_{j \neq i} v_j(D(X))] \geq \\ &E[\max\{v_i(D(X')) - v_i(D(x_i)), 0\}] + E[\textstyle\sum_{j \neq i} v_j(D(X'))]. \end{aligned}$$

By our assumption that the expected value of $v_k(D(X'))$ (for all k decreases as X' differs from X, we know that $E[\sum_{j \neq i} v_j(D(X))] \geq E[\sum_{j \neq i} v_j(D(X'))]$. We also know that $E[\max\{v_i(D(X)) - v_i(D(x_i), 0\}] \geq E[\max\{v_i(D(X')) - v_i(D(x_i)), 0\}]$, since either the last expression is zero, in which case the first expression is greater than or equal to zero, the last expression is greater than zero, in which case the first expression is greater than or equal to the last expression by our assumption. Therefore, the mechanism is incentive compatible.

Proof (Individual Rationality): To show that the mechanism is individually rational, we need only show that the mechanism has a utility of at least zero (since we have normalized the reservation utility to zero). Note, once again, that the utility of player i is given by

$$\begin{aligned} u_i(x_i, D(X)) &= \max\{v_i(D(x)) - v_i(D(x_i)), 0\} \\ &\quad -p_i(X, D(X)) - c(D) \end{aligned}$$

Since $\max\{v_i(D(x)) - v_i(D(x_i)), 0\}$ is at least zero, and $-c(D)$ is offset by the term in

$p_i(X, D(X))$, we need only show that $E[\sum_{j \neq i} v_j(D(X_{-i})) - \sum_{j \neq i} v_j(D(X))] \leq 0$. Note that, X_{-i} has a nonzero distance from X. Therefore, by our assumption, $E[v_j(D(X_{-i}))] \leq E[v_j(D(X))]$ for all j. Because of this, $E[\sum_{j \neq i} v_j(D(X_{-i})) - \sum_{j \neq i} v_j(D(X))] \leq 0$, and the mechanism is individually rational.

5.2.1 The Cooperative Solution

In order to encourage the truthful sharing of data in the cooperative setting, we employ the Shapley value. Specifically, we offer the players the Shapley value of their contribution to the data mining process, as determined by the independent test set held by the mediator. In order to calculate this Shapley value, the mediator computes $2^{|N|} - 1$ data mining models. Each of these models corresponds to a different non-empty subset of N, and uses only the data for the players belonging to that subset. We then use the formula

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} (v_c(S \cup \{i\}) - v_c(S))$$

to calculate the Shapley value. We then add this value to the individual payout of each player. Figure 5.2 shows the process involved in computing each player's Shapley value.

Theorem 5.2 The above mechanism is expected to be individually rational and incentive compatible under the assumption outlined in Theorem 5.1.

*Proof (Individual Rationality)*Recall that the Shapley value is individually rational if the coalitional game is *superadditive*. Also recall that our assumption states that, over expectation, the best model comes from the data closest to the true data. Now, let $S, T \subseteq N$, where $S \cap T = \emptyset$. We claim that

$$E[v_c(S \cup T)] \geq E[v_c(S) + v_c(T)]$$

By the above formula for v_c,

$$v_c(S) = |S| acc(D(X_S)) - \sum_{i \in S} acc(D(x_i))$$

and

$$v_c(T) = |T|acc(D(X_T)) - \sum_{i \in T} acc(D(x_i))$$

Now,

$$v_c(S \cup T) = |S \cup T|acc(D(X_{S \cup T})) - \sum_{i \in S \cup T} acc(D(x_i)) =$$

$$(|S| + |T|)acc(D(X_{S \cup T})) - \sum_{i \in S \cup T} acc(D(x_i))$$

Since S and T are disjoint,

$$\sum_{i \in S} acc(D(x_i)) + \sum_{i \in T} acc(D(x_i)) = \sum_{i \in S \cup T} acc(D(x_i))$$

Because of this, we need only confirm that

$$(|S| + |T|)acc(D(X_{S \cup T})) \geq |S|acc(D(X_S)) + |T|acc(D(X_T))$$

Since, by our assumption,

$$E[acc(D(X_{S \cup T}))] \geq E[acc(D(X_S))]$$

and

$$E[acc(D(X_{S \cup T}))] \geq E[acc(D(X_T))]$$

we have, over expectation, that

$$|S|acc(D(X_{S \cup T})) + |T|acc(D(X_{S \cup T})) =$$

$$(|S| + |T|)acc(D(X_{S \cup T})) \geq |S|acc(D(X_S)) + |T|acc(D(X_T))$$

Therefore, the function is expectedly superadditive, and the mechanism is expectedly individually rational.

Proof (Incentive Compatibility): Given that the mechanism is individually rational, we need only confirm that the grand coalition N is not out-performed by any subcoalition. Let $S \subseteq N$. Because the game is expectedly superadditive, we have

$$E[v_c(N)] \geq E[v_c(S) + v_c(N \setminus S)$$

Therefore, the mechanism is incentive compatible.

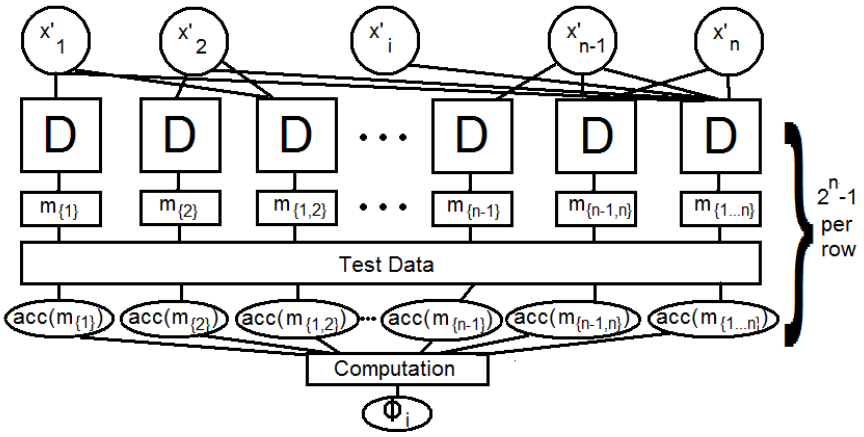


Figure 5.2. Shapley value calculation for player i

5.2.2 A Note on Efficiency

One major issue with the Shapley value is that computing the Shapley value over the test data requires building $2^{|N|} - 1$ models (as the empty model does not need to be built), which is of course extremely unwieldly for large N. There exist several good approximation algorithms for the Shapley value, the latest of which is found in (Fatima, Wooldridge, and Jennings 2008). If the exact computation is required, however, the computation of the Shapley value can be parallelized using a cloud architecture, as each model computation is independent of the others. One method to parallelize the computation is as follows:

- For each non-empty subset of players, $S \subseteq N$, create a process to build the model on the data belonging to that non-empty subset of players. We label each process by its set S.
- Each process computes the utility value (the accuracy measure) of its model on the test set.
- Each process S where $|S| = 1$ sends its value to the processes S' where $|S'| = 2$ and $S \subset S'$.

- Each process S where $|S| = 2$ receives the values from the lower processes, and computes the difference between its value and the values it receives. It labels the differences based on the player missing from the value calculation d_i. It then multiplies this d_i by $\frac{(|S|-1)!(n-|S|)!}{n!}$ to get a partial value for the computation of ϕ_i, which we will call ϕ_{i_S}. It then sends its utility value, and the partial computations ϕ_{i_S} to all processes S' where $|S'| = 3$ and $S \subset S'$.

- Each process S where $|S| = k$ receives the values from the lower processes and computes the partial Shapley values in the same manner, adding the results of ϕ_{i_S} to the other values of $\phi_{i_{S'}}$ it receives. It then sends its values to the processes S' where $|S'| = k + 1$.

- Process N simply calculates the final Shapley values from the partial Shapley values and its own value, finally returning the results.

As there are N layers of processes in the calculation, and at most N^2 additions and N subtractions taking place in each process, if the process were completely parallelized (that is, each process on a separate machine), the entire process would take only $O(N^3)$ time, after the time it takes to spawn the individual processes. However, since the number of processes is exponential, this is only feasible if N is small. It is expected that for most real world applications, N will be small, and the overall calculation will be feasible.

In addition to efficiency concerns over the Shapley value, there are also some concerns over the efficiency of the secure computation of the data mining models themselves. Certain secure implementations of the data mining functions may themselves be prohibitively expensive. However, there do exist relatively fast implementations for some. For example, the Naive Bayes classification algorithm can be implemented quite efficiently, without using homomorphic encryption, with random data hiding techniques (Kantarcioglu and Vaidya 2003).

5.3 Experiments

Having proven that the mechanisms are incentive compatible under reasonable assumptions, we now set out to show how the mechanism performs in practice. As previously mentioned, the assumption that the best model is given by the true data is not always correct. This can happen when the data is stacked in particular ways, or due to the simple overfitting phenomenon. However, most of data mining relies on this assumption when aggregating results. We therefore ran a series of experiments on real data to show the mechanism's practical viability.

5.3.1 Methodology

We tested the mechanism on the following three different data mining models: naive Bayes classification, ID3 decision tree classification, and support vector machine (SVM) classification. For the decision tree and SVM, we used the Weka data mining library(Hall, Frank, Holmes, Pfahringer, Reutemann, and Witten). We used three different data sets from the UC Irvine Machine Learning Repository(Asuncion and Newman 2007).

Adult(census-income). This is the data set used in chapter 4 for naive Bayes and C4.5. For our purposes, we included only 20,000 randomly selected rows of this data set, 18,000 for training, and 2,000 for the independent test set. In addition, certain fields were omitted due to their continuous nature, and others (such as age) were generalized to more discrete values to prevent overfitting.

German-credit. This data set contains credit applications in Germany, and classifies people as either a good credit risk (+) or a bad credit risk (-). There were two continuous attributes (duration and amount) which we generalized to avoid overfitting.

Car-evaluation. This data set takes the characteristics of cars and classifies them as unacceptable, acceptable, good, or very good. Since we wished to deal only with binary classification problems for the purposes of this experiment, we generalized the class into

simply unacceptable (unacc) or acceptable (acc) with those vehicles which were originally evaluated as good or very good being listed as acceptable. No attributes were adjusted.

We chose to use real data, rather than fabricated data, because the mechanisms in question deal with the actions of real people. The incentive to lie for an individual row of the data set is not in play here. We are looking at the incentive for the owner of several pieces of data to lie about his input to a classification process. It is the potential for knowledge discovery, and the exclusive discovery thereof, which would drive someone to lie about the data they have.

In each case, 10% of the data was set aside as an independent test set (to be used by the mediator).

Each training data set was partitioned vertically into three pieces, each piece having as close to the same number of fields as possible. Each of these pieces was designated as belonging to a player. Thus, all the experiments involve three parties, for simplicity.

For each data set and data mining method, we first ran 50 trials to determine the overall accuracy using the truthful data, and the estimated payouts to each player in this case. In order to combat overfitting, each trial consisted of the classification of 20 separate bootstrap samples of the test data (that is, a sample with replacement). The size of these samples was 25% of the test set size.

After this, for each player, we varied the truthfulness of that player's data. Any choice of x'_i is either honest or dishonest. However, the dishonest choices may have varying degrees of dishonesty, with some applying merely a small perturbation to the input, and some blatantly dishonest about every data row. We classify moves by the amount of dishonesty in them. Let $x'_i[k]$ refer to an input for which k times the total number of rows in the input are falsified, that is, k is the fraction of falsified rows in the data. Thus, $x'_i[.01]$ would be an input for which a mere 1% of the data would be falsified. $x'_i[1]$, on the other hand, would essentially be a random set drawn from the domain.

In order to test the results of the falsification (or, equivalently, the perturbation) of the data, we tested the model with several different perturbation values. For each player i, we used $x'_i[.01]$, $x'_i[.02]$, $x'_i[.04]$, $x'_i[.08]$, $x'_i[.16]$, $x'_i[.32]$, $x'_i[.64]$, and $x'_i[1]$. Note that only one

player's data was perturbed at any given time. This was because we wished to determine what a player's unilateral deviation would do when other players were truthful. To calculate the expected payout for player i, we would subtract the overall accuracy for the model without the data belonging to player i from the overall accuracy of the full model.

To determine what happens in the cooperative game setting, we ran several additional experiments. Using the same three data sets (census-income, german-credit, and car-evaluation), and the same three data mining models (naive Bayes, ID3 decision tree, and SVM) we determined the Shapley value for each player given every possible subset of truthful players, over 50 trials. If a player is indicated as truthful, then the player truthfully shared the data, and if the player is listed as a liar, then the player has replaced the data with randomly generated values from the set of possible values. This "full lie" was chosen because it is intuitively the most likely to disrupt the coalition of the truthful.

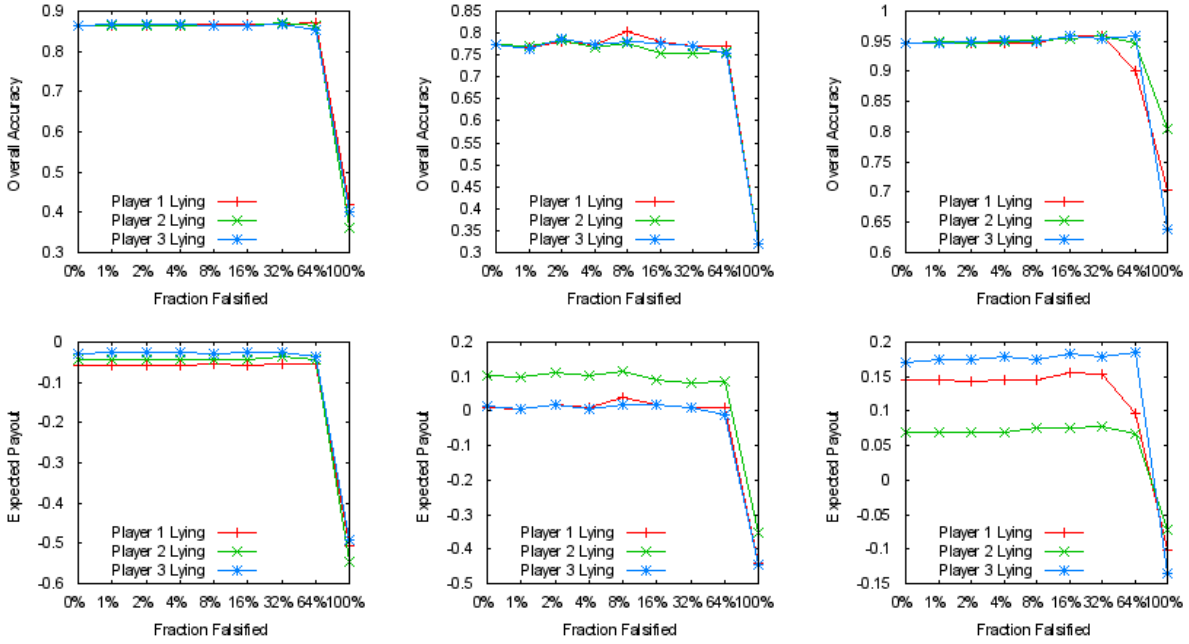
5.3.2 Results

The Non-Cooperative Case

Figures 5.3 through 5.5 show the overall accuracy and estimated payouts to each player for each model, data set, and perturbation. For the estimated payouts, each line shows the payout to the player that is lying, for each perturbation value.

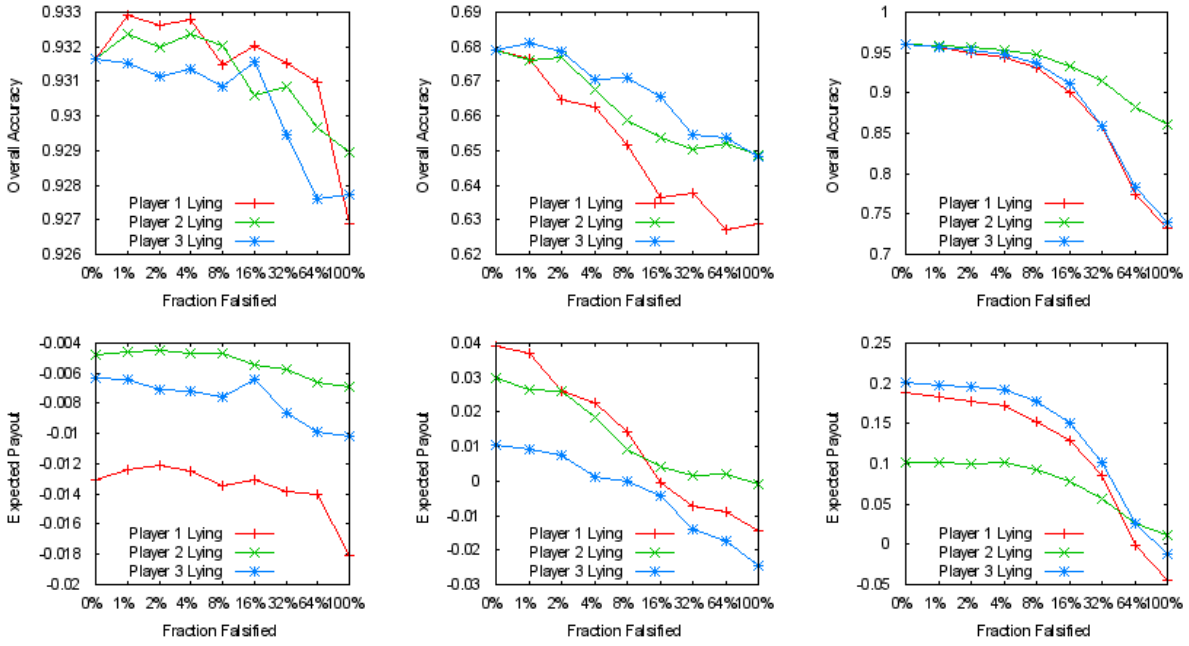
In the vast majority of cases, deviation from the truth produces a lower payout, on the average. Some cases produce a small payout increase on the average, however. Smaller deviations have a higher probability of increasing payout than larger deviations. In practice, a small (1-4%) deviation from the truth has the effect of reducing the impact of overfitting, and can result in a slightly more accurate classifier. However, rarely is the amount gained significant.

It is worth mentioning that in several cases, the calculation would not qualify as individually rational without further subsidy. For example, the Adult data set, under naive Bayes classification and ID3 decision tree classification, produces payouts for each player which



Census-Income German-Credit Car-Evaluation

Figure 5.3. Results for Naive Bayes Classification



Census-Income German-Credit Car-Evaluation

Figure 5.4. Results for ID3 Decision Tree Classification

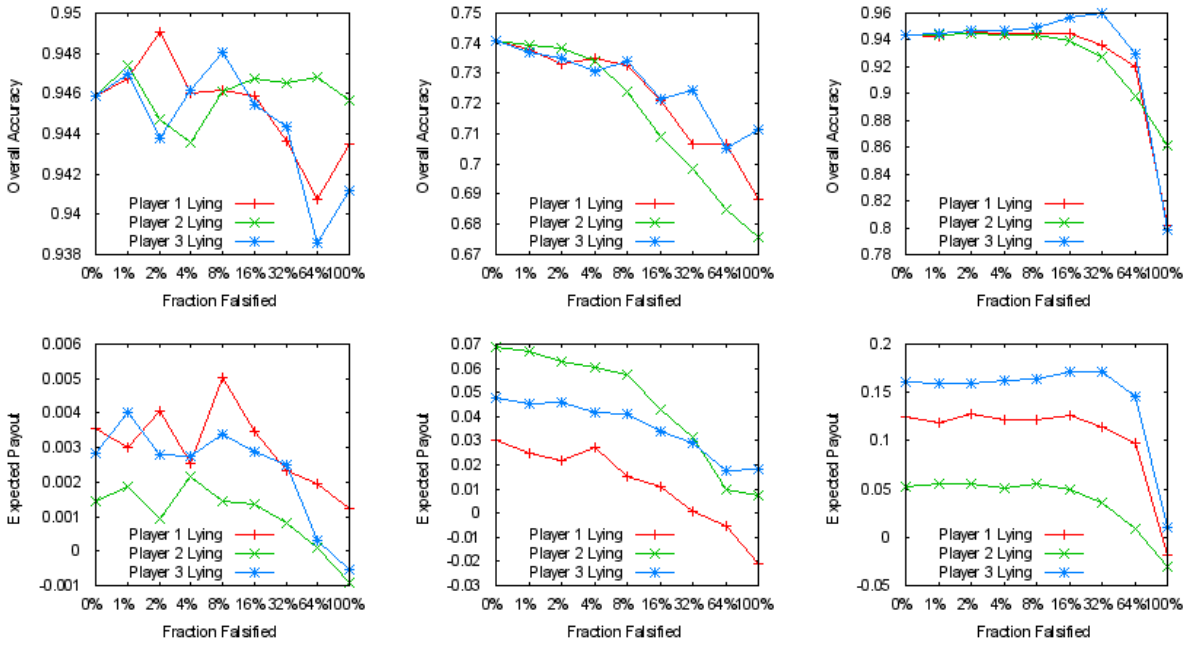


Figure 5.5. Results for SVM Classification

are negative. This means that the addition of a third player's data decreases the accuracy of the classifier. This is likely due to the presence of many fields in the data. While each player's fields perform well, combining the fields results in a slight reduction in accuracy due to redundant or irrelevant fields.

There are a few exceptions to the generalization about small deviations and small payout increases, such as the volatile looking graph for the estimated payouts for the SVM data mining on the Adult data set, as the graph moves up and down very quickly, and does appear to increase sharply in a few places. However, the scale of this graph shows that this fluctuation is actually very small. The difference between any two points on this graph is no more than 0.6% in terms of the overall accuracy of the classifier.

While a risk-neutral player might attempt to perturb the data slightly to gain a slight average profit, a risk-averse player would certainly never perturb the data. In all cases, at least one bootstrap sample produced a lower classifier accuracy for any perturbed data.

Therefore, if the player is risk-averse, then the player would provide true data, since otherwise there would be a risk of losing profit.

The Cooperative Case

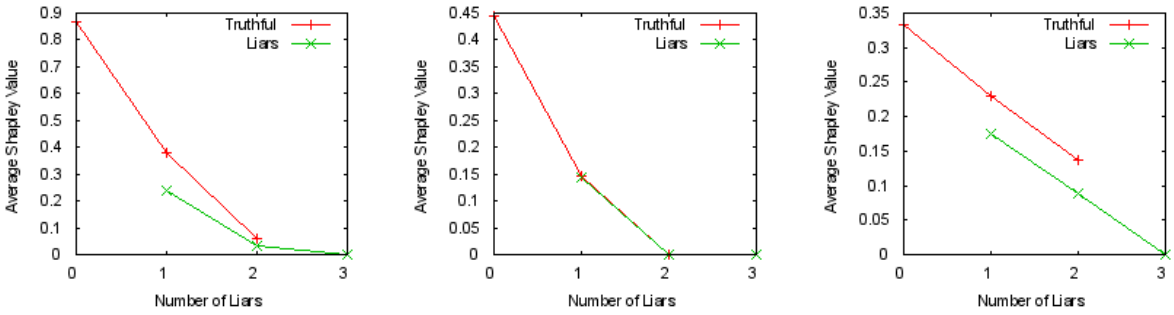
For the cooperative case, the results are documented in figures 5.6 through 5.8. We show, in each graph, the average Shapley value achieved for each player for each number of liars for truthful and lying players. All possible subsets of lying players were tried, but the two-dimensional nature of paper prevents the meaningful graphing of all data points. We used this projection to convey the findings of the data without resorting to a listing of data points.

Without exception, the Shapley value for a given player decreased when the player lied. The results for the truthful players wildly varied. Sometimes a lie would improve the values for the other, truthful players, other times the lie would reduce the value for the remaining players. In only very few cases would the average value of a truthful player's Shapley value prove lower than the liars' values, but even in this case, moving to a lie would only reduce the Shapley value. In many cases, the Shapley value would become zero when many players lied, this is because no player's data improved upon any other player's data.

For one combination of data and model (census-income, decision tree), the Shapley value is always negative. This is most likely due to the fact that the model overfits the training data quite severely for the decision tree, likely due to the vastly larger size of the census-income data set. Even with the values becoming negative, however, lying still decreased the Shapley value for the liar.

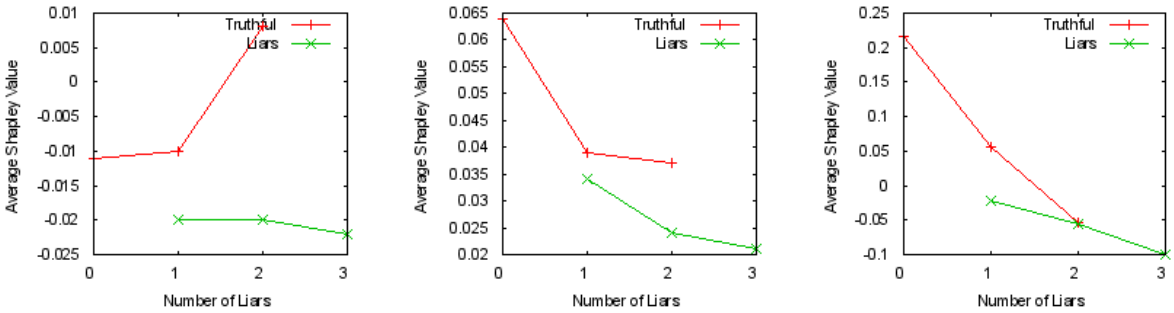
5.4 Conclusions

We have shown that, under a reasonable assumption, our mechanisms which reward players based on their contribution to the model is incentive compatible. We then determined the usefulness of the mechanism in practice by running our mechanism using real data. This



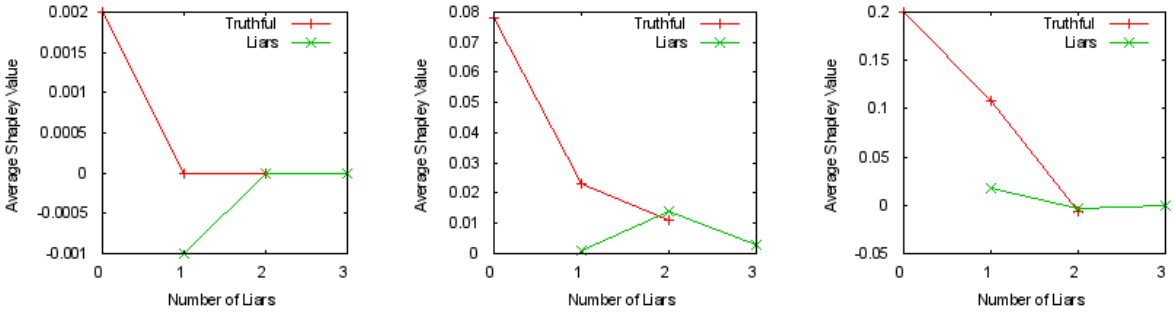
Census-Income German-Credit Car-Evaluation

Figure 5.6. Cooperative Results for Naive Bayes Classification



Census-Income German-Credit Car-Evaluation

Figure 5.7. Cooperative Results for ID3 Decision Tree



Census-Income German-Credit Car-Evaluation

Figure 5.8. Cooperative Results for SVM Classification

shows that, while the assumption used in the incentive compatibility proof is not always strictly true, the mechanism yields proper motivation for the vast majority of cases.

While our primary goal has been to ensure that players truthfully reveal their data, one could also take a different approach to the problem. If a deviation from the truth affords a player a payout advantage, then this means that the deviation has necessarily increased the overall accuracy of the final classifier. So, in the cases where it is advantageous to lie, we have created a better classifier than the truthful data would provide! Thus, while the mechanisms do not *guarantee* truthfulness *every* time, in the cases where it does not, it results in a better classifier. If the goal of the process is then changed to the creation of the best model, rather than ensuring truth, the mechanism works even better.

CHAPTER 6
GENERAL RESULTS IN INCENTIVE BASED COMPUTATION

We now turn our attention to the general process of multiparty computation. While the process in chapter 5 dealt with data mining, we contend that the non-cooperative process can be generalized to encourage truthful behavior in arbitrary multyparty computation. We only require that there be some method to evaluate the utility of a given result which can be evaulated by an uninterested party.

6.1 A Game Theoretic Formalization of Multiparty Computation

In order to show that we can encourage honesty in multiparty computation, we must first formally define the game associated with multiparty computation.

Definition 5 *Mediated Multiparty Computation Game*

Players: $P_1, P_2, ..., P_n$, *and a mediator* P_t.

Preconditions: Each player $P_i \in \{P_1, ..., P_n\}$ *has* x_i, *a input to a function* $f(x_1, ..., x_n)$. P_t *is another party who is bound to compute the function* f *securely. We assume there exists some valuation function for a given result,* $v(result)$, *and that this valuation function is the same for all players (that is, for all* i, $v_i(result) = v(result)$. *There also exists a function* $V_f(result)$ *which gives a good estimate for the intrinsic utility of the result,* $v(result)$. *Formally, we state that the expected value of* $V_f(result)$ *is within* ϵ *of* $v(result)$. *It is reasonable that* P_t *would be able to have such a function in many cases, especially in cases where the result has a real-world consequence that can be observed.*

Game Progression:

1. Each player $P_i \in \{P_1, ..., P_n\}$, selects x'_i, which may or may not equal be equal to x_i, or chooses not to participate. These inputs are committed. Define X to be the vector of original values x_i, and X' to be the vector of chosen values x'_i.

2. Players send X' to P_t for secure computation of the function. P_t then computes $f(X')$.

3. All players receive the function result, $m = f(X')$.

Payoffs: For each $P_1...P_n$, define the utility of a participating player as the following:

$$u_i(x_i, f(X')) = max\{v_i(m) - v_i(f(x_i)), 0\} - p_i(X', m) - c(f)$$

$v_i(m)$ is the intrinsic utility of the function result. In the case where a function result is required to be exact, this will be nonzero only for the correct value, and zero for all others. It is also possible that an incomplete vector would lead to a non-evaluation of the function. We denote a failed evaluation by $v_i(X') = \perp$. Thus, we also define $v_i(\perp) = 0$. Other functions might have varying intrinsic values. We normalize each player's reservation utility, that is, the utility received if the player chooses not to participate, to zero. This can be done without loss of generality by subtracting the reservation utility (which is $v_i(D(x_i))$, based on the accuracy of the model based only on one's own data), from the valuations in the mechanism. Note that a player will always recieve at least this much utility, so we obtain the expression $max\{v_i(m) - v_i(D(x_i)), 0\}$. $p_i(X', m)$ is the amount paid by P_i, based on the inputs and the results. Note that if p_i were to be negative, it would mean that P_i receives money instead. $c(f)$ is the computational cost of computing f. Since f is securely computed, there will be some cryptography involved in the computation of the model, hence computational cost should be considered.

6.2 The VCG Solution

To motivate players to truthfully reveal the information, we propose the following:

1. In addition to computing the data mining model, P_t also computes $f(X'_{-i})$ for each P_i, that is, the data mining function without using the data provided by player i.

2. For each P_i, we let $p_i(X', m) = \sum_{j \neq i} V_f(f(X'_{-i})) - \sum_{j \neq i} V_f(m) - c(f)$, where V_f is calculated by P_t. This pays each player an amount expectedly equal to the difference in intrinsic value of the function evaluated with the given input, minus the value of the function evaluated with out that player's input, essentially rewarding each player based on their own contribution to the evaluation. We include the $-c(f)$ term in order to balance out the cost of the calculation. Figure 1 shows the process used to calculate the payment for a given player i.

Theorem 7.1 The above mechanism motivates participating players to truthfully reveal their inputs, under the following assumption:

Assumption: For each player i, the probability of an increase in the intrinsic value of $f(x'_i, X_{-i})$ decreases significantly with the distance between the player's actual input and the input the player provides to the computation. More formally, we may state that the expected value of the classifier's accuracy does not increase with said distance. Mathematically, for $X = x_i \cup X_{-i}$ and $X' = x'_i \cup X_{-i}$, this can be written as

$$E[v_i(f(X))] \geq E[v_i(f(X'))] + f(dist(X, X'))$$

where f is a non-negative, increasing function for all i, x_i, x'_i and X_{-i}.

Again, we feel this assumption is appropriate, since, unless the function's value is independent of a player's input, a player cannot easily find a $x'_i \neq x_i$ such that $v_i(f(X')) \approx v_i(f(X))$. If a the function's value is independent of a player's input, then that player has no incentive to provide the input, and therefore will not participate in the protocol. As the player's input was not necessary for the computation, we feel this does not cause any problems.

Proof (Incentive Compatibility): Our proof closely resembles our proof of incentive compatibility in chapter 5. For any given i, x_i, X_{-i}, and x'_i, we must show that

$$E[u_i(X = x_i \cap X_{-i})] \geq E[u_i(X' = x'_i \cap X_{-i})].$$

The utility of i (u_i) for X is given by

$$\begin{aligned} u_i(x_i, f(X)) &= \max\{v_i(f(X)) - v_i(f(x_i)), 0\} \\ &\quad -p_i(X, f(X)) - c(f) \end{aligned}$$

where

$$p_i(X, f(X)) = \sum_{j \neq i} V_f(f(X_{-i})) - \sum_{j \neq i} V_f(f(X)) - c(f).$$

Likewise,

$$\begin{aligned} u_i(x_i', f(X')) &= \max\{v_i(f(X')) - v_i(f(x_i)), 0\} \\ &\quad -p_i(X', f(X')) - c(f) \end{aligned}$$

where

$$p_i(X', f(X')) = \sum_{j \neq i} V_f(f(X_{-i})) - \sum_{j \neq i} V_f(f(X')) - c(f).$$

Over expectation, in order for incentive compatibility to exist, this requires that

$$\begin{aligned} &E[\max\{v_i(f(X)) - v_i(f(x_i), 0\}] + E[\textstyle\sum_{j \neq i} V_f(f(X))] \geq \\ &E[\max\{v_i(f(X')) - v_i(f(x_i)), 0\}] + E[\textstyle\sum_{j \neq i} V_f(f(X'))]. \end{aligned}$$

By our assumption that the expected value of $V_f(f(X'))$ decreases as X' differs from X, we know that $E[\sum_{j \neq i} v_j(f(X))] \geq E[\sum_{j \neq i} v_j(f(X'))]$. We also know that $E[\max\{v_i(f(X)) - v_i(f(x_i), 0\}] \geq E[\max\{v_i(f(X')) - v_i(f(x_i)), 0\}]$, since either the last expression is zero, in which case the first expression is greater than or equal to zero, the last expression is greater than zero, in which case the first expression is greater than or equal to the last expression by our assumption. Therefore, the mechanism is incentive compatible.

Proof (Individual Rationality): To show that the mechanism is individually rational, we need only show that the mechanism has a utility of at least zero (since we have normalized the reservation utility to zero). Note, once again, that the utility of player i is given by

$$\begin{aligned} u_i(x_i, f(X)) &= \max\{v_i(f(x)) - v_i(f(x_i)), 0\} \\ &\quad -p_i(X, f(X)) - c(f) \end{aligned}$$

Since $\max\{v_i(f(x))-v_i(f(x_i)), 0\}$ is at least zero, and $-c(f)$ is offset by the term in $p_i(X, f(X))$, we need only show that $E[\sum_{j\neq i} V_f(f(X_{-i})) - \sum_{j\neq i} V_f(f(X))] \leq 0$. Note that, X_{-i} has a nonzero distance from X. Therefore, by our assumption, $E[v_j(f(X_{-i}))] \leq E[v_j(f(X))]$ for all j. Because of this, $E[\sum_{j\neq i} v_j(f(X_{-i})) - \sum_{j\neq i} v_j(f(X))] \approx E[\sum_{j\neq i} V_f(f(X_{-i})) - \sum_{j\neq i} V_f(f(X))] \leq 0$, and the mechanism is individually rational. ∎

6.3 Offloading Computation from the Trusted Party

Our computation process has one major drawback: the trusted party must compute $n + 1$ function results. Here, we show that all the function results except for one ($f(X')$ itself) can be computed by the parties whose inputs are involved (that is, for $f(X'_{-i}$, all parties except P_i), without the need for interaction with the mediator P_t. Formally, we state:

Theorem 7.2 In the above computation process, the computation of the function $f(X'_{-i})$, and the evaluation thereof ($V_f(X'_{-i})$) may be performed in a distributed, secure manner by the parties $P_{-i} = \{P_1, ..., P_n \backslash \{P_i\}$, and no player in this set has any incentive to behave dishonestly for the calculation.

Proof: As before, the utility of a given player i is given by:

$$\begin{aligned} u_i(x_i, f(X)) &= \max\{v_i(f(x)) - v_i(f(x_i)), 0\} \\ &\quad -p_i(X, f(X)) - c(f) \end{aligned}$$

where

$$p_i(X, f(X)) = \sum_{j \neq i} V_f(f(X_{-i})) - \sum_{j \neq i} V_f(f(X)) - c(f).$$

Note that, for any given player i, the function $V_f(f(X_{-j})$ for $j \neq i$ does not affect the payout of player i. If we account for the increased computation cost in both instances of the $c(f)$ term, the payout is not altered by an increase in the computation load. Therefore, since the outcome of the function $f(X_{-j})$ has no bearing on the eventual payout of P_i, P_i has no incentive to lie about inputs to $f(X_{-j})$. Thus, for all players $P_j \in P_{-i}$, no player P_j has incentive to falsely compute $V_f(f(X_{-i}))$. ∎

The last computation, $V_f(f(X))$, cannot be calculated by any players, as they would all have incentive to report a value greater than the actual valuation. This is due to the fact that the payout directly increases with the difference between $V_f(f(X))$ and $V_f(f(X_{-i}))$, thus increasing $V_f(f(X))$ would increase every player's payout. Therefore, only an outside party can be trusted to compute $V_f(f(X))$.

These two results show that there exists a class of functions which admits a mechanism enforcing honesty among non-cooperating parties.

CHAPTER 7
GAME THEORETIC QUERY VERIFICATION ON OUTSOURCED DATA

7.1 Introduction

As the amount of data that we generate increases, so does the time and effort necessary to process and store the data. With an increase in time and effort comes an increase in monetary cost. To this end, many have turned to outsourcing their data processing to "the cloud." Cloud computing services are offered by many large companies, such as Amazon, IBM, Microsoft, and Google, as well as smaller companies such as Joyent and CSC. For example, Google (Google 2011) recently launched the Google BigQuery Service, which is designed for exactly this purpose: outsourced data processing. The distributed nature of these cloud services shortens data processing time significantly. In addition, these cloud services provide a massive amount of data storage.

In a perfect world, these cloud providers would impartially devote all the computation necessary to any task paid for by the subscribers. In such a world, the querying process would look like figure 7.1 (minus the verifier), where the subscriber outsources the data D to the cloud, sends queries (Q), and the cloud does the necessary calculations and returns the result ($Q(D)$). However, a cloud provider is a self-interested entity. Since it is very difficult for the users of the cloud to see the inner workings of the cloud service, a cloud provider could "cut corners," delivering a less accurate or incomplete computation result which would take fewer system resources to compute. This would, of course, save computational resources for the provider, provided the subscriber was unable to tell a false result from a true one. Because of this, query verification, or the assurance of query result correctness, has been identified as one of the major problems in data outsourcing (Sion 2007).

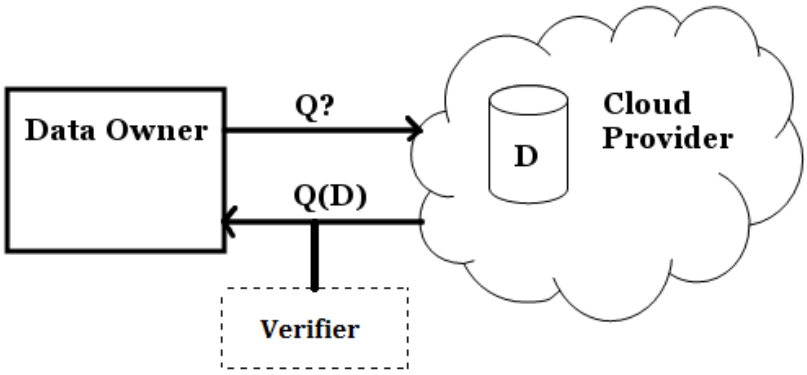


Figure 7.1. Data Outsourcing with Verification

Many techniques have been developed and employed for query verification. In figure 7.1 above, the subscriber sends a query to the outsourcing service, and receives a response. Query verification would then be another process where the subscriber determines if the response is, in fact, the result of the query. The verification process may belong to the owner, or it may be another process entirely. In any case, the verifier aims to make sure that the outsourced server responded correctly. These verification techniques range from simple to extremely complex, and generally rely on the subscriber storing some sketch of the data (much smaller in size), or some cryptographic protocols. Such protocols do a good job verifying the data, but are often slow, or only work with specific types of queries. Many of them assume that the subscriber knows which queries he will execute in advance, so that a sketch can be created for each one. None of these, however, consider the heart of the problem: the self-interest of the parties.

The problem of data outsourcing, and the resultant query verification, is fundamentally a problem of *incentives*. A cloud subscriber wants to get the result of his queries accurately and efficiently, with as low a cost as possible. A cloud provider, however, is most concerned about the profitable use of its computing resources. These incentives can be at odds with each other. The natural way of analyzing competing incentives is to use *game theory*. An

interaction between parties is cast as a game, where players use strategy to maximize their gains. The gains from an interaction can be offset artificially by contracts, which can be enforced by law. These adjustments can make actions which were once profitable, such as "cutting corners" in a calculation, less profitable through the use of penalties. The contracts, therefore, aim not to detect whether a cloud provider is cheating, but to remove the incentive for the provider to cheat altogether.

We propose a game theory-based approach to query verification on outsourced data. We model the process of querying outsourced data as a game, with contracts used to enforce behavior. Data outsourcing does not take place in a vacuum. Service Level Agreements (SLAs) exist for all types of cloud services(Patel, Ranabahu, and Sheth 2009), and are enforceable contracts in court. Thus, we can augment the SLA with an incentive structure to encourage honest behavior. Using a very simple query verification technique, we show that even the threat of verification is enough to deter cheating by a cloud provider.

We consider the case where multiple, non-colluding cloud providers exist. Non-colluding means that the cloud providers do not share information. We believe this is realistic, since cloud providers are competing entities and do not wish to share data with their competitors. In this scenario, we show that without the use of special verification techniques, a data owner can guarantee correct results from rational cloud providers, while incurring an additional cost that is only a small fraction of the overall computation cost.

Our contributions can be summarized as follows:

- We develop a game theoretic model of query verification on outsourced data.
- We show that the model has an equilibrium where the cloud provider behaves honestly.
- Finally, we show that our incentives can improve the expected runtime of *any* query verification method, making it extremely flexible.

This chapter does not consider the privacy of the outsourced data (similar to (Canetti, Riva, and Rothblum 2011)). However, any privacy-preserving technique for outsourcing data

could still be used in our framework. The use of our game theoretic techniques will not affect the privacy-preserving properties of such schemes.

Portions of the work in this chapter have been published in "Contractual Agreement Design for Enforcing Honesty in Cloud Outsourcing (Nix and Kantarcioglu 2012a)." The author of this dissertation was the principal author in the aforementioned work, and has permission from the co-author to include this work.

7.2 The First Solution

We consider the case where multiple non-colluding cloud providers exist. This means that the parties do not exchange strategies and do not exchange information. Since multiple providers exist, our strategy will be to choose two of them, checking the results of one against the other. We model the query verification process as a game. The game has the following characteristics:

Players (3): the Data Owner(O), and two outsourced servers (S_1 and S_2).

Actions: The data owner begins the game by selecting a probability α, and declares this probability to the servers. He then sends the query (Q) to one of the two servers, with equal probability. With probability α he also sends the query to the other server. If server S_i receives the query, they then respond to the query with either $Q(D)$, that is, the query result on the database D, or $Q'_i(D)$ which is some result other than $Q(D)$. We apply the subscript i to Q' to indicate that one player's method of cheating is different from the other players' method of cheating. We denote the honest action as h, and the cheating action as c. These actions are depicted in figure 7.2.

Information: Data Owner O has given his database D to S_1 and S_2, with an HMAC message authentication code appended to each tuple. Any message authentication scheme would work here, but its purpose and only effect is that it maintains the integrity of the data. This means that the servers cannot alter any tuples and cannot add any tuples without being detected. The players have entered into an agreement (a contract) before the game, and the

contents of this contract are known to all players. The contract could contain the probability α. We assume that no updates are to be made to the database once they are outsourced (they are outsourced purely for the purposes of querying).

Payoffs: The owner recieves the information value of the results received, given by $I_v(Q)$, where Q is either $Q(D)$ or $Q'_i(D)$, minus the amount paid to the servers $P(Q)$. The servers recieve this payment, minus the cost of computing the query, $C(Q)$. For simplicity's sake, we assume that both outsourcing services have the same cost of computation and receive the same payment for the query. The logic below easily applies to the case where costs are different, but this assumption simplifies the equations involved. These payoffs are additionally adjusted by the aforementioned contract. We assume the reservation utility of all parties is zero, and if any party declines the contract, then none of the parties participate.

We assume that $I_v(Q(D)) \geq (1+\alpha)P(Q)$ and $P(Q) \geq C(Q)$. If this were not the case, then the game would not be individually rational without some outside subsidies (that is, some player's expected payout would be less than zero). In essence, we want to ensure that the data owner would want to pay $(1+\alpha)P(Q)$ to receive the result, and the cloud provider would accept $P(Q)$ for the computation. To do this, we make sure that the value that the data owner places on the query is at least the expected payment, and the cost to the cloud providers is no more than the amount they would be paid. No one takes a loss on the transaction.

We now present two contracts, both of which provide simple solutions to the above game in which neither server has incentive to cheat. The first is very simple and requires no additional computation. The second is intuitively more fair, and thus more likely to be accepted in a real world scenario. Both contracts, however, would be accepted by rational players. It should be noted that both of these contracts are loosely based on the results from Auditing Game II and III in (Rasmusen 2007).

Contract 1 If the owner asks for query responses from both servers, and the results do not match, both servers pay a penalty of F to the owner, and return the money paid for the computation $P(Q)$ as well.

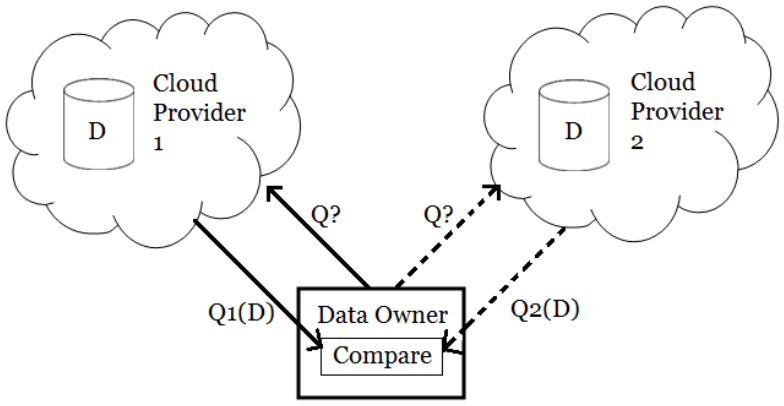


Figure 7.2. The Two-Cloud Query Verification System

Theorem 7.1 The above game with contract 1 has an individually rational, incentive compatible equilibrium in which the servers behave honestly.

Proof: Let $C(Q'_i)$ be the cost of computing Q'_i for S_i. Note that, because S_1 and S_2 do not collude, S_1 does not know Q'_2, and S_2 does not know Q'_1. The only function both know for sure is Q. Without additional knowledge, we can assume that the probability that $Q'_1(D) = Q'_2(D)$ is negligible. For a player to even consider returning Q'_i instead of Q, we must have $C(Q'_i) \leq C(Q)$, since a player will not cheat if they do not gain anything from it. We also assume that $I_v(Q'_i(D)) < 0 < I_v(Q(D))$, since not only is the false result not what the owner asked for, but also appears to be the true result if not verified. If the wrong answer is believed to be correct, this would lead to wrong decisions, and ultimately, financial loss, on the part of the owner. Now, we can define the expected payoffs to each player, where $u_P(x, y)$ is the expected utility for player P when S_1 takes action x and S_2 takes action y. Note that, in these equations and throughout the rest of the paper, we omit the argument D from Q, since D is fixed. We begin with O. If both players are honest (equation 7.1), O recieves the value of the information gained from the query, minus the expected payment for the calculation, $1 + \alpha$ times $P(Q)$. If one player is dishonest (equations 7.2 and 7.3), then with probability α, O detects this and gets both the honest and the dishonest result and the

fine F from both players. With probability $1-\alpha$, he does not detect this, and gets either the correct value or the incorrect value with equal probability. In the event that both players cheat (equation 7.4), they are once again caught with probability α, but in this case, when they are not caught, O receives only bogus values. This results in the following equations:

$$u_O(h,h) = I_v(Q(D)) - (1+\alpha)P(Q) \tag{7.1}$$

$$\begin{aligned} u_O(h,c) = {} & \alpha(2F + I_v(Q) + I_v(Q'_2)) \\ & + (1-\alpha)(\frac{1}{2}(I_v(Q) + I_v(Q'_2)) - P(Q)) \end{aligned} \tag{7.2}$$

$$\begin{aligned} u_O(c,h) = {} & \alpha(2F + I_v(Q) + I_v(Q'_1)) \\ & + (1-\alpha)(\frac{1}{2}(I_v(Q) + I_v(Q'_1)) - P(Q)) \end{aligned} \tag{7.3}$$

$$\begin{aligned} u_O(c,c) = {} & \alpha(2F + I_v(Q'_1) + I_v(Q'_2)) \\ & + (1-\alpha)(\frac{1}{2}(I_v(Q'_1) + I_v(Q'_2)) - P(Q)) \end{aligned} \tag{7.4}$$

For the servers, if both servers are honest (equations 7.5 and 7.8), they receive the payment for the query, minus the cost of the query, provided they are selected to perform the calculation. This selection probability is why the equations below contain $\frac{1}{2}$. Otherwise, they gain nothing and lose nothing. If one player is dishonest, that player (equations 7.7 and 7.10), regardless of whether the other player is honest, with probability α is caught, and loses the fine F. With probability $1-\alpha$, the player is not caught, and gains the payment $P(Q)$, minus the cost of computing his cheat, $C(Q'_i)$, if he is chosen for the computation. If a player is honest while the other player is dishonest (equations 7.6 and 7.9), that player similarly is punished with probability α, but invests a cost of $C(Q)$ instead of $C(Q'_i)$ in the computation. This gives us the following equations:

$$u_{S_1}(h,h) = \frac{1}{2}(1+\alpha)(P(Q) - C(Q)) \tag{7.5}$$

$$u_{S_1}(h,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q)) - \alpha F \tag{7.6}$$

$$u_{S_1}(c,h) = u_{S_1}(c,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q'_1)) - \alpha F \tag{7.7}$$

$$u_{S_2}(h,h) = \frac{1}{2}(1+\alpha)(P(Q) - C(Q)) \tag{7.8}$$

$$u_{S_2}(c,h) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q)) - \alpha F \tag{7.9}$$

$$u_{S_2}(h,c) = u_{S_1}(c,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q_2')) - \alpha F \tag{7.10}$$

We can now find the α for which the expected value for S_1 is less when he cheats than when he is honest, assuming S_2 is honest. By symmetry, this will be the same for S_2. Thus, we set:

$$\frac{1}{2}(1-\alpha)(P(Q) - C(Q_1')) - \alpha F \leq \frac{1}{2}(1+\alpha)(P(Q) - C(Q))$$

Let H represent the quantity $P(Q) - C(Q)$, and H' represent the quantity $P(Q) - C(Q_1')$. Distribute the $(1+\alpha)$ and $(1-\alpha)$ to get:

$$\frac{1}{2}(H') - \frac{\alpha}{2}(H') - \alpha F \leq \frac{1}{2}(H) + \frac{\alpha}{2}(H)$$

Rearranging and combining terms, we get:

$$\begin{aligned} \frac{1}{2}(C(Q) - C(Q_1')) &\leq \alpha F + \alpha P(Q) \\ &\quad + \frac{\alpha}{2}(C(Q) - C(Q_1')) \end{aligned}$$

Let G represent the quantity $C(Q) - C(Q_1')$, that is, the amount the server would gain from cheating. Substituting this in and factoring out an α, we get:

$$\frac{1}{2}G \leq \alpha(F + P(Q) + \frac{1}{2}G)$$

Multiplying through by two, we get:

$$G \leq \alpha(2F + 2P(Q) + G)$$

And, solving for α,

$$\frac{G}{2F + 2P(Q) + G} \leq \alpha \tag{7.11}$$

Since we can define F to be whatever we want in the contract, we can make this minimum α value arbitrarily small. If α is at least this much, then S_1 (and by symmetry, S_2) has no incentive to cheat. If S_2 is not honest, then S_1 has no incentive to be honest, but the payout is less for both (much less, if F is large). Therefore, the best outcome is for both players to behave honestly.

Now, we need to show that choosing α is incentive compatible for O. Given that both players are honest, O's utility is given as:

$$u_O(h,h) = I_v(Q(D)) - (1+\alpha)P(Q)$$

which, by our assumption, is greater than or equal to zero. Thus, it is individually rational for O. If α is increased, it merely decreases this value, so O has no incentive to increase α. If we decrease α, then S_1 and S_2 will see cheating as the more profitable choice, and will begin cheating. This leads to:

$$\begin{aligned} u_O(c,c) = \alpha(2F + I_v(Q_1') + I_v(Q_2')) \\ + (1-\alpha)(\frac{1}{2}(I_v(Q_1') + I_v(Q_2')) - P(Q)) \end{aligned}$$

Now, since our α is less than our prescribed value in equation (7.11), F is bounded above by $\frac{G}{\alpha} - 2P(Q) - G$. Because of this, as α approaches zero, the first term of the above equation decreases. The second term is negative (as $I_v(Q_1')$ and $I_v(Q_2')$ are less than zero), and gets worse as α approaches zero. Thus, if α is less than our prescribed value, O expects to lose value from cheating. So, O has no incentive to deviate from $\alpha = \frac{G}{2F+2P(Q)+G}$.

Now, in practice, O does not know G. Thus, he must choose the smallest α that he knows he can use. Since $P(Q) \geq C(Q) \geq G$, O can choose $\alpha = \frac{P(Q)}{2F+2P(Q)-P(Q)} = \frac{P(Q)}{2F-P(Q)}$.

Now, based on the above analysis, it is clear that a cheater will gain less than an honest player when the value of α is chosen as above, regardless of whether the other player is honest. Thus, S_1 and S_2 have no incentive to cheat, and this is incentive compatible for these players as well.

Now, quickly, a note on individual rationality: O has expected payout of $Iv(Q) - (1+\alpha)(P(Q)$. If this is greater than the reservation utility (zero), then the contract is individually

rational for O. By our initial assumption about the value of the query, this is true. S_1 and S_2, in equilibrium, have an expected payout of $\frac{1}{2}(1+\alpha)(P(Q)-C(Q))$. Again, by the above assumption, this is true.

As this is both incentive compatible and individually rational for all players, this contract creates the best possible equilibrium where S_1 and S_2 do not cheat, and O pays only $(1+\alpha)$ times the price of a single computation (where α is small). ∎

7.3 A More Intuitively Fair Solution

Now, it might seem unfair to punish both players when only one player cheats. The rational player would see the above contract as completely fair, but humans are not always completely rational. Thus, we also examine a contract which identifies the cheater and punishes only the cheater.

Contract 2 If the owner asks for query responses from both servers, and the results do not match, the owner performs a potentially costly audit of the computation. Each server whose result does not match the result given by the owner's process pays a fine F to the owner.

Theorem 7.2 The above game under contract 2 also has an equilibrium where both servers remain honest.

Proof: Let all variables be defined as above, and let $c(A(Q, Q'_1, Q'_2))$ represent the cost of auditing the computation. The payout functions associated with this contract are as follows:

We begin with O. If both players are honest (equation 7.12), O recieves the value of the information gained from the query, minus the expected payment for the calculation, $1+\alpha$ times $P(Q)$. If one player is dishonest (equations 7.13 and 7.14), then with probability α, O detects this and gets both the honest and the dishonest result and the fine F from the dishonest player. In this case, he also pays for a costly audit ($c(A(Q, Q'_1, Q'_2))$ to determine which player cheated. With probability $1-\alpha$, he does not detect this, and gets either the correct value or the incorrect value with equal probability. In the event that both players

cheat (equation 7.15), they are once again caught with probability α, and both pay the fine. However, O only recieves false values, and still pays for the audit. This results in the following equations:

$$u_O(h,h) = I_v(Q(D)) - (1+\alpha)P(Q) \tag{7.12}$$

$$u_O(h,c) = \alpha(F + I_v(Q) + I_v(Q'_2) - c(A(Q,Q'_1,Q'_2))) + (1-\alpha)(\frac{1}{2}(I_v(Q) + I_v(Q'_2)) - P(Q)) \tag{7.13}$$

$$u_O(c,h) = \alpha(F + I_v(Q) + I_v(Q'_1) - c(A(Q,Q'_1,Q'_2))) + (1-\alpha)(\frac{1}{2}(I_v(Q) + I_v(Q'_1)) - P(Q)) \tag{7.14}$$

$$u_O(c,c) = \alpha(2F + I_v(Q'_1) + I_v(Q'_2) - c(A(Q,Q'_1,Q'_2))) + (1-\alpha)(\frac{1}{2}(I_v(Q'_1) + I_v(Q'_2)) - P(Q)) \tag{7.15}$$

For the servers, if both servers are honest (equations 7.16 and 7.19), they receive the payment for the query, minus the cost of the query, provided they are selected to perform the calculation. This selection probability is why the equations below contain $\frac{1}{2}$. Otherwise, they gain nothing and lose nothing. If one player is dishonest, that player (equations 7.18 and 7.21), regardless of whether the other player is honest, with probability α is caught, and loses the fine F. With probability $1-\alpha$, the player is not caught, and gains the payment $P(Q)$, minus the cost of computing his cheat, $C(Q'_i)$, if he is chosen for the computation. In this case, if a player is honest while the other player is dishonest (equations 7.17 and 7.20), the player is not punished, and therefore receives exactly the same payment as if both players were honest. This gives us the following equations:

$$u_{S_1}(h,h) = \frac{1}{2}(1+\alpha)(P(Q) - C(Q)) \tag{7.16}$$

$$u_{S_1}(h,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q)) \tag{7.17}$$

$$u_{S_1}(c,h) = u_{S_1}(c,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q'_1)) - \alpha F \tag{7.18}$$

$$u_{S_2}(h,h) = \frac{1}{2}(1+\alpha)(P(Q) - C(Q)) \tag{7.19}$$

$$u_{S_2}(c,h) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q)) \tag{7.20}$$

$$u_{S_2}(h,c) = u_{S_1}(c,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q_2')) - \alpha F \tag{7.21}$$

We can now find the α for which the expected value for S_1 is less when he cheats than when he is honest, assuming S_2 is honest. By symmetry, this will be the same for S_2. Thus, we set:

$$\frac{1}{2}(1-\alpha)(P(Q) - C(Q_1')) - \alpha F \leq \frac{1}{2}(1+\alpha)(P(Q) - C(Q))$$

This inequality is exactly the same as in theorem 7.1. Thus, letting G represent the quantity $C(Q) - C(Q_1')$, we get:

$$\frac{G}{2F + 2P(Q) + G} \leq \alpha \tag{7.22}$$

Since we can define F to be whatever we want in the contract, we can make this minimum α value arbitrarily small. If α is at least this much, then S_1 (and by symmetry, S_2) has no incentive to cheat. If S_2 is not honest, then S_1 has no incentive to be honest, but the payout is less for both (much less, if F is large). Therefore, the best outcome is for both players to behave honestly.

Now, we need to show that choosing α is incentive compatible for O. Given that both players are honest, O's utility is given as:

$$u_O(h,h) = I_v(Q(D)) - (1+\alpha)P(Q)$$

which, by our assumption, is greater than or equal to zero. Thus, it is individually rational for O. If α is increased, it merely decreases this value, so O has no incentive to increase α. If we decrease α, then S_1 and S_2 will see cheating as the more profitable choice, and will begin cheating. This leads to:

$$\begin{aligned} u_O(c,c) = \alpha(2F + I_v(Q_1') + I_v(Q_2') - c(A(Q, Q_1', Q_2'))) \\ + (1-\alpha)(\frac{1}{2}(I_v(Q_1') + I_v(Q_2')) - P(Q)) \end{aligned}$$

As in theorem 7.1, the first term of this equation decreases as α tends to zero (regardless of $c(A(Q, Q'_1, Q'_2))$, and the second term is negative. Thus, as α decreases, O's expected payout decreases. Therefore, O has no incentive to adjust α up or down, and this α is incentive compatible for O. The arguments in theorem 7.1 for the incentive compatibility of the servers and the individual rationality of both players continue to apply in this case. Thus, the contract is both incentive compatibile and individually rational for all parties involved.■

The audit process mentioned above could be done in several ways. The simplest, although most expensive, of these would be for the owner to retrieve all the data, then perform the query himself. Obviously, this defeats the purpose of data outsourcing. Based on the fact that the outsourced data uses some message authentication codes to keep the data from being modified, we can improve this. First, for selection queries, if one player fails any MAC checks, then they are obviously cheating. If one player returns fewer results than the other, then they are also obviously cheating. For aggregate queries, we can have each source return the tuples which were selected for the aggregation process. We can then check to see if the aggregate query result matches the values returned by the server. Finding a tuple set that matches a false query result might prove incredibly difficult if the false query was not generated from a sample. We can also apply the same techniques used for selection queries, noting that the cloud that returns fewer tuples must be cheating (provided all tuples returned are authenticated). Essentially, for a given query, we end up asking the providers to "show their work," or face consequences.

Note the generality of this result. In contrast with many other results, it works for **any query** on **any database** (with the caveat that the query is deterministic), and it works in only one round of computation.

7.4 The Single Cloud Case

Though the above scenario is quite simple and very efficient, it does require giving both money and data to multiple parties. It might be that the cost of maintaining two cloud services (due

to storage costs and other overhead) is expensive. A data owner might also want to minimize the outside exposure of his data set. It is possible, then, to use a similar scheme to verify the result from a single cloud. For the single cloud case, we focus on a database with a single relation. The extension to include joins will be considered in future work. We once again cast the process of query verification as a game. The game has the following characteristics:

Players (2): the Data Owner (O) and the outsourced Server(S).

Actions: The data owner begins the game by selecting a probability α, and declares this probability to the server. This probability α is the probability with which the result of the query (Q) will be verified (v). With probability $1 - \alpha$, the query will not be verified (n). After receiving this probability, the server may choose to cheat (c), revealing $q'_S = Q'(R)$, an incorrect result, or honestly (h) give the result $q_S = Q(R)$. The data owner then verifies with the probability α, first by performing a local evaluation, then, if necessary, a full query audit.

Information: O has given his database relation (R) to S, along with authenication codes for each tuple (to prevent modification). O retains a sketch of the database (R') which will be used for verification. O has a process $V(Q, q)$ which determines whether the argument q is equal to q_S with high probability, using the sketch R'. In addition, an auditing method $A(Q, q, R)$ exists which will determine whether the query was executed correctly with certainty but is very expensive. The players have entered into a contract before the game, and the contents of the contract are known to all players. This contract can adjust the payoffs below.

Payoffs: Let p_{tp} be the probability that $V(Q, q_S)$ returns true, p_{tn} be the probability that $V(Q, q'_S)$ returns false, $p_{fp} = 1 - p_{tn}$, and $p_{fn} = 1 - p_{tp}$ (These are the probabilities of true positives, true negatives, false positives and false negatives from V, respectively). Let $C(X)$ be the cost of computing the expression X. Let $I_v(X)$ be the value of the information given by X. The expected utilities (payoffs) for each player, without the intervening contract, are as follows:

When the owner decides not to verify, he simply receives the value of the query result (honest or not), minus the amount paid for the calculation, resulting in:

$$u_O(n,h) = I_v(q_S) - P(Q)$$
$$u_O(n,c) = I_v(q'_S) - P(Q)$$

Similarly, the server simply gains the amount paid, minus the cost of the calculation:

$$u_S(n,h) = P(Q) - C(q_S)$$
$$u_S(n,c) = P(Q) - C(q'_S)$$

If the owner chooses to verify, he also pays the cost of verification, and in the case of a failed V, also pays the cost of an audit. If the audit fails (which would only happen in the case of a cheating server), he does not pay the price for the calculation.

$$\begin{aligned} u_O(v,h) = I_v(q_S) - P(Q) - C(V(Q,q_S)) \\ - p_{fn} \cdot C(A(Q,q_S,R)) \\ u_O(v,c) = I_v(Q'(R)) - C(V(Q,q'_S)) \\ - p_{tn} \cdot C(A(Q,q'_S,R)) - p_{fp} \cdot P(Q) \end{aligned}$$

An honest server, in the case of the verification, gets the same payout he would without verification. This is the price of the query minus the cost to calculate it. A cheating server is only paid if he is not caught, so he is only paid in the case of a false positive from V.

$$u_S(v,h) = P(Q) - C(q_S)$$
$$u_S(v,c) = p_{fp} \cdot P(Q) - C(q'_S)$$

Now, since O declares a verification probability in advance, we can write the above as:

$$u_O(\alpha, h) = \alpha u_O(v, h) + (1 - \alpha) u_O(n, h)$$
$$u_S(\alpha, h) = \alpha u_S(v, h) + (1 - \alpha) u_S(n, h)$$
$$u_O(\alpha, c) = \alpha u_O(v, c) + (1 - \alpha) u_O(n, c)$$
$$u_S(\alpha, c) = \alpha u_S(v, c) + (1 - \alpha) u_S(n, c)$$

Note that, in practice, a payment might not be rendered for every query, and instead the server might charge a flat fee for its services, or some other payment structure. In these cases, one could consider the total payments spread out throughout the queries. This assumption that payment is rendered for each query will not invalidate our solution.

We make some assumptions about the values used above. First, we assume that $I_v(q_S) \geq P(Q) \geq C(q_S)$. This is because this inequality is necessary for participation in the game to be individually rational (since this guarantees that the best expected payoff for each player, assuming no one cheats, is at least zero). Naturally, if the query was not worth enough to the owner, he would not pay the price, and if the price did not cover the cost of computation for the server, he would not perform the calculation. Second, we assume that as q'_S approaches q_S, $C(q'_S)$ approaches $C(q_S)$. This implies that it is difficult to compute a q'_S such that $V(Q, q'_S)$ is expectedly true. As q'_S moves away from q_S, the cost can decrease. This provides the initial incentive for the server to cheat. These assumptions are logical, since computing a value close to actual result becomes more and more like computing the actual result. For example, if a cheating server were to run the query on a sample of the data and extrapolate the result, the estimated result would get more accurate as the sample size got larger, but the computational resources used would also increase. We assume that the cost of V and A are constant for a given Q (no matter if q_S or q'_S is provided to them as an argument). Finally, we once again assume that $I_v(q'_S) < 0 < I_v(q_S)$, due to the result not being what O asked for. We also assume that $C(A(Q, q'_S, R)) < I_v(q_S) - I_v(q'_S)$, since if the audit were more costly than the amount of information supplied by the query, the audit would not take place.

We now outline a contract which removes the incentive for the server to cheat. It is quite similar to the contract for the two-cloud case.

Contract 3 If the owner chooses to verify, and it is determined that the server has cheated, the server pays a penalty of $F + C(A(Q, q'_S, R))$. (Note: We explicitly force a cheating server to pay the audit cost.)

Theorem 7.3 The game, using the above contract (depicted in figure 7.3), has an equilibrium in pure strategies. O will select an α which makes cheating less profitable (expectedly) than correctly revealing the result. S chooses not to cheat.

Proof: Contract 3 makes the following changes to the payoffs of the single cloud game:

$$\begin{aligned} u_O(v,c) &= I_v(q'_S) - C(V(Q,q'_S)) \\ &\quad -p_{tn} \cdot F - p_{fp} \cdot P(Q) \\ u_S(v,c) &= p_{fp} \cdot P(Q) - C(q'_S) \\ &\quad -p_{tn} \cdot (C(A(Q,q'_S,R)) + F) \end{aligned}$$

We first begin with the above equations for u_O and u_S in terms of α. We want to find the α such that $u_S(\alpha, h) \geq u_S(\alpha, c)$. Substituting in, we get:

$$\begin{aligned} u_S(\alpha,h) &= \alpha(P(Q) - C(q_S)) \\ &\quad +(1-\alpha)(P(Q) - C(q_S)) \\ &= P(Q) - C(q_S) \\ u_S(\alpha,c) &= \alpha(p_{fp}P(Q) - C(q'_S) \\ &\quad -p_{tn}(C(A(Q,q'_S,R)) + F)) \\ &\quad +(1-\alpha)(P(Q) - C(q'_S)) \end{aligned}$$

Multiplying through by α and $1-\alpha$:

$$\begin{aligned} P(Q) - C(q_S) &\geq \alpha p_{fp}P(Q) - \alpha C(q'_S) \\ &\quad -\alpha p_{tn}(C(A(Q,q'_S,R)) + F) \\ &\quad +P(Q) - C(q'_S) - \alpha P(Q) \\ &\quad +\alpha C(q'_S) \end{aligned}$$

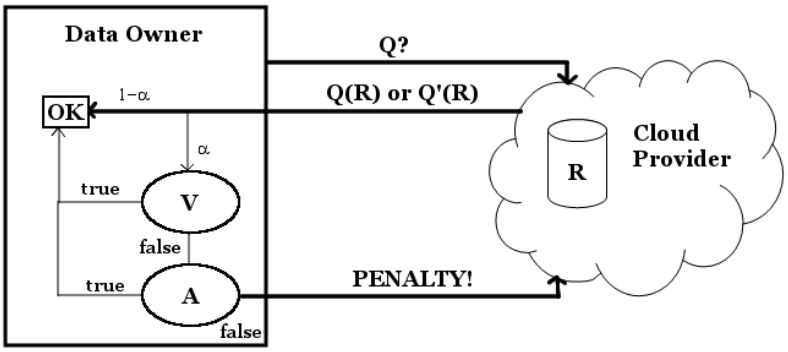


Figure 7.3. Verifying Queries With a Single Cloud

$$C(q'_S) - C(q_S) \geq \alpha(p_{fp}P(Q) - C(q'_S) - P(Q) + C(q'_S)) \\ - p_{tn}(C(A(Q, q'_S, R)) + F)$$

Canceling out like terms, we get:

$$C(q'_S) - C(q_S) \geq \alpha(p_{fp} - 1)P(Q) \\ - \alpha p_{tn}(C(A(Q, q'_S, R)) + F)$$

Now, since Q' is easier to compute than Q, $p_{fp} < 1$, both sides of this inequality are negative. We therefore multiply both sides by -1 and simplify to get the following:

$$C(q_S) - C(q'_S) \leq \alpha((1 - p_{fp})P(Q) \\ + p_{tn}(C(A(Q, q'_S, R)) + F))$$

Since $1 - p_{fp}$ is equal to p_{tn}, we can substitute in p_{tn}, then divide by the coefficient of α, yielding the final expression:

$$\alpha \geq \frac{C(q_S) - C(q'_S)}{p_{tn}(C(A(Q, q'_S, R)) + F + P(Q))}$$

When α increases, the payout for cheating decreases, provided $C(A(Q, q'_S, R))$ and F are large enough. So, as long as the expression above is satisfied, the server will choose not to cheat.

Now, while $C(A(Q, q'_S, R))$ is fixed, F is something that can be adjusted in the contract. Therefore, if the penalty F is astronomically high, we can severely reduce α, while maintaining that there is no incentive to cheat for S. This is what is known as a "boiling-in-oil" contract (Rasmusen 2007).

We must also show that this α is incentive compatible for O. Consider what happens when α is increased. If α is greater than the above value, O ends up verifying more, while S continues telling the truth. Because of this, O loses valuation. So, O will not choose α higher than this. If α is less than this value, then S will start cheating. The possible increase in payout to O would be αF, but since α is small, and $I_v(q_S)$ is so much greater than $I_v(q'_S)$, this would likely result in a decrease in payout for O. Therefore, α is not less than the above expression either. Thus, we have an equilibrium. ∎

7.5 Implementation Details

The game outlined above is fairly general, and allows for any local verification method V to be used. Here, we outline a simple sampling verification method which becomes much more viable when the verification process is not being run with every query. First, let us assume that the data consists of N signed tuples, each of which has a unique, consecutive id from 1 to N. Let O maintain some sample of size k of these N tuples, together with the value N. This sample is selected uniformly at random from the entire data set, with replacement. This sample can be used to compute $V(Q, q'_S)$ for many different types of queries. For aggregate queries such as count, sum, average, standard deviation, etc., one could simply perform the

action on the k tuples, and extrapolate based on N. If this sample value is within some ε of the query result q'_S, then we declare the result correct. Otherwise, we perform the audit.

For selection queries, note that because each tuple is signed, we know that the server cannot modify any tuples, nor can it insert new tuples. It can only either remove relevant tuples from the result, or insert irrelevant tuples into the result. If the server inserts irrelevant tuples, this can be easily verified by O by simply noting that the tuple does not match the query. Thus, it is only difficult to verify when a tuple has been left out. As before, we can perform the selection query on the sample of k tuples, and extrapolate the number of tuples that should be returned by q_S. If the number of tuples in q'_S is within some ε, we declare the result correct. If the number of tuples in q'_S is greater than our estimate, then we should also declare the result correct, since a greater number of tuples cannot be wrong. Otherwise, we perform the audit.

There are plenty of other methods used to verify queries on outsourced data, and **any** of them would work as a verification method V in our scheme. We choose this one, however, because of its simplicity. Note that it does not require expensive cryptographic operations.

One thing remains in the definition of the verification mechanism, and that is the definition of ε. As the selection of k tuples can be considered a selection of k random variables $X_1, ..., X_k \in R$, and in each case we are interested in a function f which maps $R \rightarrow \Re$, and any alteration in a given X_i can only change the value of the aggregate function by at most some c_i (this c_i is 1 for count, the max value of the given attribute for sum, the max value squared for standard deviation, etc), we can apply McDiarmid's inequality (McDiarmid 1989), giving us the following:

$$Pr\{|E[f(X_1, ..., X_k)] - f(X_1, ..., X_k)| \geq \varepsilon\} \leq 2e^{-\frac{2\epsilon^2}{\sum_{i=1}^{k} c_i^2}}$$

Note, this inequality does not depend on the value of N. It simply depends on the sample size. For example, say we want to devise a sample size k such that the probability that an average query on attribute a of the sample is within $\varepsilon = 1\%$ of the true result with probability .999. c_i is given as $\frac{\max\{|a|\}}{k}$. The probability in the above works out to $2e^{-\frac{.0002\text{average}\{a\}^2}{\max\{|a|\}^2/k}}$. We

want this to be less than or equal to .001. Solving for k, we get

$$ln(.0005) \leq -\frac{.0002\text{average}\{a\}^2 k}{max\{|a|\}^2}$$

$$\frac{-ln(.0005)max\{|a|\}^2}{.0002\text{average}\{a\}^2} \leq k$$

This gives us a k value of approximately 38004.51 times the maximum value of the attribute a, divided by the square of the result. As the average of the result is no more than the maximum value of a, and its square can be much larger, k can be 38 thousand tuples, or less, depending on the distribution of a, even if the number of tuples is in the millions. Note that this is does not help the data owner find the value of k, as the owner does not know the actual result. This merely shows that a good k exists, and it is independent of the number of tuples in the dataset for many common queries.

38,000 tuples is not a particularly small number, especially with some sketches using only three bytes (Yi, Li, Cormode, Hadjieleftheriou, Kollios, and Srivastava 2009). However, this sample can be used to verify many different types of queries, and does not have to plan for the queries in advance. In addition, the verification will only be performed a fraction (α) of the time. This fraction, through the use of the penalty in the contract, can be made arbitrarily small, leading to a very fast expected runtime.

Now, one method of generating a false query (for aggregate queries) might be to use the same sampling method as above. Note, however, that in order to ensure that the sample chosen by the server has a result within ε of the result of the owner's sample, the server would need a much tighter epsilon.

With some probability δ, the owner's sample result is within ε of the correct result. This is also true for the server. However, consider the worst case where the owner's sample value is $Q(R) - \varepsilon$ and the server's sample value is $Q(R) + \varepsilon$. The probability δ is not sufficient to bound the difference between the two sample values. To ensure that with probability δ the owner's sample result is within ε of the result returned by the server (with the given high probability), the server would have to return the actual result, as any leeway would lead to a worst case scenario where the difference is greater than ε.

In order to prevent sampling bias, a protocol could be implemented to resample from the data. As each tuple has a unique id, the owner could, at some interval, request the tuples with a given set of ids. The owner would know if the tuples he desired were not returned. In order to prevent the server from learning the exact sample (which would lead to the server simply using that sample for calculations), the owner would select some dummy tuples, or in some cases, the entire data set. A similar method, selecting all tuples involved, can be employed for auditing the queries.

7.6 Experiments

To test the effectiveness of the sampling protocol for catching cheating on real data, we ran a series of experiments. The mechanisms outlined in sections 3 and 4 do **not** need any experimentation, as they are proven and mathematically sound. These experiments were designed to show that the sampling technique can identify cheating with a non-trivial probability. Other verification methods will work similarly in our framework, as long as they can identify cheating with non-trivial probability. For example, if a simple method exists to verify a certain query deterministically, then it could be called in place of our sampling scheme, and would allow our α to be even smaller. The sampling protocol is important, however, due to its generality and simplicity.

7.6.1 Methodology

We used the US Census 1990 data set (census-income) from the UC Irvine Machine Learning repository, which contains over 2.3 million tuples (Asuncion and Newman 2007). This is again the same data set used in the two previous chapters. We focused on a few major fields in this data set.

We processed results for eight different aggregation queries of varying types. Since selection queries can be estimated via counts, we chose to focus on aggregation queries.

The query types are as follows:

1. Count, equality selection (count the people whose race value is 2–black)
2. Count, range selection (count the people whose income is greater than 40000)
3. Count, range and equality conjunction (count the people who are over age 30 and never married)
4. Count, range disjunction (count the people who are under age 18 or have an income of less than 10000)
5. Sum, equality selection (find the sum of the incomes of all people who never married)
6. Sum, range and equality conjunction (find the sum of the incomes of all people who are over age 40 and whose place of birth is the place they work)
7. Average, range selection (find the average age of all people who have an income greater than 80000)
8. Average, equality conjunction, sparse result (find the average income of all people who are male and of race 9–Japanese)

For each query, we ran 100 trials, estimating the full result of the query with five different sample sizes: 1000 tuples, 5000 tuples, 10000 tuples, 20000 tuples, and 40000 tuples. As above, these samples are selected uniformly at random with replacement. We determined the likelihood that each sample would accept the correct value for varying values of ε from 0 to $.5r$ where r is the estimated result. Since the verification process would not know the actual result, we based the ε on the estimated result given by the sample, as we expected it to be close to the actual result.

We then ran the samples against different means of falsifying the result, to determine if the sample method could catch a cheater. The first type of falsification was the same as our verification technique, sampling the data. We once again ran 100 trials for 1000, 5000, 10000, 20000, and 40000 tuple samples, both for detecting the cheating and for creating the cheating.

The second type of falsification was a "worst case" falsification, where the exact result was computed, but then Laplace noise was added to the final result. An adversary would never actually do this, as it would be more expensive than simply computing the result itself, but this provides a way to test our scheme beyond the normal means. The mean of the Laplace noise was of course the result itself (which we will call r), whereas the width parameter was varied from $r/5$, $r/10$, $r/20$, and $r/50$. We chose Laplace noise as opposed to any other type of noise because it is used in differential privacy as a means of masking query results while still achieving meaningful results (Dwork 2006). Each of these sets of noise ran 100 trials against each sample size as before.

7.6.2 Results

Space restrictions prevent the inclusion of the hundreds of graphs generated by the experiments. However, if we examine one factor at a time, we can show the general trend of the sampling protocol to correctly or incorrectly identify cheating values. The omitted graphs show similar trends.

Query Type. Figure 7.4 shows the ROC (receiver operating characteristic) curves for each of the eight queries, for a sample size of 10000, against every type of result falsification we used. These ROC curves shows the tradeoff between the probability of a false negative and the probability of a true negative. The queries themselves all behave similarly. At a sample size of 10000, we can always find an ε where some nontrivial fraction of cheating will be caught. There is always a tradeoff, however. As ε decreases, more legitimate results will be marked as wrong, and forced to be fully audited. Proper use of the sampling technique involves careful selection of epsilon in order to increase p_{tn}, while reducing p_{fn} as much as possible. In practice, it is more important that p_{tn} be high, since we can mitigate the effect of false positives by increasing the penalty, thereby decreasing α and reducing the number of times that we do the verification. p_{tn} is acceptably high for fairly small ϵ ($0.00125r$ to $0.005r$).

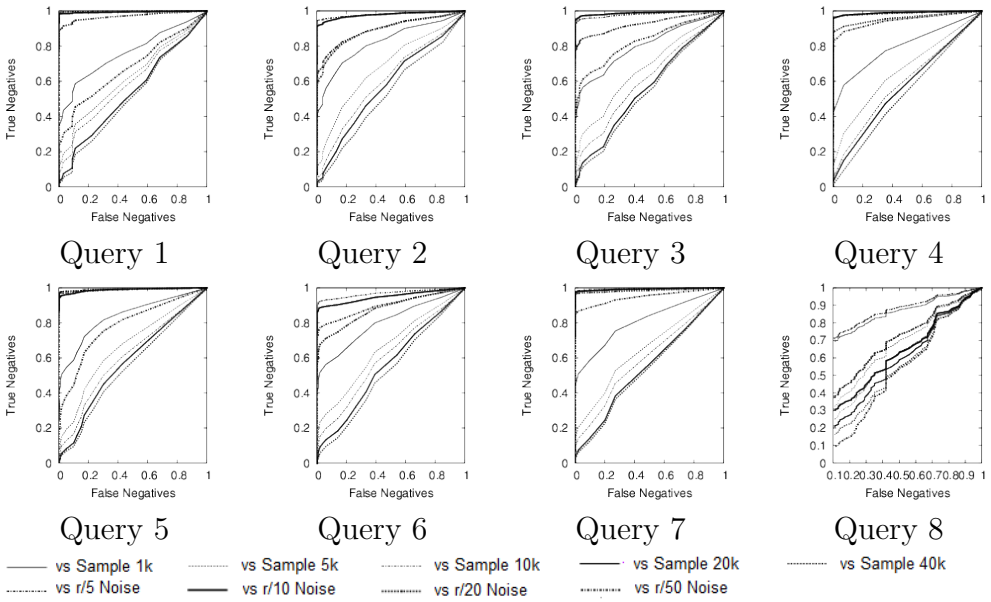


Figure 7.4. ROC Curves for the 8 Query Types: Sample Size 10000

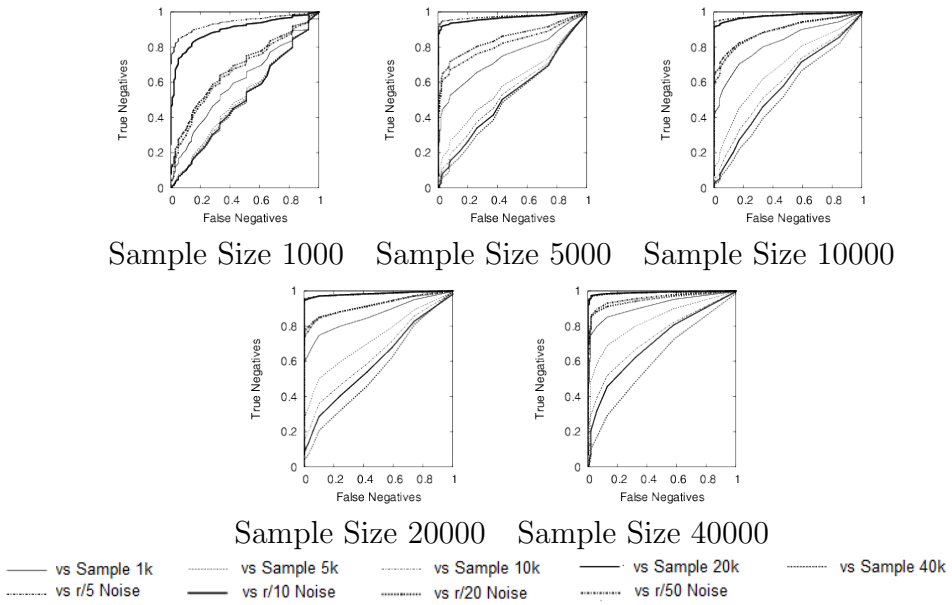


Figure 7.5. ROC Curves for Five Sample Sizes: Query 2

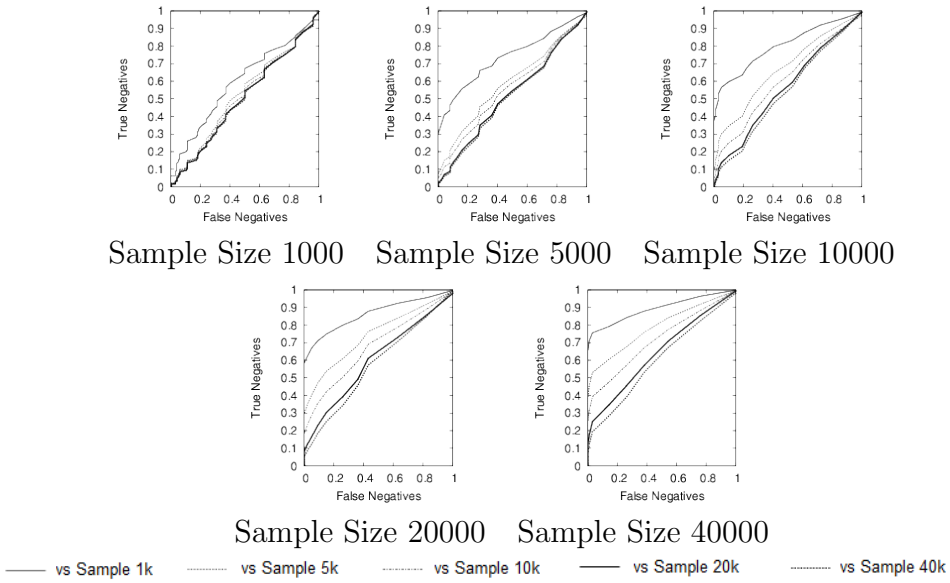


Figure 7.6. ROC Curves for Five Sample Sizes: Query 3, Sampling Cheater Only

Sample Size. Figure 7.5 shows the ROC curves for query 2 for each sample size used. Clearly, as the sample size increases, the potential for better choices of epsilon increases. With a sample size of 40000, there is even a type of cheating ($r/5$ noise) that allows for a false negative rate of .02 and a true negative rate of .95. The obvious tradeoff here, though, is that while you will do fewer full audits with a larger sample size, the verification process will take more resources. The smaller sample sizes still have the ability to catch cheating, but they will end up auditing many more legitimate results.

Cheater Sample Size. Figure 7.6 shows the ROC curves for query 3, against cheaters using sampling only. We can clearly see that the curves move down and to the right as our adversary's sample size increases. This makes sense, as the cheater gets better at impersonating the correct result, it becomes more difficult to distinguish the incorrect results from the correct ones. However, in every case, even with a sample size of 1000, we are able to detect cheating better than random guessing. Keep in mind that, in order to be useful, we merely

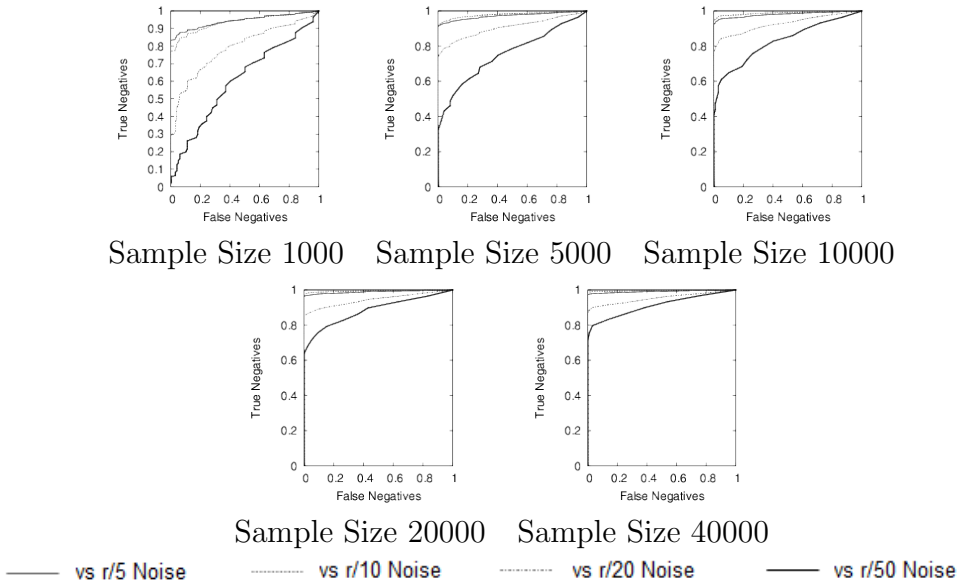


Figure 7.7. ROC Curves for Five Sample Sizes: Query 3, Noisy Cheater Only

need to be able to detect cheating with some non-negligible probability, and that any means we choose to do that is acceptable.

Cheater Laplace Noise. Figure 7.7 shows the same query as figure 7, but this time with our cheater only using the Laplace noise. Surprisingly, the noise addition version ends up being easier to catch. This is due to the fact that the parameter on the Laplace noise is large enough to cause issues. Still, at $r/50$, the cheating becomes quite difficult to detect for low sample size.

7.6.3 Conclusions

In summary, by thinking about the problem of query verification from a different perspective, namely, that of an economist, we can drastically reduce the computation required to ensure that the result asked for is the result received. Using the game-theoretic framework outlined here, we show that using a multiple servers, contracts can be designed that will ensure that results obtained from an outsourced computation service are genuine, while requiring only a

fractional increase in cost. This is, of course, in contrast to most methods of query verification, which rely on complicated security technologies. The various query verification technologies that are out there are still quite useful, however. Specialized verification methods which take up very little space work well for common queries. They are, however, not generic and can rely on some expensive operations. The outside-the-box approach of using a redundant data service for verification vastly simplifies this process, and incurs a minimal cost.

In addition, we can similarly achieve a small cost using only one cloud provider. Note that our verification method V does not have to be the one that we used in the experiments. Existing verification methods could also be used, and if they are used, will only need to be used a small fraction of the time.

CHAPTER 8
CONCLUSIONS

We have demonstrated various methods to improve data sharing in several cases. First, we used approximations to improve the runtime of privacy-preserving data mining protocols. We then used the tools of game theory to enforce honest behavior in secure data mining, data outsourcing, and multiparty computation.

We demonstrated methods for approximating the scalar product of two vectors which can then, in turn, be used to reduce the time complexity of several data mining tasks. We proved the approximation secure under our definition, and we showed that the methods work in practice through experimentation. In the best case, we were able to achieve two orders of magnitude improvement on the initial time complexity.

We created game theoretic frameworks for enforcing honesty in data mining. The first, more efficient framework, involved the Vickrey-Clarke-Groves mechanism and worked when players did not collude. The second worked even when players collude, but is far more computationally intensive. These mechanisms, however, enforce an outcome which cryptographic protocols alone cannot enforce. This outcome is truthful behavior.

We then turned our game theory to the subject of data outsourcing. We showed several mechanisms which will deter cheating on the part of the outsourced server. The first two involve checking query results against a second outsourcing service. The third involves the use of query verification methods. In all cases, we showed that we can make the probability that we execute a query verification method arbitrarily small, and showed a simple verification method which becomes practical when used in this framework.

Finally, we generalized the notion set forth in chapter 5 to multiparty computation. We showed that there exists a class of functions which allow us to enforce honesty, and we also

showed that some of the computation can be done outside the trusted server (although the final evaluation still needs to be done in a trusted fashion). These various methods show that with some outside-the-box thinking, be it with game theory or vector projections, secure data sharing can be improved.

REFERENCES

Abraham, I., D. Dolev, R. Gonen, and J. Halpern (2006). Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pp. 53–62. ACM New York, NY, USA.

Achlioptas, D. (2003). Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences 66*(4), 671–687.

Aggarwal, C. and P. Yu (2008). A general survey of privacy-preserving data mining models and algorithms. *Privacy-preserving data mining*, 11–52.

Agrawal, R. and R. Srikant (2000). Privacy-preserving data mining. In *ACM Sigmod Record*, Volume 29, pp. 439–450. ACM.

Agrawal, R. and E. Terzi (2006). On honesty in sovereign information sharing. *Lecture Notes in Computer Science 3896*, 240.

Annas, G. (2003). HIPAA Regulations–A New Era of Medical-Record Privacy? *The New England Journal of Medicine 348*(15), 1486.

Ashlagi, I., A. Klinger, and M. Tenneholtz (2007). K-NCC: Stability Against Group Deviations in Non-Cooperative Computation. *LECTURE NOTES IN COMPUTER SCIENCE 4858*, 564.

Asuncion, A. and D. Newman (2007). UCI Machine Learning Repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html, Accessed December 2010.

Atallah, M., Y. Cho, and A. Kundu (2008). Efficient data authentication in an environment of untrusted third-party distributors.

Aumann, R. (1961). The core of a cooperative game without side payments. *Transactions of the American Mathematical Society 98*(3), 539–552.

Bloom, B. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM 13*(7), 422–426.

Broder, A. and M. Mitzenmacher (2004). Network applications of bloom filters: A survey. *Internet Mathematics 1*(4), 485–509.

Canetti, R., B. Riva, and G. Rothblum (2011). Practical delegation of computation using multiple servers. In *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 445–454. ACM.

Chen, H., X. Ma, W. Hsu, N. Li, and Q. Wang (2008). Access control friendly query verification for outsourced data publishing. *Computer Security-ESORICS 2008*, 177–191.

Clifton, C., M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu (2002). Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter 4*(2), 28–34.

Dean, J. and S. Ghemawat (2008, January). Mapreduce: simplified data processing on large clusters. *Commun. ACM 51*(1), 107–113.

Dekel, O., F. Fischer, and A. Procaccia (2008). Incentive compatible regression learning. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 884–893. Society for Industrial and Applied Mathematics.

Du, W. and M. Atallah (2001). Privacy-preserving cooperative statistical analysis. In *Proceedings of the 17th Annual Computer Security Applications Conference*, pp. 102. IEEE Computer Society.

Dwork, C. (2006). Differential privacy. *Automata, languages and programming*, 1–12.

Dwork, C. (2008). Differential privacy: A survey of results. *Theory and Applications of Models of Computation*, 1–19.

Dwork, C., K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor (2006). Our data, ourselves: Privacy via distributed noise generation. *Advances in Cryptology-EUROCRYPT 2006*, 486–503.

Fatima, S., M. Wooldridge, and N. Jennings (2008). A linear approximation method for the shapley value. *Artificial Intelligence 172*(14), 1673–1699.

Feigenbaum, J., Y. Ishai, T. Malkin, K. Nissim, M. Strausse, and R. Wright (2006). Secure multiparty computation of approximations. *ACM transactions on Algorithms (TALG) 2*(3), 435–472.

Fradkin, D. and D. Madigan (2003). Experiments with random projections for machine learning. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 517–522. ACM.

Gennaro, R., C. Gentry, and B. Parno (2010). Non-interactive verifiable computing: Outsourcing computation to untrusted workers. *Advances in Cryptology–CRYPTO 2010*, 465–482.

Goethals, B. (2005). Frequent Itemset Mining Implementations Repository. http://fimi.ua.ac.be/data/, Accessed June 2011.

Goethals, B., S. Laur, H. Lipmaa, and T. Mielikainen (2005). On private scalar product computation for privacy-preserving data mining. *Information Security and Cryptology–ICISC 2004*, 104–120.

Google (2011). Google BigQuery Service. http://code.google.com/apis/bigquery, Accessed September 2012.

Gordon, S. and J. Katz (2006). Rational secret sharing, revisited. *Lecture Notes in Computer Science 4116*, 229.

Haber, S., W. Horne, T. Sander, and D. Yao (2006). Privacy-preserving verification of aggregate queries on outsourced databases. Technical report, Citeseer.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA Data Mining Software: An Update. http://www.cs.waikato.ac.nz/ẽibe/pubs/weka_update.pdf, Accessed July 2011.

Halpern, J. and V. Teague (2004). Rational secret sharing and multiparty computation: extended abstract. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 623–632. ACM New York, NY, USA.

Hoeffding, W. (1965). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58*(301), 13–30.

Huang, Z., W. Du., and B. Chen (2005). Deriving private information from randomized data. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pp. 37–48. ACM.

Ioannidis, I., A. Grama, and M. Attallah (2002). A secure protocol for computing the dot-products in clustered and distributed environments. In *International Conference on Parallel Processing, 2002.*, pp. 379–384. IEEE.

Islam, M. and L. Brankovic (2003). Noise Addition for Protecting Privacy in Data Mining. In *Proceedings of The 6th Engineering Mathematics and Applications Conference (EMAC2003), Sydney*, pp. 85–90. Citeseer.

Izmalkov, S., S. Micali, and M. Lepinski (2005). Rational secure computation and ideal mechanism design. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pp. 585–594.

Jiang, W., C. Clifton, and M. Kantarcıoğlu (2008). Transforming semi-honest protocols to ensure accountability. *Data & Knowledge Engineering 65*(1), 57–74.

Johnson, W. and J. Lindenstrauss (1984). Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics 26*(189-206), 1–1.

Kantarcioglu, M. and C. Clifton (2004). Privacy-preserving distributed mining of association rules on horizontally partitioned data. *Knowledge and Data Engineering, IEEE Transactions on 16*(9), 1026–1037.

Kantarcioglu, M. and R. Nix (2010). Incentive compatible distributed data mining. In *SocialCom/PASSAT*, pp. 735–742.

Kantarcioglu, M., R. Nix, and J. Vaidya (2009). An efficient approximate protocol for privacy-preserving association rule mining. *Advances in Knowledge Discovery and Data Mining*, 515–524.

Kantarcioglu, M. and J. Vaidya (2003). Privacy preserving naive bayes classifier for horizontally partitioned data.

Kargupta, H., K. Das, and K. Liu (2007). Multi-party, privacy-preserving distributed data mining using a game theoretic framework. *Knowledge Discovery in Databases: PKDD 2007*, 523–531.

Kargupta, H., S. Datta, Q. Wang, and K. Sivakumar (2003). On the privacy preserving properties of random data perturbation techniques. In *ICDM 2003. Third IEEE International Conference on Data Mining, 2003.*, pp. 99–106. IEEE.

Katz, J. (2008). Bridging game theory and cryptography: Recent results and future directions. *Lecture Notes in Computer Science 4948*, 251.

Kol, G. and M. Naor (2008). Cryptography and game theory: Designing protocols for exchanging information. *Lecture Notes in Computer Science 4948*, 320.

Layfield, R., M. Kantarcioglu, and B. Thuraisingham (2009). Incentive and Trust Issues in Assured Information Sharing. In *Collaborative Computing: Networking, Applications and Worksharing: 4th International Conference, CollaborateCom 2008, Orlando, FL, USA, November 13-16, 2008, Revised Selected Papers*, pp. 113. Springer.

Li, P., T. Hastie, and K. Church (2006). Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 287–296. ACM.

Lindell, Y. and B. Pinkas (2000). Privacy preserving data mining. In *Advances in Cryptology-CRYPTO 2000*, pp. 36–54.

Liu, K., C. Giannella, and H. Kargupta (2006). An attackers view of distance preserving maps for privacy preserving data mining. *Knowledge Discovery in Databases: PKDD 2006*, 297–308.

Liu, K., H. Kargupta, and J. Ryan (2006). Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on Knowledge and Data Engineering*, 92–106.

Lysyanskaya, A. and N. Triandopoulos (2006). Rationality and adversarial behavior in multi-party computation. *Lecture Notes in Computer Science 4117*, 180–197.

Machanavajjhala, A., D. Kifer, J. Gehrke, and M. Venkitasubramaniam (2007). l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD) 1*(1), 3.

Manyika, J., M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Byers (2011). Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute*, 1–137.

McDiarmid, C. (1989). On the method of bounded differences. *Surveys in combinatorics 141*(1), 148–188.

McGrew, R., R. Porter, and Y. Shoham (2003). Towards a general theory of non-cooperative computation. In *Proceedings of the 9th conference on Theoretical aspects of rationality and knowledge*, pp. 59–71. ACM New York, NY, USA.

Menezes, A., P. Van Oorschot, and S. Vanstone (1997). *Handbook of applied cryptography*. CRC.

Merkle, R. (1979). Secrecy, authentication, and public key systems.

Moulin, H. (1992). An application of the Shapley value to fair division with money. *Econometrica: Journal of the Econometric Society 60*(6), 1331–1349.

Mykletun, E., M. Narasimha, and G. Tsudik (2006). Authentication and integrity in outsourced databases. *ACM Transactions on Storage (TOS) 2*(2), 107–138.

Nash, J. (1951). Non-cooperative games. *The Annals of Mathematics 54*(2), 286–295.

National (2004, August). *The 9/11 Commission Report: Final Report of the National Commission on Terrorist Attacks Upon the United States (Indexed Hardcover, Authorized Edition)*. W. W. Norton & Company.

National Institute of Standards and Technology (2002, August). FIPS 180-2, secure hash standard, federal information processing standard (FIPS), publication 180-2. Technical report, DEPARTMENT OF COMMERCE.

Nisan, N. (2007). Introduction to mechanism design (for computer scientists). *Algorithmic Game Theory*, 209–242.

Nix, R. and M. Kantarcioglu (2012a). Contractual agreement design for enforcing honesty in cloud outsourcing. In *GameSec*.

Nix, R. and M. Kantarcioglu (2012b). Incentive compatible privacy-preserving distributed classification. *IEEE Transactions on Dependable and Secure Computing. ©2012 IEEE. Reprinted with permission. 9*(4), 451–462.

Nix, R., M. Kantarcioglu, and K. J. Han (2012). Approximate privacy-preserving data mining on vertically partitioned data. In *DBSec*, pp. 129–144.

Ong, S., D. Parkes, A. Rosen, and S. Vadhan (2009). Fairness with an honest minority and a rational majority. In *Sixth Theory of Cryptography Conference (TCC)*. Springer.

Pang, H., A. Jain, K. Ramamritham, and K. Tan (2005). Verifying completeness of relational query results in data publishing. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pp. 407–418. ACM.

Pang, H., J. Zhang, and K. Mouratidis (2009). Scalable verification for outsourced dynamic databases. *Proceedings of the VLDB Endowment 2*(1), 802–813.

Patel, P., A. Ranabahu, and A. Sheth (2009). Service level agreement in cloud computing. In *Cloud Workshops at OOPSLA*.

Pinkas, B. (2002). Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations Newsletter 4*(2), 19.

Pub, F. (2002). 198, the keyed-hash message authentication code (hmac). *Federal Information Processing Standards Publication 198*.

Rasmusen, E. (2007). *Games and information: An introduction to game theory*. Wiley-blackwell.

Ravikumar, P., W. Cohen, and S. Feinberg (2004). A secure protocol for computing string distance metrics. In *Proceedings of the Workshop on Privacy and Security Aspects of Data Mining at the International Conference on Data Mining*, pp. 40–46. IEEE.

Rivest, R. (1992). The MD5 Message-Digest Algorithm. *Internet Request For Comments 1321*.

Shapley, L. (1952). A Value for n-Person Games. *Contributions to the Theory of Games II*, 307–317.

Shoham, Y. and M. Tennenholtz (2005). Non-cooperative computation: Boolean functions with correctness and exclusivity. *Theoretical Computer Science 343*(1-2), 97–113.

Sion, R. (2005). Query execution assurance for outsourced databases. In *Proceedings of the 31st international conference on Very large data bases*, pp. 601–612. VLDB Endowment.

Sion, R. (2007). Secure data outsourcing. In *Proceedings of the 33rd international conference on Very large data bases*, pp. 1431–1432. VLDB Endowment.

Sweeney, L. (2002). k-Anonymity: A Model For Protecting Privacy. *World 10*(5), 557–570.

Tan, X. and T. Lie (2002). Application of the Shapley Value on transmission cost allocation in the competitive power market environment. In *Generation, Transmission and Distribution, IEE Proceedings-*, Volume 149, pp. 15–20. IET.

Vaidya, J. and C. Clifton (2002). Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 639–644. ACM.

Vaidya, J. and C. Clifton (2004). Privacy preserving naıve bayes classifier for vertically partitioned data. In *2004 SIAM International Conference on Data Mining, Lake Buena Vista, Florida*, pp. 522–526.

Vaidya, J. and C. Clifton (2005a). Privacy-preserving decision trees over vertically partitioned data. *Data and Applications Security XIX*, 924–924.

Vaidya, J. and C. Clifton (2005b). Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security 13*(4), 593–622.

Von Neumann, J. and O. Morgenstern (1967). *Theory of games and economic behavior*. John Wiley & Sons Inc.

Wang, W., M. Garofalakis, and K. Ramchandran (2007). Distributed sparse random projections for refinable approximation. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pp. 331–339. ACM.

Xiao, X. and Y. Tao (2007). M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pp. 689–700. ACM.

Xie, M., H. Wang, J. Yin, and X. Meng (2007). Integrity auditing of outsourced data. In *Proceedings of the 33rd international conference on Very large data bases*, pp. 782–793. VLDB Endowment.

Xu, J. and E. Chang (2010). Authenticating aggregate range queries over multidimensional dataset. Technical report, Cryptology ePrint Archive, Report 2010/050.

Yang, Y., D. Papadias, S. Papadopoulos, and P. Kalnis (2009). Authenticated join processing in outsourced databases. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pp. 5–18. ACM.

Yi, K., F. Li, G. Cormode, M. Hadjieleftheriou, G. Kollios, and D. Srivastava (2009). Small synopses for group-by query verification on outsourced data streams. *ACM Transactions on Database Systems (TODS) 34*(3), 1–42.

Zhang, N. and W. Zhao (2005). Distributed privacy preserving information sharing. In *Proceedings of the 31st international conference on Very large data bases*, VLDB '05, pp. 889–900. VLDB Endowment.

VITA

Robert Nix was born in Anson, Texas, in 1983. He has, since then, grown to become a nerd of Biblical proportions. He has received a best paper award for the work in chapter five of this very dissertation, and will hopefully be a professor by the beginning of January. For those of you keeping score at home, his tuna salad has graduated from mean to surly.